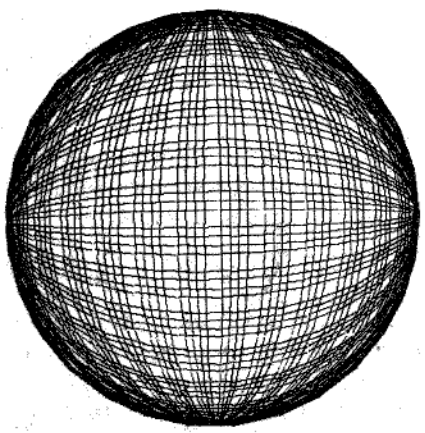
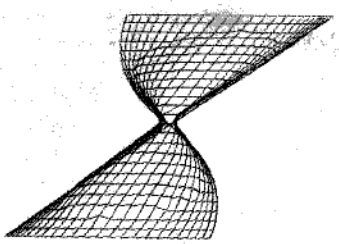
PROGRAM nittinian BITEN 84-1





Innehåll

Redaktören...

Vårt första nummer i år har nu kommit ut. Med stor hjälp från Björn Gustavsson och vår "allt-i-allo", Claes Schibler, håller du nu Programbiten i handen, den nya redaktionens allra första nummer.
Vi vet att ägare av TI99/4A finner nöje i detta och de som har en TI59 glädjs åt att Programbiten fortfarande existerar.
Detta nummer tar upp början på en Assembler-skola, skriven av våra duktiga medarbetare och en fortsättning av artikeln om Forth, den fjärde generationens programmerings-språk, vilken lovar gott för framtiden

Givetvis tar vi upp några nya Basic program, exempelvis hur du skapar koder för Sprites. Några program för TI59 finns också till nytta för er som äger en räknare.

Vi har ju en "brevlåda" i tidningen som heter INSÄNT. Där du som läsare kan fråga om datorer, räknare och program.
Vad händer OM...? Dessa "bugs and error". Vi kommer att svara dig efter bästa förmåga, ge dig tips och hjälp med hänvisningar om så fordras. Skicka dina frågor till:

Göran Nygren
Uppingegränd 10
163 61 SPÅNGA

DU!
Som har program och ideer. Och vill ta upp det du anser är bra tips och knep. Skicka in dina program och tankar till:

Föreningen Pogrambiten
c/o Schibler
Wahlbergsgatan 6 nb.
121 46 JOHANNESHOV

Vi önskar dig en skön sommar !

Göran Nygren

I REDAKTIONEN

Redaktör	Göran Nygren
Biträdande redaktör	Michael öhman
Spelredaktör	Peter Olin
Utmaningsredaktör	Anders Persson
Programförmedlare	Björn Mårtensson

Föreningens och redaktionens adress :

Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 6 nb
121 46 JOHANNESHOV

MONTERING DETTA NUMMER :

Allt i allot
Göran Nygren
Michael öhman
Tony Rogvall

TECKNINGAR : LARS EKEROTH
Allt i allot

ANVÄNDARTIPS MED MINI MEMORY

Denna bok skrevs av Björn Gustavsson på uppdrag av Texas Instruments. Precis när boken var klar lade TI ner hemdatorn. Föreningen fick då rätten till boken och har tryckt upp den. Boken innehåller 60 sidor och kostar för medlemmar 60 SEK som sätts in på föreningens postgiro 19 83 00-6.

VIKTIGT MEDDELANDE

Föreningen kan från och med nr 84-1 inte lämna någon ersättning för artiklar och program enligt notis i Nittinian nr 2.

FRISTÅENDE RS232
via FÖRENINGEN

SLUT

Tryck: K-TRYCK AB STOCKHOLM

ORDFÖRANDEN har ORDET

Tack för brev!

En hel del både medlemmar och andra har hört av sig. Det är vi glada för. Vi i styrelsen och även andra svarar så gott vi kan och hinner. Somt har blivit liggande för länge - jag vet. Vissa svar av allmänt intresse kanske kommer i form av artiklar. Vi vill ha fler brev!

För att underlätta kontakterna direkt mellan medlemmar sänder vi med tidningen den här gången en medlemsmatrikel, sorterad i postnummerordning. Tag fatt i Din granne som har en 99-a! Träffas och prata dator!

För de medlemmar som bor i Stockholm - eller som av andra skäl är i Stockholm lördagen den 15 september - ordnar styrelsen en träff med start kl 14 och längst till kl 18. Vi söker lokal i centrala Stockholm. Boka tiden! Jag återkommer med detaljer om programmet. Skriv gärna och ge ideer!

"99-er Magazine" - den amerikanska tidningen för 99-ägare - har omorganiserat sig efter TI:s nedläggning av 99-an. Den har nu kommit i ny skepnad och heter då "Home Computer Magazine" - HCM. Den verkar mycket matnyttig och anmäls på annan plats här i tidningen.

Det verkar också som om Texas Instruments känner ansvar för sina slutkonsumenter - i det här fallet för oss 99-ägare - som man tidigare hävdat. Så kan man nämligen se på det faktum att TI i USA fr o m 31 mars överlåtit lagret och rätten att tillverka och saluföra 99-artiklar på ett separat bolag. Något liknande har hänt i Sverige, där en tidigare TI-medarbetare bildat ett företag som fått samma rättigheter. Saken anmäls på annan plats i tidningen. Vi får hoppas att de här företagen är seriösa och gör vad vi väntar av dem - dvs fortsätter att tillhandahålla de program och prylar vi vill ha.

99-an lever alltså!
När du bläddrar i matrikeln finner Du antagligen att den inte alls är så tjock som den mångtusenskattade försäljningen i Sverige ger underlag för. Värva! Varhelst Du anar en 99-hacker eller - vilket jag tror tyvärr förekommer - en 99-a som står undanställd och outnyttjad: presentera föreningen, giv hopp om gemenskap och vidareutveckling, förklara att ryktet om vår död är betydligt överdrivet osv. Du kan ju t ex locka med vidgade internationella kontakter Detta är nämligen vår plan.

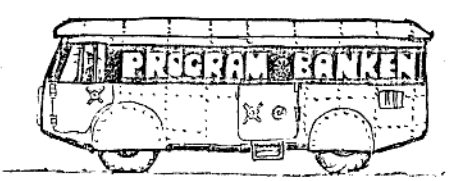
Via sådana kontakter har vi nyligen fått den senaste versionen av TI FORTH, som vi snart kommer att tillhandahålla jämsides med föreningens FORTH. Björn Gustavsson berättar om det på annan plats i tidningen.

Slutligen: Ett par styrelsemedlemmar har avgått. Hans Attersjö har fått jobb i Belgien. Ny programförmedlare är Björn Mårtensson. Barbro Nygren har dragit sig tillbaka av personliga skäl. Jag tackar de båda för deras insatser och hälsar Björn välkommen. Vi får väl se om inte Hans hör av sig. 99-föreningen i Belgien är mycket aktiv.

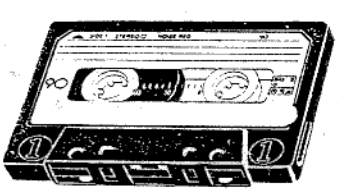
Jag hoppa vi ses den 15 september!

Lennart Lindberg

Lennart Lindberg, ordförande



Vi har fått en ny Programbanksförmedlare. Vår nye man heter Björn Mårtensson. Vår förre förmedlare, Hans Attersjö har påbörjat ett nytt arbete i Belgien. Björn Mårtensson går just nu igenom alla program, reviderar och uppdaterar banken. Vi kommer nu att kunna få ordentliga kommentarer och förklaringar av varje program. Vilket slag av utbyggnad du behöver, Extended Basic eller Mini-Memory och liknande. Även en beskrivning av hur man använder programmen, tangenter, joystick, matematiska uttryck etc. Dessa kommentarer kommer att publiceras i tidningen vartefter det är genomgånget. Alla nödvändiga förklaringar kommer att följa med program som skickas ut till medlemmar från programbanken.



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

Eftersom Utmaningen med inriktning på TI 58/59 har förekommit länge i tidningen Programbiten, och åtskilliga av de problem som har lösts där kan lösas bättre med 99:an, finns det naturligtvis ingen anledning att glömma bort de tidigare problemen, vilket Björn Gustavsson påpekade. Därför numrerar jag nu om problemen som gavs förra gången.

I stället för ett och två gäller dessa nummer:

46. Ett generellt program som kan köra vilket annat program som helst.

47. Sortering.

Denna numrering ansluter till den som finns i Programbiten 83-4, och ska alltså ses som en fortsättning på densamma. Vissa utmaningar som publicerats tidigare är kanske för enkla eller ointressanta att skriva på dator, medan andra passar mycket bra.
I Programbiten 83-2 finns ett register över de första 37 problemen, och jag tänker inte låta det gå i repris här. Tidningarna finns att köpa hos föreningen. Eftersom problem 45 är direkt avsett för en dator tar jag med det problemet i slutet av den här artikeln.
Kom gärna med lösningar på dessa tidigare utmaningar.

Nu till förra numrets problem:

46. GENERELL PROGRAMSTARTARE

Det har inte kommit någon lösning på detta problem, men det beror kanske på att det redan har behandlats i 99:er Magazine. Eftersom det var i de numera svåråtkomliga första utgåvorna av den tidningen tycker jag inte att det är något hinder för att försöka lösa problemet, även om det är frågan om att uppfinna hjulet en gång till. I novembernumret 1983 av 99:er finns en av lösningarna publicerad på nytt, som svar på en brevfråga. Det är där frågan om en "Poor Man's Program Loader", eftersom den inte kräver RAM-expansionen. Användarvänligheten blir lidande, men det fungerar.

47. SORTERING

Även problem 13 behandlar sortering, med det är så specialiserat för TI 59, att jag behåller det här problemet som ett eget.
Det har inte kommit lösningar från många personer, men Lennart Thelander i Lund presterade å andra sidan nio program med olika sorteringsmetoder. Vissa av dessa implementerar samma metod på olika sätt, men han har ändå lyckats hitta sex helt olika metoder. Alla Lennarts program är skrivna i Pascal. Det är antagligen inte så många 99-ägare som har utrustning för att kunna använda UCSD-systemet, men jag visar ändå Lennarts snabbaste program eftersom det kan vara intressant för dem som som bara skriver program i BASIC att se ett Pascalprogram någon gång.
Programmet som följer här är inspirerat av det BASIC-program som finns på sidan 20 i nittinian 83-4/5. Eftersom även variabelnamnen i huvudsak är identiska bör det inte vara svårt att förstå hur Pascalprogrammet fungerar. Orsaken till likheten mellan programmen är helt enkelt att den modifierade Quicksortmetoden är en av de effektivaste som finns. Problemet är snarast att skriva effektiva program som utnyttjar metoden. Lennarts program sorterar i de flesta fall 1000 flyttal på mindre än 30 sekunder. Värsta tänkbara fall, vilka inträffar mycket sällan i praktiken, förlänger tiden till ungefär två minuter.

Underprogrammet, eller proceduren som det heter i Pascal, ser ut så här:

```
PROCEDURE quicksort(VAR a:vaktor; n:INTEGER);
(* Vektorn A ska vara av typen datatype *)

CONST stackdepth=10;
      m=9;

VAR l,r,i,j   : INTEGER;
    swap,key : datatype;
    done      : BOOLEAN;
    level,ssfl,ssfr,lsfl,lsfr  : INTEGER:
    leftstack,rightstack  : ARRAY[0..stackdepth] OF
                                                INTEGER;
(*$R- *)

BEGIN
  level  :=0;
  l := 1;
  r := n;
  done := n<=m;

  WHILE NOT done DO
  BEGIN
    swap :=a[(l+r) DIV 2];
    a[(l+r) DIV 2] := a[l+1];
    a[l+1] := swap;

    IF a[l+1]>a[r] THEN
    BEGIN
      swap := r[l+1];
      a[l+1] := a[r];
      a[r] := swap;
    END;

    IF a[l]>a[r] THEN
    BEGIN
      swap := a[l];
      a[l] := a[r];
      a[r] := swap;
    END;
```

```
    IF a[l+1]>a[l] THEN
    BEGIN
      swap := a[l+1];
      a[l+1] := a[l];
      a[l] := swap;
    END;

    i := l+1;
    j := r;
    key := a[l];

    WHILE i<=j DO
    BEGIN
      REPEAT
        i := i+1;
      UNTIL a[i]>=key;

      REPEAT
        j := j-1;
      UNTIL a[j]<=key;

      IF j>i THEN
      BEGIN
        swap := a[i];
        a[i] := [j];
        a[j] := swap;
      END;
    END;
    swap := a[l];
    a[l] := a[j];
    a[j] := swap;

    IF ((j-l)<=m) AND ((r-i+1)<=m) THEN
    BEGIN
      IF level=0 THEN
        done := true;
      ELSE
      BEGIN
        l := leftstask[level];
        r := rightstack[level];
        level := level-1;
      END
    END
    ELSE
    BEGIN
      IF (j-l))=(r-i+1) THEN
      BEGIN
        lsfl := l;
        lsfr := j-1;
        ssfl := i;
        ssfr :=r;
      END
      ELSE
      BEGIN
        lsfl := i;
        lsfr := r;
        ssfl := l;
        ssfr := j-1;
      END;
      IF ((j-l)<=m) OR ((r-i+1)<=m) THEN
      BEGIN
        l := lsfl;
        r := lsfr;
      END
      ELSE
      BEGIN
        level := level+1;
        leftstack[level] := lsfl;
        rightstack[level] := lsfr;
        l := ssfl;
        r := ssfr;
      END;
    END;
  END;

  FOR i := n-1 DOWNTO 1 DO
  BEGIN
    IF a[i]>a[i+1] THEN
    BEGIN
      key := a[i];
      j := j+1;
      REPEAT
        a[j-1] := a[j];
        j := j+1;
      UNTIL (j>n) OR (a[j]=key);
      a[j-1] := key;
    END;
  END;
END;
```

Kommentaren (*$R- *) stänger av den kontroll av index i listor som normalt sker. Detta gör att programmet exekveras snabbare, men det kan inte rekommenderas innan man vet säkert att det fungerar!

Nu kunde det vara intressant att se hur fort föreningens Forth klarar av att sortera! Har du ett sorteringsprogram i Forth, så skicka in det! (fort) Själv håller jag på med ett assemblerprogram, avsett att anropas från Pascalprogram, men det är inte färdigt än. Dessutom finns det ibland behov av att sortera så mycket på en gång att minnet inte räcker till. Då måste diskenheterna utnyttjas, och de metoder som kommer till användning skiljer sig åtskilligt från den här ovan. Intressant är att en av de metoder som Lennart använder sig av ganska lätt kan modifieras för sortering av mycket stora datamängder. Jag kommer att återkomma till detta i en Utmaning längre fram.

HJÄLPPROGRAM FÖR TI 59 (45)

Detta problem fanns formulerat i Programbiten 83-3, men då det är avsett för en dator kommer det i repris här. (Jag har kopierat Robert Prins text utan ändringar)

45a. Det här är ett program som ska lösas i BASIC: Skriv ett program som läser in 59:ans tangent koder och ger en listning liknande den på PC-100.

45b. Lägg till ett program som gör om alla hopp till labels till absoluthopp och sedan listar det ändrade programmet.

45c. (Troligen väldigt svårt.) Skriv ett program i BASIC eller i assembler om du är ett riktigt geni, som simulerar en utökad TI-59, dvs ett program som uppför sig exakt som en TI-59 med 1000 register och 10000 (tio tusen) steg. Det betyder att utökade koder kan vara t.ex. RCE (ReCall Extended), där de två följande stegen ger minnesadressen eller GTE (GOTO Extended). Du kanske undrar varför jag vill se ett sådant program ? Tja, jag tycker det är mycket lättare att göra något i "59-language" än BASIC.

Så långt Robert Prins. Vilket programspråk som är lättast att använda finns det säkert olika åsikter om, men det ska vi inte diskutera här.

Beträffande problemen vill jag lägga till följande:

45a. Kanske någon kan komma på ett utseende på listan som är klart bättre än den PC-100 producerar?

45b. Det vore praktiskt med ett program som gör motsatsen också, dvs från absoluthopp till labels. Om man vill göra stora ändringar i ett absolutadresserat program kan det vara en god hjälp att få hoppen ändrade.

45c. Haka inte upp er på orden "exakt som en TI-59". Det är synnerligen besvärligt att få med alla egenheter som TI-59 har (jag tänker då på Fast Mode, Graphics Mode etc.). Dessutom kan 10000 steg och 1000 minnen vara i mesta laget för ett BASIC-program.

45d. Skriv ett program som bär sig åt praktiskt taget som en TI-59 med 1000 programsteg och 100 minnen samtidigt. Ordalydelsen "praktiskt taget" ska tolkas så här:
Alla program som inte utnyttjar speciella knep (hexhoder etc.) ska kunna köras med ungefär samma resultat.
Att jag skriver "ungefär samma" beror på att jag tycker att 99:ans beräkningsnoggrannhet kan behållas (13-14 siffror, talområde från 1 E-128 till 1 E+127). Dessutom kommer det ju inte att finnas någon överensstämmelse mellan minnen och programsteg, eftersom de kommer att vara tillgängliga samtidigt.
OP 16 ska ge resultatet 999.99 och OP 17 skall inte göra någonting alls.
Dessutom finns det andra programspråk än BASIC som kan komma ifråga...

Därför vill jag presentera detta problem:

STAVNINGSKOLLARE (48)

Ordbehandlingsprogrammet TI Writer (med vilken den här artikeln är (in-)skriven), har tre olika huvudfunktioner:

1. Textredigering
2. Formatering (utskrift alltså)
3. Egna rutiner

Dessa "egna rutiner" gör att det går att starta vilket assemblerprogram som helst med TI Writer-modulen (Forth t.ex.).

Problem 48:

Skriv ett program som kontrollerar stavningen i en textfil som skapats med TI Writer.
Det här är förvisso inget enkelt problem, men värre sådana har lösts med TI 59! Själva stavningskontrollen är inte så svår, men lagringen av den nödvändiga ordlistan kräver eftertanke, eftersom det blir en väldig massa bokstäver. Idén kommer från Bo Nordlin.

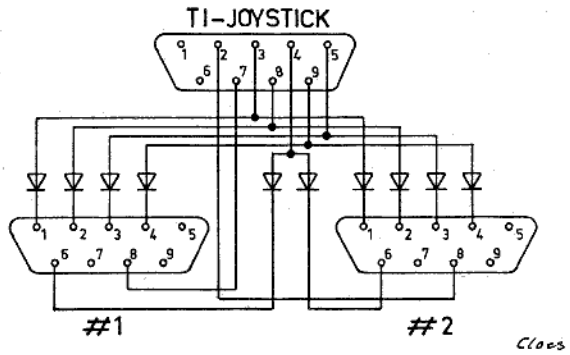
Slut för den här gången. Kom gärna med lösningar eller nya problem!

På återhörande
Anders Persson

TEKNISKA Avdelningen

KOMMENTARER TILL TEKNISKA AVDELNINGEN OCH ALLT OM DISK-DRIVAR I 99:AN 83-4/5

Den joystickadapter som Lars Ekeroth beskriver i Tekniska avdelningen i nummer 83-4/5 fungerar inte som avsett om man försöker att samtidigt använda två stycken joysticks. Om adaptern inte kompletteras med dioder enligt figur påverkar den ena joysticken den andra. Upp på en och höger på den andra ger snett upp åt höger. Upp på en och ner på den andra ger ingenting alls som resultat.



Hans Attersjös artikel om disk-drivar innehöll det mesta av intresse inom det området. Jag vill dock tillägga att det finns ytterligare en anslutningsmöjlighet om man vill ha två stycken drivenheter i expansionsboxen.

Själv har jag utnyttjat Texas styrenhet tillsammans med drivenheter från TEAC, typ FD-55B. Det är en dubbelsidig drive i halvhöjdsformat, varför det går att montera in två stycken i expansionsboxen vid sidan om varandra. Strömförsörjningen är så låg att det går bra att utnyttja expansionsboxens transformator. Däremot får man byta ut och/eller lägga till spänningsstabilisering till drivenheterna, eftersom de inbyggda stabiliseringskretsarna är på en ampere och två enheter förbrukar mer än så tillsammans. Någon kanske tycker att man bara behöver en enhet i taget igång, men det är faktiskt önskvärt att drivspindlarna i drivenheterna snurrar parallellt. Då behöver datorn mämligen inte vänta på att den andra enheten ska starta när man omväxlande läser och skriver på två olika disketter.

Efter denna lilla orientering om hur mitt system ser ut ska jag nu beskriva det som är annorlunda med min anslutning av signalerna till drivenheterna.
Hans A. skriver att man ska låta alla drivenheterna vara inställda på DS1 (eller DS0 om det finns) samt HS, tack vare det lilla kretskortet som man kopplar ihop anslutningskablarna med. Detta är helt riktigt, men om man ska ha båda enheterna i expansionsboxen finns det ett mer estetiskt tilltalande sätt.

TEKNISKA Avdelningen

TEKNIKTIPS

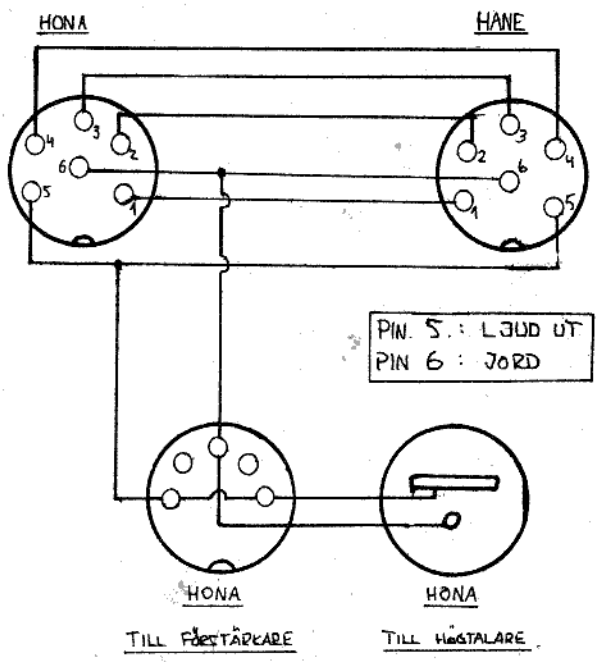
LJUDFÖRBÄTTRING FÖR DIN 99'A.

AV: LARS EKEROTH

Om du har störningar på ljudet när du använder din dator är denna kopplingen lösningen till problemet. Med denna enkla koppling kan man få ut ljudet i en förstärkare eller en högtalare utan minsta störning. Ljudet blir så bra att man tillochmed hör signalerna från datorn när man listar program.

Man behöver en DIN-kontakt hane 6-pol, en 6-pol hona, en 5-pol hona och en högtalarkontakt hona. (+ kopplingstråd)

Löd ihop komponenterna enligt schemat och koppla in detta mellan datorn och PAL-modulatorn. Sen kan du koppla in högtalaren eller förstärkaren och lyssna.



Tillsammans med diskkontrollerenheten kommer det två flatkablar. Den ena är avsedd för den speciella interna drive som TI säljer, och skall anslutas till kontakten på ena sidan av diskkontrollerkortet. Den kabeln använder man till att ansluta den drivenhet som ska vara nummer ett, lämpligtvis. Det spelar egentligen ingen roll vilket nummer den ska ha, men det är vettigast att ha nummer ett till vänster i boxen. Att den enhet som är ansluten till internuttaget på kontrollkortet ska sitta till vänster beror helt enkelt på att kabeldragningen blir enklare då. Till den andra driven använder man den andra kabeln, logiskt nog. Eftersom jag i det här fallet har en drive som elektriskt sett är en yttre enhet (även om den befinner sig i expansionsboxen, den också) kunde jag använda denna extra kabel direkt, om än med vissa modifikationer. Det finns nämligen en styrguide i kontakten som passar till adapterkortet och inte till en diskdrive, men den styrklacken kan man avlägsna med våld i lämplig dosering. Sedan ansluter man bara enhet nummer två helt parallellt med nummer ett. Här finns dock inte det lilla kretskortet med, som annars växlar identifikationssignalerna så att olika drivenheter får olika adresser. Kontrollkortet måste därför få information om vilken drive som är vilken genom att omkopplarna på själva diskdriven ställs in på respektive nummer, DS0 och DS1 i mitt fall. Det motsvarar alltså DSK1 och DSK2.

När man kopplar så här låter man resistorpacken sitta kvar i båda enheterna. Det går inte att koppla till en tredje drive på det här sättet med de befintliga kablarna, men om man gör en grenkabel kan även den tredje enheten kopplas in på samma sätt. Den ska då ställas in som DS2, och blir DSK3 i systemet. Lämplig låda och yttre strömförsörjning måste naturligtvis ordnas.

Anders Persson

SERVICE, DISKDRIVAR, INTERFACE, ...

LARS MAGNUSSON

tidigare servicetekniker på Texas Instruments, håller på att jaga utrustning till 99/4A Han har lovat informera om verksamheten på separat medskick i detta nummer av PB.

Tankar om FORTH

av Björn Gustavsson

Nyligen har föreningen fått tag på den fullständiga versionen av TI FORTH och fått rätten att distribuera även denna. Här kommer en liten historik om hur föreningens FORTH har tillkommit, vilket utmynnar i en jämförelse mellan de båda.

I slutet av 1982 var jag irriterad över att 99:an inte hade något vettigt högnivåspråk. Pascal anser jag vara för långsamt på 99:an. Vad jag längtade efter var FORTH. Jag hade hört rykten om att TI skulle komma med FORTH och väntade ivrigt. Tiden gick utan några nyheter om TI FORTH. Då beslöt jag mig för att själv implementera FORTH. Jag beställde en listning av FORTH för 9900 från FORTH Interest Group.

Våren -83 var hela assemblerlistan inmatad. Det som återstod var felsökning och tillägg av rutiner för kommunikation med tangentbordet, skärm och flexskivan (allt detta är helt beroende på vilken dator man har och fanns ju inte med i FIG:s lista). Men jag kom aldrig längre. I samband med mikrodatormässan i Sollentuna fick jag tag på TI:s prototypversion av FORTH. Det blev inget mer gjort med min egen implementation.

Prototypen var en helt "naken" FORTH. Den innehöll bara grundordlistan på cirka 7K, och hade inte ens en editor. Det första jag gjorde var att implementera FIG:s radorienterade editor. Den fick matas in två gånger: först en gång från tangentbordet med omedelbar kompilering och sedan en gång till med hjälp av sig själv in på skivan.

När jag hade en fungerade editor kunde jag sätta igång på allvar. Jag implementerade en disassemblator, skrivarrutiner, dekompilator, en skärmorienterad editor, teckendefinitionsrutiner, små bokstäver och svenska tecken, assemblator och en del annat.

I slutet av 1983, någon gång efter nedläggningen av hemdatorn, fick föreningen TI:s tillstånd att sprida FORTH till medlemarna.

FORTH kunde bara köras med Editor/assembler och skivminne, så bara ett fåtal av medlemmarna skulle kunna köra det. Jag tänkte modifiera FORTH så att det åtminstone kunde köras med Mini Memory. Att få det att fungera med Extended BASIC verkade omöjligt utan tillgång till källkoden.

Jag definierade ordet SAVE-SYSTEM som sparade hela minnet inklusive ordlistan i programfiler med namnen UTIL1, UTIL2, UTIL3 och UTIL4. Programfiler med det namnet kan köras automatiskt med val 5 i Editor/-assembler. Mini Memory saknar denna möjlighet att köra assemblerprogram i det formatet, men det var relativt enkelt att skriva ett laddningsprogram som laddade in programfiler.

Ett större problem med Mini Memory-varianten var att få anrop av rutiner i GPL att fungera. Utan GPL-anrop skulle det inte gå att göra en kassettbaserad FORTH, eftersom drivrutinen för kassettbandspelare är skriven i GPL. Har man Editor/assembler är det inga problem eftersom modulen innehåller möjligheter för GPL-anrop. Men Mini Memory fungerade annorlunda och det verkade som jag var tvungen att göra separata versioner för de båda modulerna. Men till slut fick jag det att fungera helt oberoende av vilken modul man hade, och t o m utan modul!

Sedan var det relativt enkelt att få det att fungera med kassettbandspelare.

Problemet Extended BASIC återstod. Man måste ha ett laddningsprogram i assembler som laddar programfiler. Det var enkelt i Mini Memory, eftersom det fanns 4K extra att lägga det i. Men i Extended BASIC skulle laddningsprogrammet skrivas över när FORTH lästes in. Men även det gick att lösa. En del av programmet fick läggas i de 256 bytes RAM som alltid finns i datorn. Denna del skrivs då nte över och kan sköta inladdningen av den sista programfilen och starta FORTH.

Det sista problemet var att få detta att fungera med kassettbandspelare. Extended BASIC har ingen möjlighet att direkt läsa in assemblerprogram från kassettbandspelare. Enda möjligheten att ladda laddningsprogrammet var byte för byte med CALL LOAD. Första gången matade jag in CALL LOAD-satserna manuellt (mycket roligt!). Andra gången när jag hade hittat några mindre fel hade jag fått tag på Börje Hälls utmärkta program som gör om ett assemblerprogram till ett BASIC-program med CALL LOAD-satser (enligt red kommer det i det här numret av Programbiten).

När detta sista problem var löst informerade vi om FORTH i Nittinian. Några veckor efter att numret kommit ut kom den fullständiga versionen av TI FORTH från Maurice Swinnen, USA. Den innefattade två editorer och en massa annat. Det mest intressanta är en editor med 64 tecken per rad på skärmen!
Bokstäverna blir då ganska smala, men går att läsa.

Vad finns det nu för skillnader mellan dessa två varianter av FORTH?

Den grundläggande ordlistan är identisk. Jag och TI har sedan byggt på denna med editor och annat. Eftersom vi har arbetat oberoende av varandra skiljer påbyggnaderna en hel del.
Föreningens FORTH har möjligheten att köras med flera olika moduler och med kassettbandspelare, medan TI FORTH endast kan köras med Editor/assembler och skivminne. En annan finess i föreningens FORTH är riktiga små bokstäver och svenska tecken.
(Se mera i artikeln om teckendefinitioner i FORTH i detta nummer.) Dessutom kan man ganska enkelt använda 64-teckens editorn och annat från TI FORTH i föreningens FORTH, eftersom källkoden för allt detta finns på skivan med TI FORTH. Denna källkod finns också listad i manualen.
Senare tänkte vi visa i tidningen hur man utnyttjar godbitar ur TI FORTH i föreningens FORTH.

Enligt min mening bör man köpa föreningens FORTH eller båda två. Föreningen kommer att sälja skivan med TI FORTH för 60 SEK. Manualen kommer att säljas separat till självkostnadspris. Priset meddelas senare när vi vet vad det kostar att trycka den. Meddela föreningen FORTHast möjligt om du är intresserad - som bekant sjunker priset per exemplar ju fler man trycker. Även den som har föreningens FORTH kan ha nytta av manualen.

Tyvärr måste jag meddela att jag inte hinner syssla lika mycket med 99:an som tidigare. Den första mars började jag som konsult i Compis-projekt på Teli i Nynäshamn. När man sitter vid en dator hela dagarna är man inte så värst pigg att sätta sig vid en dator efter jobbet. Jag skall ändå försöka uppfylla en del av de mål som jag har satt upp:
att utöka dokumentationen till föreningens FORTH och att skriva om FORTH i tidningen. Jag tänkte också sammanställa all källkod för t ex editorn på en skiva som kommer att säljas för självkostnadspris (troligen 60 SEK).

Jag hoppas att alla som har tips eller frågor om FORTH hör av sig till föreningen. I nästa nummer kommer vi att starta en frågespalt om FORTH.

VDP REGISTER FRÅN BASIC.

av Tony Rogvall

Om man antingen har Mini-memory eller Editor/Assembler så har man tillgång till ett kommando som heter POKEV, som är ett underprogram och anropas med CALL POKEV(adress,värde). Med hjälp av detta kommando kan man nå VDP ram, men också VDP registerna. Orsaken är att VDP är minneskartat och nås via tre adresser i CPU ram, nämligen VDPWA ()8C02 VDP Write Address), VDPWD ()8C00 VDP Write data) och VDPRD ()8800 VDP read data). Dessa adresser skall du vara försiktig med, alltså varken peeka eller poka på dom. Genom att sätta den mest sianifikanta biten när man skriver till VDPWA markerar man att det är ett register man vill skriva i.
När man från assembler vill skriva till ett VDP register kan man antingen använda sig av VWTR (VDP write to register) eller VSBW (VDP single byte write) rutinerna. VWTR fungerar så att man i CPU register R0 skriver värdet i den minst sianifikanta, och registret i den mest signifikanta, byten. Om man i stället vill använda sig av VSBW för att sätt registret gör man på liknande sätt, fast man får addera 128 ()80) till den mest signifikanta byten. I CPU register R1 (där värdet normalt skrivs) skriver man ingenting.

Om man nu vill ändra på ett register från BASIC måste man räkna ut vilken adress som motsvarar det registret man skall sätta med det värde man vill att det skall innehålla. Lättast är det att först räkna ut det i hexadecimal. 1.först skall MSB vara satt, det ger)8000 2. register X skall läggas till)8X00 3. värdet VV skall adderas till detta)8XVV.

Exempel (initiera text-mode)

VDP registret som skall sättas är nummer 1.Värdet det skall innehålla är 11110000 binärt, 240 decimalt eller F0 hexadecimalt. Adressen blir således)81F0 eller -32272 decimalt. När du skriver från BASIC blir det alltså CALL POKEV(-32272,0) och vips du är i text-mode.
Om du vill utnyttja text mode för att skriva i måste du också ändra på en CPU ram adress som innehåller en kopia av VDP register 1, och det är)83D4 eller -31788. Annars kommer detta värde att läggas tillbaka i VDP 1 efter en tangent ned tryckning.
Det finns många saker du kan göra med VDP registerna t.ex slå av skärmen, initiera bit map mode, initiera multicolor mode, ändra på skärm minnets adresser osv. Program exemplet nedan initierar text-mode, tar emot en sträng och skriver sedan ut den.
LYCKA TILL!

```
100 CALL CLEAR
110 REM RENSA NEDRE DELEN AV TEXT SKÄRMEN
120 FOR I=767 TO 959
130 CALL POKEV(I,128)
140 NEXT I
150 REM SÄTT TEXT MODE
160 CALL POKEV(-32272,0)
170 REM SÄTT BAKGRUNDS FÄRG
180 CALL POKEV(-30735,0)
190 REM LÄGG VÄRDET I VDP 1 KOPIAN
200 CALL LOAD(-30735,240)
210 I=206
220 CALL SOUND(200,1400,0)
230 CALL KEY(0,K,S)
240 IF S=0 THEN 230
250 IF K=13 THEN 300
260 I=I+1
270 A$=A$&CHR$(K)
280 CALL POKEV(I,K+96)
290 GOTO 230
300 FOR I=1 TO LEN(A$)
310 CALL POKEV(399+I,ASC(SEG$(A$,I,1))+96)
320 NEXT I
330 REM ÅTERSTÄLL REGISTER
340 CALL POKEV(-32288,0)
350 CALL LOAD(-31788,244)
360 END
```

PROGRAM LADDAREN

Program: Börje Häll
Beskrivning: Björn Gustavsson

Programmet som beskrivs här gör om ett assemblerat assemblerprogram till ett BASIC-program som laddar detta med hjälp av CALL LOAD i följande form:

CALL LOAD(adr,data,data,data...)

Den största nyttan av detta program har man när man vill att ett assemblerprogram skall kunna laddas och köras med Extended BASIC, 32k extra minne och kassettbandspelare. Utån skivminne finns nämligen ingen annan möjlighet att ladda ett assemblerprogram från Extended BASIC. Observera att den som skall köra omvandlingsprogrammet måste ha skivminne.

Jag har haft stor nytta av programmet när jag skulle göra en kassettversion av FORTH för Extended BASIC. Att göra en manuell omvandling av laddningsprogrammet tar flera timmar (jag har provat - en gång).

Köranvisning:

1. Programmet som skall omvandlas måste vara assemblerat. Programmet måste vara absolut-relokerat, dvs ett AORG-direktiv måste ingå i källkoden.

2. Kör programmet LOADASMBAS i Extended BASIC. Programmet frågar efter namnet på objektfilen. Ange namnet på filen där den assemblerade koden finns.

3. Programmet läser filen och skriver in ett BASIC-program i filen DSK1.MERGE. Programmet skriver också ut vilka "referenser" som finns. Detta är de namn som programmet kan anropas med (CALL LINK).

4. Skriv NEW. Skriv MERGE DSK1.MERGE. Nu finns programmet som laddar assemblerprogrammet i minnet. Det kan sparas på vanligt vis, antingen kassett eller flexskiva.

Exempel:

Programmet på sidan 344-346 i assemblermanualen har assemblerats och objektkoden ser ut som program 3. (före assemblering har ett AORG-direktiv satts in och REF till VMBW, VMBR m fl har bytt mot EQU enligt sidan 416.) Efter att ha kört LOADASMBAS har vi fått program 4 som laddar assemblerprogrammet. Program 4 kan köras med endast Extended BASIC och 32k extra minne.

Program 1.
Loadasmbas i originalversion. Remsatser läggs in i Basic-programmet som skapas. Lämpligt att använda under uttestning.

```
 10 ! ********************************
    ********************
 20 ! *****  LOADASMBAS
                  *****
 30 ! *****  Program f!r att skriva e
    tt program,   *****
 40 ! *****  som laddar ett assembler
    program.      *****
 50 ! *****  B!rje H@ll
                  *****
 60 ! *****  Vasav(gen 101
                  *****
 70 ! *****  175 32 J(rf(lla
                  *****
 80 ! ********************************
    ********************
 90 CALL CLEAR
100 ! ********************************
    ********************
110 ! * Definiera svenska tecken
120 ! *
130 CALL CHAR(91,"00280038447C4444002
    8007C4444447C003828384447C4444")
140 CALL CHAR(123,"000028003847C4400
    002800?C44447C000038283847C44")
150 ! ********************************
    ********************
160 ! * Definiera funktioner
170 ! *
180 DEF HB(X)=INT(X/256)
190 DEF LB(X)=X-256*HB(X)
200 DEF ADR$(X)=CHR$(200)&CHR$(LEN(ST
    R$(X)))&STR$(X)
210 DEF HB$(X)=CHR$(179)&CHR$(200)&CH
    R$(LEN(STR$(X)))&STR$(X)&CHR$(179
    )
220 DEF LB$(X)=CHR$(200)&CHR$(LEN(STR
    $(X)))&STR$(X)
230 DEF REF$(X$)=CHR$(179)&CHR$(200)&
    CHR$(2)&STR$(ASC(X$))
240 DEF AST$(X)=CHR$(HB(X))&CHR$(LB(X
    ))&CHR$(131)&CHR$(200)&CHR$(54-LE
    N(STR$(X)))&" "&RPT$("*",53-LEN(S
    TR$(X)))&CHR$(0)
250 DEF AST1$=CHR$(A)&CHR$(B)&CHR$(13
    1)&CHR$(200)&CHR$(8)&" *****  "
260 DEF AST2$(X$)=AST1$&SEG$(X$,1,37)
    &RPT$(" ",54-LEN(STR$(LINE))-LEN(
    AST1$)-LEN(SEG$(X$,1,37)))&"*****
    "
270 DEF AST3$(LINE)=CHR$(HB(LINE))&CH
    R$(LB(LINE))&CHR$(131)&CHR$(200)
280 ! ********************************
    ********************
290 ! * Initialisera konstanter och v
    ariabler
300 ! *
310 LOAD$=CHR$(157)&CHR$(200)&CHR$(4)
    &"LOAD"&CHR$(183)
320 HEX$="123456789ABCDEF"
330 TAG$="6789B"
340 LINE=100
350 REFTAB=16384
360 ! ********************************
    ********************
370 ! * Fr)ga efter filnamn f!r objek
    tkoden
380 ! *
390 INPUT "Filnamn f!r objektkod ?
    ":FILE$
400 IF POS(FILE$,"DSK",1)THEN 420
410 FILE$="DSK1."&FILE$
420 CALL CLEAR
430 ON ERROR 1790
440 ! ********************************
    ********************
450 ! * \ppna filer
460 ! *
470 OPEN #1:FILE$,DISPLAY ,FIXED 80,I
    NPUT
480 OPEN #2:"DSK1.MERGE",DISPLAY ,OUT
    PUT,VARIABLE 163
490 PRINT "L(ser objektkod "&SEG$(FIL
    E$,POS(FILE$,".",1)+1,14): ;
500 PRINT "och skriver BASIC-program"
    ; ;
510 PRINT "i fil MERGE."
520 ! ********************************
    ********************
530 ! * Skriv kommentar i fil MERGE
540 ! *
```

```
550 PRINT #2:AST$(LINE)
560 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
570 PRINT #2:AST2$("MERGE")
580 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
590 PRINT #2:AST2$("Programmet {r skr
    ivet av programmet")
600 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
610 PRINT #2:AST2$("LOADASMBAS.")
620 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
630 PRINT #2:AST2$("LOADASMBAS 1(ser
    en assembler OBJEKT-")
640 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
650 PRINT #2:AST2$("kod och g!r om ob
    jektkodens data till")
660 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
670 PRINT #2:AST2$("CALL LOAD-satser.
    ")
680 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
690 PRINT #2:AST2$("LOADASMBAS {r skr
    ivet av")
700 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
710 PRINT #2:AST2$("B!rje H(11")
720 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
730 PRINT #2:AST2$("Vasav{gen 101")
740 LINE=LINE+10 :: A=HB(LINE):: B=LB
    (LINE)
750 PRINT #2:AST2$("175 32  J{rf{lla"
    )
760 LINE=LINE+10
770 PRINT #2:AST$(LINE)
780 LINE=LINE+10
790 ! * Skriv "radnummer" CALL INIT i
     fil MERGE
800 ! *
810 LINE$=CHR$(HB(LINE))&CHR$(LB(LINE
    ))&CHR$(157)&CHR$(200)
820 LINE$=LINE$&CHR$(4)&"INIT"&CHR$(0
    )
830 PRINT #2:LINE$
840 LINE=LINE+10
850 ! ********************************
    ********************
860 ! * Skriv kommenter i fil MERGE
870 ! *
880 PRINT #2:AST$(LINE)
890 LINE=LINE+10
900 PRINT #2:AST3$(LINE)&CHR$(28)&" *
     Ladda assemblerprogrammet"&CHR$(
    0)
910 LINE=LINE+10
920 PRINT #2:AST3$(LINE)&CHR$(2)&" *"
    &CHR$(0)
930 LINE=LINE+10
940 GOSUB 1900
950 A$=SEG$(A$,14,80)
960 GOSUB 2030
970 LINE$=ADR$(ADR)
980 ! ********************************
    ********************
990 ! * Omvandla objektkodens hexadec
    imala koder
1000 ! * till decimala koder
1010 ! *
1020 A$=SEG$(A$,6,80)
1030 FOR I=1 TO 80 STEP 5
1040 ! ********************************
     ********************
1050 ! * Kontrollera om objektkoden ka
     n laddas fr}n X-BASIC
1060 ! *
1070 TAG$=SEG$(A$,I,1)
1080 IF POS(TAGS$,TAG$,1)=0 THEN 1830
1090 ! ********************************
     ********************
1100 ! * Tag 7 eller 8 f!reg}r checksu
     mman, som {r sist
1110 ! * i varje record i objektkoden
1120 ! *
1130 IF TAG$="7" OR TAG$="8" THEN 1170
1140 GOSUB 1730
1150 LINE$=LINE$&HB$(HIGH)&LB$(LOW)
1160 NEXT I
1170 GOSUB 1950
1180 GOSUB 1900
1190 ! ********************************
     ********************
1200 ! * Tag 6 f!reg}r referenser
1210 ! *
1220 IF SEG$(A$,1,1)="6" THEN 1290
1230 GOSUB 2030
1240 LINE$=ADR$(ADR)
1250 GOTO 1020
1260 ! ********************************
     ********************
1270 ! * Ta reda p} referenser
1280 ! *
1290 PRINT : :"REFERENSER": :
1300 ! ********************************
     ********************
1310 ! * Skriv kommentar i fil MERGE
1320 ! *
1330 PRINT #2:AST$(LINE)
1340 LINE=LINE+10
1350 PRINT #2:AST3$(LINE)&CHR$(29)&" *
     Uppdatera REF/DEF-tabellen"&CHR$
    (0)
1360 LINE=LINE+10
1370 PRINT #2:AST3$(LINE)&CHR$(13)&" *
     Referenser"&CHR$(0)
1380 LINE=LINE+10
1390 FOR I=1 TO 80 STEP 11
1400 TAG$=SEG$(A$,I,1)
1410 ! ********************************
     ********************
1420 ! * : anger slut p} objektkoden
1430 ! *
1440 IF TAG$=":" THEN 2180
1450 ! ********************************
     ********************
1460 ! * Tag 7 eller 8 f!reg}r checksu
     mman, som {r sist
1470 ! * i varje record i objektkoden
1480 ! *
1490 IF TAG$="7" OR TAG$="8" THEN 1640
1500 GOSUB 1730
1510 PRINT SEG$(A$,I+5,6): :
1520 ! ********************************
     ********************
1530 ! * Skriv kommentar i fil MERGE
1540 ! *
1550 PRINT #2:AST3$(LINE)&CHR$(9)&" *
     "&SEG$(A$,I+5,6)&CHR$(0)
1560 LINE=LINE+10
1570 GOSUB 2110
1580 LINE$=LINE$&HB$(HIGH)&LB$(LOW)
1590 REFTAB=REFTAB-8
1600 NEXT I
1610 ! ********************************
     ********************
1620 ! * Skriv kommentar i fil MERGE
1630 ! *
1640 PRINT #2:AST3$(LINE)&CHR$(2)&" *"
     &CHR$(0)
1650 LINE=LINE+10
1660 LINE$=ADR$(REFTAB)&LINE$
1670 GOSUB 1950
1680 GOSUB 1900
1690 GOTO 1390
1700 ! ********************************
     ********************
1710 ! * Omvandla hexadecimal kod till
     decimal kod i high- och low byte
1720 ! *
1730 HIGH=16*(POS(HEX$,SEG$(A$,I+1,1),
     1))+POS(HEX$,SEG$(A$,I+2,1),1)
1740 LOW=16*(POS(HEX$,SEG$(A$,I+3,1),1
     ))+POS(HEX$,SEG$(A$,I+4,1),1)
1750 RETURN
1760 ! ********************************
     ********************
1770 ! * Felhantering
1780 ! *
1790 CALL ERR(ERRCODE,ERRTYPE,ERRSEV,L
     INE)
1800 PRINT : :
1810 IF ERRCODE=130 THEN PRINT "Felakt
     igt filnamn.": : :: RETURN 390
1820 PRINT "Fel";ERRCODE;"i rad";LINE:
      :: GOTO 1840
1830 PRINT "Objektkoden kan inte ladda
     s": :"fr}n Extended BASIC."
1840 CLOSE #1
1850 CLOSE #2:DELETE
1860 STOP
1870 ! ********************************
     ********************
1880 ! * L{s ett record fr}n objektkod
     en
1890 ! *
1900 INPUT #1:A$
1910 RETURN
1920 ! ********************************
     ********************
1930 ! * Skriv "radnummer" CALL LOAD(x
     ,y,z) i fil MERGE
1940 ! *
1950 LINE$=CHR$(HB(LINE))&CHR$(LB(LINE
     ))&LOAD$&LINE$&CHR$(182)&CHR$(0)
1960 PRINT #2:LINE$
1970 LINE=LINE+10
1980 LINE$=""
1990 RETURN
2000 ! ********************************
     ********************
2010 ! * Ta reda p} decimal laddningsa
     dress
2020 ! *
2030 ADR=4096*(POS(HEX$,SEG$(A$,2,1),1
     ))+256*(POS(HEX$,SEG$(A$,3,1),1))
2040 ADR=ADR+16*(POS(HEX$,SEG$(A$,4,1)
     ,1))+POS(HEX$,SEG$(A$,5,1),1)
2050 IF ADR=0 THEN 1830
2060 IF ADR>32767 THEN ADR=ADR-65536
2070 RETURN
```

```
2080 ! ********************************
     ********************
2090 ! * Omvandla REF till ASCII-kod o
     ch flytta till LINE$
2100 ! *
2110 FOR J=I+5 TO I+10
2120 LINE$=LINE$&REF$(SEG$(A$,J,1))
2130 NEXT J
2140 RETURN
2150 ! ********************************
     ********************
2160 ! * Avsluta
2170 ! *
2180 CLOSE #1
2190 ! ********************************
     ********************
2200 ! * Uppdatera pekaren till REF/DE
     F-tabellen
2210 ! * Skriv kommentar i fil MERGE
2220 ! *
2230 PRINT #2:AST$(LINE)
2240 LINE=LINE+10
2250 PRINT #2:AST3$(LINE)&CHR$(42)&" *
      Uppdatera pekaren till REF/DEF-t
     abellen"&CHR$(0)
2260 LINE=LINE+10
2270 PRINT #2:AST3$(LINE)&CHR$(2)&" *"
     &CHR$(0)
2280 LINE=LINE+10
2290 LINE$=ADR$(8196)&HB$(HB(REFTAB))&
     LB$(LB(REFTAB))
2300 GOSUB 1950
2310 ! ********************************
     ********************
2320 ! * Skriv "radnummer" END i fil M
     ERGE
2330 ! *
2340 PRINT #2:CHR$(HB(LINE))&CHR$(LB(L
     INE))&CHR$(139)&CHR$(0)
2350 ! ********************************
     ********************
2360 ! * Skriv filslut i fil MERGE
2370 ! *
2380 PRINT #2:CHR$(255)&CHR$(255)
2390 CLOSE #2
2400 END
```

Program 2.

Loadasmbas ändras så att Rem-satser inte läggs in. Lämpligt för den slutgiltiga versionen där man vill ha snabbast möjliga laddning.

```
  10 ! ********************************
     ********************
  20 ! *****  LOADASMBAS
                    *****
  30 ! *****  Program f|r att skriva e
     tt program,    *****
  40 ! *****  som laddar ett assembler
     program.       *****
  50 ! *****  Birje H{ll
                    *****
  60 ! *****  Vasav{gen 101
                    *****
  70 ! *****  175 32 J|rf{lla
                    *****
  80 ! ********************************
     ********************
  90 CALL CLEAR
 100 ! ********************************
     ********************
 110 ! * Definiera svenska tecken
 120 ! *
 130 CALL CHAR(91,"00280038447C4444002
     8007C4444447C0038283847C4444")
 140 CALL CHAR(123,"00002800384 7C4400
     0028007C44447C00003828384 47C44")
 150 ! ********************************
     ********************
 160 ! * Definiera funktioner
 170 ! *
 180 DEF HB(X)=INT(X/256)
 190 DEF LB(X)=X-256*HB(X)
 200 DEF ADR$(X)=CHR$(200)&CHR$(LEN(ST
     R$(X)))&STR$(X)
 210 DEF HB$(X)=CHR$(179)&CHR$(200)&CH
     R$(LEN(STR$(X)))&STR$(X)&CHR$(179
     )
 220 DEF LB$(X)=CHR$(200)&CHR$(LEN(STR
     $(X)))&STR$(X)
 230 DEF REF$(X$)=CHR$(179)&CHR$(200)&
     CHR$(2)&STR$(ASC(X$))
 240 DEF AST$(X)=CHR$(HB(X))&CHR$(LB(X
     ))&CHR$(131)&CHR$(200)&CHR$(54-LE
     N(STR$(X)))&" "&RPT$("*",53-LEN(S
     TR$(X)))&CHR$(0)
 250 DEF AST1$=CHR$(A)&CHR$(B)&CHR$(13
     1)&CHR$(200)&CHR$(8)&" *****   "
 260 DEF AST2$(X$)=AST1$&SEG$(X$,1,37)
     &RPT$(" ",54-LEN(STR$(LINE))-LEN(
     AST1$)-LEN(SEG$(X$,1,37)))&"*****
     "
 270 DEF AST3$(LINE)=CHR$(HB(LINE))&CH
     R$(LB(LINE))&CHR$(131)&CHR$(200)
 280 ! ********************************
     ********************
 290 ! * Initialisera konstanter och v
     ariabler
 300 ! *
 310 LOAD$=CHR$(157)&CHR$(200)&CHR$(4)
     &"LOAD"&CHR$(183)
 320 HEX$="123456789ABCDEF"
 330 TAG$$="6789B"
 340 LINE=100
 350 REFTAB=16384
 360 ! ********************************
     ********************
 370 ! * Fr}ga efter filnamn f|r objek
     tkoden
 380 ! *
 390 INPUT "Filnamn f|r objektkod ?
       ";FILE$
 400 IF POS(FILE$,"DSK",1)THEN 420
 410 FILE$="DSK1."&FILE$
 420 CALL CLEAR
 430 ON ERROR 1790
 440 ! ********************************
     ********************
 450 ! * \ppna filer
 460 ! *
 470 OPEN #1:FILE$,DISPLAY ,FIXED 80,I
     NPUT
 480 OPEN #2:"DSK1.MERGE",DISPLAY ,OUT
     PUT,VARIABLE 163
 490 PRINT "L{ser objektkod "&SEG$(FIL
     E$,POS(FILE$,".",1)+1,14): :
 500 PRINT "och skriver BASIC-program"
     : :
 510 PRINT "i filen MERGE."
 790 ! * Skriv "radnummer" CALL INIT i
      filen MERGE
 800 ! *
 810 LINE$=CHR$(HB(LINE))&CHR$(LB(LINE
     ))&CHR$(157)&CHR$(200)
 820 LINE$=LINE$&CHR$(4)&"INIT"&CHR$(0
     )
 830 PRINT #2:LINE$
 840 LINE=LINE+10
 940 GOSUB 1900
 950 A$=SEG$(A$,14,80)
 960 GOSUB 2030
 970 LINE$=ADR$(ADR)
 980 ! ********************************
     ********************
 990 ! * Omvandla objektkodens hexadec
     imala koder
1000 ! * till decimala koder
1010 ! *
1020 A$=SEG$(A$,6,80)
1030 FOR I=1 TO 80 STEP 5
1040 ! ********************************
     ********************
1050 ! * Kontrollera om objektkoden ka
     n laddas fr}n X-BASIC
1060 ! *
1070 TAG$=SEG$(A$,I,1)
1080 IF POS(TAG$$,TAG$,1)=0 THEN 1830
1090 ! ********************************
     ********************
1100 ! * Tag 7 eller 8 f|reg}r checksu
     mman, som {r sist
1110 ! * i varje record i objektkoden
1120 ! *
1130 IF TAG$="7" OR TAG$="8" THEN 1170
1140 GOSUB 1730
1150 LINE$=LINE$&HB$(HIGH)&LB$(LOW)
1160 NEXT I
1170 GOSUB 1950
1180 GOSUB 1900
1190 ! ********************************
     ********************
1200 ! * Tag 6 f|reg}r referenser
1210 ! *
1220 IF SEG$(A$,1,1)="6" THEN 1290
1230 GOSUB 2030
1240 LINE$=ADR$(ADR)
1250 GOTO 1020
1260 ! ********************************
     ********************
1270 ! * Ta reda p} referenser
1280 ! *
1290 PRINT : :"REFERENSER": :
1390 FOR I=1 TO 80 STEP 11
1400 TAG$=SEG$(A$,I,1)
1410 ! ********************************
     ********************
1420 ! * : anger slut p} objektkoden
1430 ! *
1440 IF TAG$=":" THEN 2180
1450 ! ********************************
     ********************
1460 ! * Tag 7 eller 8 f|reg}r checksu
     mman, som {r sist
1470 ! * i varje record i objektkoden
1480 ! *
1490 IF TAG$="7" OR TAG$="8" THEN 1660
1500 GOSUB 1730
1510 PRINT SEG$(A$,I+5,6): :
1570 GOSUB 2110
1580 LINE$=LINE$&HB$(HIGH)&LB$(LOW)
1590 REFTAB=REFTAB-8
1600 NEXT I
1650 LINE=LINE+10
```

```
1660 LINE$=ADR$(REFTAB)&LINE$
1670 GOSUB 1950
1680 GOSUB 1900
1690 GOTO 1390
1700 ! ********************************
     ********************
1710 ! * Omvandla hexadecimal kod till
      decimal kod i high- och low byte
1720 ! *
1730 HIGH=16*(POS(HEX$,SEG$(A$,I+1,1),
     1))+POS(HEX$,SEG$(A$,I+2,1),1)
1740 LOW=16*(POS(HEX$,SEG$(A$,I+3,1),1
     ))+POS(HEX$,SEG$(A$,I+4,1),1)
1750 RETURN
1760 ! ********************************
     ********************
1770 ! * Felhantering
1780 ! *
1790 CALL ERR(ERRCODE,ERRTYPE,ERRSEV,L
     INE)
1800 PRINT : :
1810 IF ERRCODE=130 THEN PRINT "Felakt
     igt filenamn.": : :: RETURN 390
1820 PRINT "Fel";ERRCODE;"i rad";LINE:
      :: GOTO 1840
1830 PRINT "Objektkoden kan inte ladda
     s": :"fr}n Extended BASIC."
1840 CLOSE #1
1850 CLOSE #2:DELETE
1860 STOP
1870 ! ********************************
     ********************
1880 ! * L{s ett record fr}n objektkod
     en
1890 ! *
1900 INPUT #1:A$
1910 RETURN
1920 ! ********************************
     ********************
1930 ! * Skriv "radnummer" CALL LOAD(x
     ,y,z) i filen MERGE
1940 ! *
1950 LINE$=CHR$(HB(LINE))&CHR$(LB(LINE
     ))&LOAD$&LINE$&CHR$(182)&CHR$(0)
1960 PRINT #2:LINE$
1970 LINE=LINE+10
1980 LINE$=""
1990 RETURN
2000 ! ********************************
     ********************
2010 ! * Ta reda p} decimal laddningsa
     dress
2020 ! *
2030 ADR=4096*(POS(HEX$,SEG$(A$,2,1),1
     ))+256*(POS(HEX$,SEG$(A$,3,1),1))
2040 ADR=ADR+16*(POS(HEX$,SEG$(A$,4,1)
     ,1))+POS(HEX$,SEG$(A$,5,1),1)
2050 IF ADR=0 THEN 1830
2060 IF ADR>32767 THEN ADR=ADR-65536
2070 RETURN
2080 ! ********************************
     ********************
2090 ! * Omvandla REF till ASCII-kod o
     ch flytta till LINE$
2100 ! *
2110 FOR J=I+5 TO I+10
2120 LINE$=LINE$&REF$(SEG$(A$,J,1))
2130 NEXT J
2140 RETURN
2150 ! ********************************
     ********************
2160 ! * Avsluta
2170 ! *
2180 CLOSE #1
2290 LINE$=ADR$(8196)&HB$(HB(REFTAB))&
     LB$(LB(REFTAB))
2300 GOSUB 1950
2350 ! ********************************
     ********************
2360 ! * Skriv fileslut i file MERGE
2370 ! *
2380 PRINT #2:CHR$(255)&CHR$(255)
2390 CLOSE #2
2400 END
```

Program 4.

Program 3 omvandlat till ett Basic-program som laddar det.

```
 100 ! ********************************
     ********************
 110 ! *****  MERGE
                  *****
 120 ! *****  Programmet (r skrivet av
      programmet   *****
 130 ! *****  LOADASMBAS.
                  *****
 140 ! *****  LOADASMBAS l{ser en asse
     mbler OBJEKT- *****
 150 ! *****  kod och gir om objektkod
     ens data till *****
 160 ! *****  CALL LOAD-satser.
                  *****
```

Program 3.

Ett exempel på ett Assemblerprogram i Assemblerad form.

```
000005BBLEX  924F4B3C7EBCFDFBFFFFB7E3CBEF20B9D50B7265B7373B20617F11FF  0001
92506B6E79B206BB6579B5072B6573B7320B4354B524CB3D20B746FB20727F2A5F    0002
9251CB6574B7572B6E74B6F20B4578B7465B6E64B6564B2042B4153B49437F29FF    0003
92532B2000B70D0BEF0FB8068BEF0DB9888BEF0CBA828BEF0ABB0B0BEF0B7F1E6F   0004
92548BBB08BEF05BD000B02E0B20BAB0200B0778B0201B24F4B0202B000B7F2BEF    0005
9255EB0420B2024B0200B0300B0201B2534B0202B001AB0420B2024B06A07F33AF   0006
92574B2602B06A0B261AB0265B24FFB000DB06A0B261AB02A5B250CB00157F2B1F   0007
9258AB06A0B261AB02E5B2521B0011B04C5B0200B0300B0420B202BB09817F2FCF   0008
925A0B02B1B00D0B130AB0601B1602B0201B00CB0AB1B0420B2020B02207F318F    0009
925B6B0004B10F0B04E0B8375B0420B201CB9820BB375B24FEB1314BD0E07F2A7F   0010
925CCB837CB20E0B2532B1301B10E0BC145B1606B05B5B0200B01E1B04207F2CFF   0011
925E2B2030B10D8B04C5B0200B01E0B0420B2030B10D2B06A0B2602B04C07F2ECF   0012
925F8BD800B837CB02E0B83E0B045BB04C0BD060B24FDB0221B6000B04207F290F   0013
9260EB2020B05B0B02B0B0300B16FAB045BBC03BBC0BBBC0FBBA0C0BD0727F25BF   0014
92624B0221B6000B0420B2020B05B0B80C0B16FBB045B7F635F                  0015
6254ESBBLEX7FCF3F                                                    0016
:       99/4 AS                                                      0017
```

```
 170 ! *****  LOADASMBAS (r skrivet av
                  *****
 180 ! *****  Birje H{ll
                  *****
 190 ! *****  Vasav{gen 101
                  *****
 200 ! *****  175 32  J{rf{lla
                  *****
 210 ! ********************************
     ********************
 220 CALL INIT
 230 ! ********************************
     ********************
 240 ! * Ladda assemblerprogrammet
 250 ! *
 260 CALL LOAD(9460,60,126,207,223,255
     ,255,126,60,239,32,157,80,114,101
     ,115,115,32,97)
 270 CALL LOAD(9478,110,121,32,107,101
     ,121,80,114,101,115,115,32,67,84,
     82,76,61,32,116,111,32,114)
 280 CALL LOAD(9500,101,116,117,114,11
     0,116,111,32,69,120,116,101,110,1
     00,101,100,32,66,65,83,73,67)
 290 CALL LOAD(9522,32,0,112,208,239,1
     5,128,104,239,13,152,136,239,12,1
     68,40,239,10,176,176,239,8)
 300 CALL LOAD(9544,184,8,239,5,208,0,
     2,224,32,186,2,0,7,120,2,1,36,244
     ,2,2,0,8)
 310 CALL LOAD(9566,4,32,32,36,2,0,3,0
     ,2,1,37,52,2,2,0,26,4,32,32,36,6,
     160)
 320 CALL LOAD(9588,38,2,6,160,38,26,2
     ,101,36,255,0,13,6,160,38,26,2,16
     5,37,12,0,21)
 330 CALL LOAD(9610,6,160,38,26,2,229,
     37,33,0,17,4,197,2,0,3,0,4,32,32,
     40,9,129)
 340 CALL LOAD(9632,2,129,0,208,19,10,
     6,1,22,2,2,1,0,200,10,129,4,32,32
     ,32,2,32)
 350 CALL LOAD(9654,0,4,16,240,4,224,1
     31,117,4,32,32,28,152,32,131,117,
     36,254,19,20,208,224)
 360 CALL LOAD(9676,131,124,32,224,37,
     50,19,1,16,224,193,69,22,6,5,133,
     2,0,1,225,4,32)
 370 CALL LOAD(9698,32,48,16,216,4,197
     ,2,0,1,224,4,32,32,48,16,210,6,16
     0,38,2,4,192)
 380 CALL LOAD(9720,216,0,131,124,2,22
     4,131,224,4,91,4,192,208,96,36,25
     3,2,33,96,0,4,32)
 390 CALL LOAD(9742,32,32,5,128,2,128,
     3,0,22,250,4,91,192,59,192,187,19
     2,251,160,192,208,114)
 400 CALL LOAD(9764,2,33,96,0,4,32,32,
     32,5,128,128,192,22,248,4,91)
 410 ! ********************************
     ********************
 420 ! * Uppdatera REF/DEF-tabellen
 430 ! * Referenser
 440 ! * SBBLEX
 450 ! *
 460 CALL LOAD(16376,83,66,66,76,69,88
     ,37,78)
 470 ! ********************************
     ********************
 480 ! * Uppdatera pekaren till REF/DE
     F-tabellen
 490 ! *
 500 CALL LOAD(8196,63,248)
 510 END
```

GRAFIK i FORTH

av Björn Gustavsson

I Nittinian 83-2 skrev jag om hur man definierar nya tecken i TI BASIC. Som du minns måste man först rita tecknet i ett rutnät på papper och sedan översätta till hexkoder. Ett sätt att göra det lättare för sig är att använda ett program som gör det möjligt att rita tecknet i ett rutnät på skärmen och låta datorn göra det tråkiga översättningsjobbet. I bruksanvisningen finns ett sådant program i BASIC.

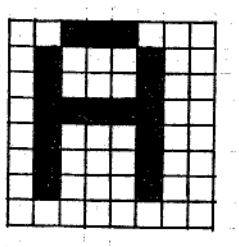
Här skall vi titta på ett sådant program i FORTH. Det finns listat här bredvid på skärmarna 17 till 20. Skärm 17 innehåller ord som går över till grafikläge (32 tecken per rad som i BASIC) och tillbaka till textläge (40 tecken per rad). Tecknet "←" som förekommer på skärm 17 är ett understrykningstecken och matas in med FCTN U.

När dessa skärmar har laddats kan vi börja definiera tecken med ordet MC . (MC står för Make Character, vilket bör låta bekant för den som har provat TI LOGO. Sättet att definiera tecknen är också väldigt lika LOGOs.)

När MC anropas skall teckenkoden för tecknet som du vill definiera ligga överst på stacken. Prova t ex följande:

```
65 MC
```

Eftersom 65 är teckenkoden för A får vi upp ett rutnät med 8 gånger 8 rutor innehållande ett stort A:



Nu kan A ändras. Man kan förflytta sig i rutnätet med piltangenterna (E, S, D och X). Om man bara trycker ned en piltangent flyttas markören i motsvarande riktning och rutan som markören stod på släcks. Om man samtidigt trycker FCTN och en piltangent flyttas markören i pilens riktning och rutan som markören stod på tänds. Om man försöker flytta utanför rutnätets gränser flyttas markören till andra sidan av rutnätet.

Prova att hålla ned FCTN och trycka S åtta gånger. Detta drar ett streck under A:et. (Markören står i början alltid nere i högra hörnet.) Om det finns några A:n på skärmen kommer dessa att ändras omedelbart varje gång en ruta fylls i. Skulle du inte tycka om det nya utseendet på A kan du sudda bort strecket genom att trycka på S åtta gånger igen, men utan att hålla nere FCTN.

Prova också att trycka FCTN 3 (ERASE).

Titta vad du har ställt till med! Hela A:et försvann. Felet är lätt att rätta till. Använd figuren av A ovan och rita in på nytt i rutnätet.

Om du har tröttnat på att leka med A kan du gå ur med FCTN 4 (CLEAR).

På motsvarande sätt kan du bygga upp dina egna tecken. Lägg bara teckenkoden överst på stacken och skriv MC. Alla teckenkoder från 0 till 255 kan anges. Teckenkoderna 32 till 126 är fördefinierade som bokstäver, siffror och övriga tecken. Du bör heller inte omdefiniera 30 och 31 eftersom dessa tecken används av MC för att visa rutnätet. Dessutom är vissa andra teckenkoder under 32 definierade för att kunna ge stortext på skärmen. (Hur man får fram stortext tänkte jag visa i en kommande artikel.)

Det finns några trick som sparar på hjärnan (man slipper komma ihåg ASCII-kodstabellen). Dels kan man skriva ordet ASCII och ett godtyckligt tecken så läggs automatiskt teckenkoden på stacken. Exempel:

```
ASCII Ä MC
```

Detta lägger först teckenkoden för Ä på stacken och anropar sedan MC så att Ä kan ändras. Ett liknade trick består i att skriva:

```
KEY MC
```

och trycka ENTER. Tryck sedan ned en tangent, t ex &. Du får upp tecknet i rutnätet och kan ändra det.

Slutligen skall vi nämna ordet MS (Make Sprite). Det används som MC men rutnätet har storleken 16 gånger 16.

När man har definierat sina tecken måste man kunna spara dem på något sätt. Detta kan göras på två sätt.

Metod 1: Om något/några av tecknen 0 till 127 skall sparas kan SAVE-SYSTEM eller CSAVE-SYSTEM användas. Dessa ord sparar hela ordlistan och dessutom definitionerna för tecken 0 till 127 på antingen skiva eller kassettband.

Metod 2: Om metod 1 inte kan användas måste i stället VSAVE användas. VSAVE sparar 1K av VDP-minnet på en skärm. Överst på stacken skall finnas start-adressen i VDP-minnet och numret på skärmen där det skall sparas. För att spara tecken 128 till 255 på skärm 40, skriv:

```
128 CHAR.ADR 40 VSAVE
```

För att spara tecken 0 till 127 på skärm 6, skriv:

```
0 CHAR.ADR 6 VSAVE
```

För att hämta tillbaka tecknen används VLOAD som gör motsatsen. För att hämta tillbaka tecken 128 till 255, skriv:

```
128 CHAR.ADR 40 VLOAD
```

Låt oss slutligen titta på orden på skärm 17. GRAPH ställer om skärmen till grafik-läge, dvs det går in 32 tecken per rad. Det är grafik-läge som BASIC alltid använder. TEXT-MODE ställer tillbaka skärmen i text-läge. I text-läge går det in 40 tecken per rad, men alla tecken måste ha samma färg och sprites kan inte användas.

Ordet SETCOLOR sätter färgen på en grupp av 8 tecken (i grafik-läge). Följande tal skall finnas på stacken:

```
förgrundsfärg bakgrundsfärg tecken
```

Färgkoderna för förgrunds- och bakgrundsfärg anges från 0 till 15 som i assembler. Det innebär att alla färgkoder skall minskas med 1 om BASICs färgtabeller används. T ex anges mörkröd i BASIC som 7, men måste anges som 6 i FORTH.

Teckenkoden anger ett tecken, men alla tecken i samma teckengrupp kommer att få samma färg.

Exempel:

1 15 ASCII A SETCOLOR

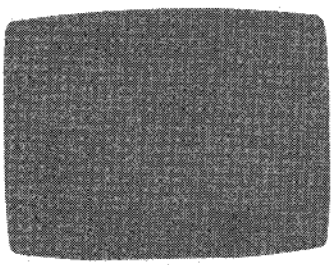
Detta ger A (och tecknen @, B, C, D, E, F och G) förgrundsfärgen svart och bakgrundsfärgen vit.

```
SCR #17
 0 ( Grundläggande grafik                                          )
 1 FORTH DEFINITIONS HEX
 2
 3 : SETCOLOR ( f b c --) ( sätter förgrunds- och bakgrundsfärg)
 4    8 / 380 + ROT 10 * ROT + SWAP V! ;
 5 : SCRN ( bredd slut --)
 6    SCRN+END ! SCRN+WIDTH ! 0 SCRN+START ! ;
 7 : GRAPH ( ställer in grafikläge)
 8    20 300 SCRN  (GRAPH)  ;
 9 : TEXT-MODE  ( ställer in textläge)
10    28 3C0 SCRN  (TEXT-MODE)  ;
11   --)
12
13
14
15

SCR #18
 0 ( Grundläggande grafik                                          )
 1 0 VARIABLE C#  0 VARIABLE CX  0 VARIABLE CY   0 VARIABLE SIZE
 2 : PUT ( n --) CX @ CY @ GOTOXY EMIT CX @ CY @ GOTOXY ;
 3 : ADRBIT ( -- adr bit) C# @ CHAR.ADR CY @ +
 4    CX @ 7 ) IF 10 + THEN 80 CX @ 7 AND SRC ;
 5 : INIT-GRID   SIZE @ 0 DO SIZE @ 0 DO I CX ! J CY ! ADRBIT
 6    SWAP V@ AND IF 1D ELSE 1F THEN PUT LOOP LOOP ;
 7 : BIT-OFF   ADRBIT OFF XOR SWAP ANDCHAR 1F PUT ;
 8 : BIT-ON    ADRBIT SWAP ORCHAR 1D PUT ;
 9 : SHOW   CX @ SIZE @ 1- AND DUP CX !
10    CY @ SIZE @ 1- AND DUP CY ! GOTOXY ;
11 : CLEARCHAR   C# @ CHAR.ADR SIZE @ 8 =
12    IF 8 ELSE 20 THEN 0 HCHAR ;
13 DECIMAL
14   --)
15

SCR #19
 0 ( Grundläggande grafik                                          )
 1 : ED-CHAR
 2    INIT-GRID UPPER
 3    BEGIN FALSE KEY
 4      CASE ASCII E ( upp)     OF BIT-OFF -1 CY +! SHOW ENDOF
 5           ASCII X ( ned)     OF BIT-OFF  1 CY +! SHOW ENDOF
 6           ASCII S ( vänster) OF BIT-OFF -1 CX +! SHOW ENDOF
 7           ASCII D ( höger)   OF BIT-OFF  1 CX +! SHOW ENDOF
 8           11 ( upp)          OF BIT-ON  -1 CY +! SHOW ENDOF
 9           10 ( ned)          OF BIT-ON   1 CY +! SHOW ENDOF
10           08 ( vänster)      OF BIT-ON  -1 CX +! SHOW ENDOF
11           09 ( höger)        OF BIT-ON   1 CX +! SHOW ENDOF
12           02 ( avbryt)       OF DROP TRUE ENDOF
13           07 ( radera)       OF CLEARCHAR INIT-GRID ENDOF
14      ENDCASE
15    DUP UNTIL DROP LOWER ; --)

SCR #20
 0 ( Grafik)
 1 : MC ( c --)
 2    C# ! 8 SIZE ! ED-CHAR 1 8 GOTOXY ;
 3 : MS ( c --)
 4    C# ! 16 SIZE ! ED-CHAR 1 16 GOTOXY ;
 5
 6
 7
 8
 9
10
11
12
13
14
15
```



EMPTY SCREEN = TOM SKÄRM

Aforismer för PROGRAMMERARE

Hämtat ur STJÄRNRÄKNAREN Red. Paul Schlüter

1. Grundläggande regler för datoraritmetik (flyttal):

   a) A=A plus minus någonting i sista siffran
   b) A+B är skilt från B+A
   c) A*B är skilt från B*A
   d) A*(B+C) är skilt från A*B+A*C
   e) A upphöjt till n är skilt från A*A*A... (n gånger).

2. Datorer gör vad man säger åt dem att göra, inte vad man vill de skall göra.

3. Det finns ingen som uppskattar dina program lika mycket som du själv.

4. Den förste (utom du själv) som använder ditt program kommer att hitta ett fel i det.

5. Inget program fungerar första gången det körs. Var speciellt misstänksam om du får det resultat du väntat dig första gången.

6. Om du förgäves prövat alla metoder att hitta felet i ditt program, förklara det då för ett sjuårigt barn.

7. Varje halvtimme av förberedande analys sparar två timmars programmering och felsökning.

8. Det är mycket lättare att skriva ett nytt program än att ändra ett gammalt.

9. Nittiofem procent av tiden ägnar man åt fem procent av problemet.

10. När du har provat allt annat, läs bruksanvisningen.

11. När man verkligen har ett problem är bruksanvisningen värdelös.

12. Sina variabler känner man till med större noggrannhet än sina konstanter.

13. Om man tar med fler siffror blir ens svar mer trovärdiga.

14. Femtio procent mer minne innebär att kapaciteten fördubblas.

15. Man har aldrig tillräckligt med minnesutrymme.

16. Alla program kan förbättras.

17. Inget program är felfritt.

18. Inget program är idiotsäkert.

19. Varje problem har mer än en lösning, dvs två program är aldrig lika.

TI 59

KALENDER

ETT KALENDERPROGRAM FÖR BÅDE JULIANSK OCH GREGORIANSK KALENDER ("GAMLA" OCH "NYA STILEN")

av Lars Hedlund

I samband med en notis i SvD apropå skottdagen 1984 blev jag uppringd av Per-Erik Holmberg som påpekade att anvisningarna till ML-20 inte gäller, då de påstår att programmet är giltigt "efter 1582" då i själva verket den gregorianska kalendern inte infördes i Sverige förrän 1753.

Jag började forska i Nordisk Familjebok m fl böcker och kom fram till följande.

Den gregorianska kalendern infördes i katolska länder den 15 okt 1582 enligt vår tidräkning. Genom det felaktiga antagandet att årets längd vore 365,25 dygn i st f (mer exakt) 365,2425 dygn hade många skottdagar inlagts så att vårdagjämningen flyttats bort från den 21 mars, där den förutsatts ligga vid påskregelns utformning vid kyrkomötet i Nicaea år 325. Genom att helt enkelt ta bort dagarna 2-15 okt kom man "i takt" igen. De protestantiska länderna följde emellertid inte med utan behöll den julianska kalendern ända till mitten av 1700-talet. Den första dagen enligt gregoriansk kalender i det brittiska imperiet (inkl de amerikanska kolonierna!) var torsdagen den 14 sept 1752. I Sverige var första dagen enligt "nya stilen" torsdagen den 1 mars 1753. I båda fallen hade de närmast föregående elva dagarna strukits. I Sverige hade man fr o m den 1 mars 1700 t o m den 30 febr 1712 (ja, faktiskt!) krånglat till det hela ytterligare genom att varken ha juliansk eller gregoriansk kalender utan ligga en dag före den gamla julianska kalendern.

Ti-59-modulens program ML-20 visar sig vid närmare betraktande ha en hel massa brister. Dels kan det bara räkna efter gregoriansk kalender, dels har det en spärr mot årtal före 1582 så man inte kan använda det för juliansk kalender före det året ens med en korrektionstabell. I den användarframställda modulen till HP-41 visade det sig finnas en mycket bättre algoritm, som medger valfri användning av juliansk eller gregoriansk kalender ända tillbaka till 1 mars år 0. Med utgångspunkt från det så gjorde jag vidstående program som faktiskt är ännu användarvänligare än det i "PPC ROM".

Anvisningar: Ge datum på formen AAAA.MMDD och tryck A för juliansk kalender eller E för gregoriansk. "Julian Date" som är ett i astronomiska sammanhang använt ordningsnummer för dagen visas. Om nu D trycks visas veckodag med en siffra från 0 (söndag) till 6 (lördag). Antal dygn mellan två datum kan beräknas genom att det förstas Julian Date läggs i t-reg och att man efter det andras trycker "-x:t=".

Ex 1. Hur många dygn är det mellan samma klockslag den 17 feb 1753 enligt juliansk kalender och den 1 mars 1753 enligt gregoriansk? Tryck 1753.0217 A och se 2361389. Tryck x:t och 1753.0301 E och se 2361390. "-x:t =" ger alltså 1. Så var det i Sverige - efter onsdagen den 17 febr 1753 följde torsdagen den 1 mars.

Ex 2. Vilken veckodag var det i Sverige den "historiska" dagen den 30 febr 1712? Minska enligt texten ovan med 1 dag, tryck 1712.0229 A (juliansk kal.) så fås 2346425. Tryck D så visas 5, alltså fredag.

Ex 3. Vilken veckodag valdes Gustav Vasa till Sveriges konung (6 juni 1523)? Gregorianska kalendern var ännu inte uppfunnen. 1523.0606 A ger 2277490. D ger 6, alltså lördag.

Ex 4. Vilken veckodag dog Gustav II Adolf? I Sverige och protestantiska Tyskland gällda "gamla stilen". Tryck 1632.1106 A så visas 2317456. D ger 2, alltså tisdag.

(Nu säger kanske någon: "Hur vet vi att '6 nov' inte är enligt 'nya stilen', dvs enligt katolsk tidräkning i Tyskland?" Grimberg relaterar emellertid i 'Svenska folkets underbara öden' en episod som visar att svensk kalender gällde. En "lördag i början av november 1632" hade en svensk officer med fyra man kommit till staden Delitzsch ca 35 km norr om Lützen för att lämna ett viktigt meddelande till kungen före den väntade drabbningen. Medan de var i staden rapporterades att kejserliga trupper stod utanför stadsporten och uppmanade staden kapitulera. En svensk trumpetare skickades då upp i rådhustornet och fick där blåsa sitt regementes signal. De kejserliga hade hört den förut och trodde att en större svensk styrka fanns i staden och gav sig i väg. Till minne av detta blåstes varje lördagsmorgon ända in på 1900-talet samma signal från rådhustornet. Enligt "nya stilen" var den 6 nov den första lördagen i november 1632 och då var redan slaget vid Lützen. Enligt "gamla stilen" var däremot 3 nov en lördag, vilket stämmer gott!)

För den som vill köra detta program på t ex TI-99 lämnas här formeln för beräkning av Julian Date.

$$JDN = Int(Int(Int(367 * y') - 1.75 * Int(y') + D) - 0.75 * Int(y'/100)) + 1721115$$

där y' = år + (månad - 2.85)/12 och D är dag.

Det underprickade uttrycket Int(y'/100) utbytes vid juliansk kalender mot 2.

TI 59

KALENDER

Sedan jag skrivit ovanstående tyckte jag att programmet borde kompletteras med omvändningen, dvs från Julian Date till datum. Det fanns ju gott om plats. Detta avsnitt ligger i stegen 112-249. Beräkningsgången är även där enligt "PPC ROM".

N är antal dagar sedan början av mars år 0, dvs N = JDN - 1 721 119. Ett gregorianskt sekel är i genomsnitt 36524.25 dagar och antalet gregorianska sekel sedan mars år 0 är därför C = Int((N-e)/35624.25 där e ligger mellan 0 och 0.25 (0.2 användes). N' = N + C - Int(C/4) ger då antalet dagar sedan mars år 0 om alla jämna sekelår varit skottår (som de faktiskt var i julianska kalendern). För juliansk kalender väljes N' = N + 2 eftersom båda kalendrarna stämde under sekel nr 2. Härifrån blir beräkningen lika för båda kalendrarna.

Låt Y' och M' vara "skiftade" år och månad, där ett "skiftat" år går från mars (M' = 0) ett år med normalt årtal till följande års februari (M' = 11).
Då blir Y', dvs antalet "skiftade" år sedan mars år 0, Int((N' - e')/365.25) där e' valts till 0.2 enligt samma grunder som e ovan. Antalet dagar sedan början av det "skiftade" året är N" = N' - Int(365.25 * Y'), och den "skiftade" månaden fås ur M' = Int((N" - d)/30.6) där d ligger mellan 0.4 och 0.6 (0.5 har valts). 30.6 är den genomsnittliga längden av en månad under en femmånaderscykel som börjar med mars. Dagen i månaden blir då D = Int(N" - 30.6 * M' + d'), där d' är lika d ovan. Slutligen får man verkligt år och månad ur Y = Y' och M = m' + 3 om M' <= 9 och Y = Y' + 1 och M = M' - 9 om M' > 9.

Anvisningar för Julian Date till datum: Ge Julian Date och tryck B för juliansk, C för gregoriansk kalender.

Ex 5. Vilket datum enligt julianska kalendern kom före 14 sept 1752 enligt gregoriansk? 1752.0914 E ger 2361222; - 1 = ger 2361221; B ger 1752.0902. Den 2 sept 1752 var sista dagen enligt juliansk kalender i brittiska imperiet inkl. USA.

PS. Per-Erik påpekade för mig att år 0 aldrig har existerat utan hette 1 f.Kr. Den som vill undersöka ett datum detta år måste dock kalla det år 0.

```
KALENDER
001  11  A
004  15  E
083  02  2
098  14  D
113  13  C
116  12  B
156  02  2

000  76  LBL
001  11  A
002  22  INV
003  76  LBL
004  15  E
005  86  STF
006  00  00
007  75  -
008  59  INT
009  82  HIR
010  08  08
011  95  =
012  65  ×
013  01  1
014  00  0
015  00  0
016  75  -
017  59  INT
018  82  HIR
019  07  07
020  95  =
021  82  HIR
022  06  06
023  82  HIR
024  17  17
025  75  -
026  02  2
027  93  .
028  08  8
029  05  5
030  95  =
031  55  ÷
032  01  1
033  02  2
034  85  +
035  82  HIR
036  18  18
037  95  =
038  82  HIR
039  08  08
040  65  ×
041  03  3
042  06  6
043  07  7
044  95  =
045  59  INT
046  75  -
047  82  HIR
048  18  18
049  59  INT
050  65  ×
051  07  7
052  55  ÷
053  04  4
054  85  +
055  82  HIR
056  16  16
057  65  ×
058  01  1
059  00  0
060  00  0
061  95  =
062  59  INT
063  75  -
064  93  .
065  07  7
066  05  5
067  65  ×
068  22  INV
069  87  IFF
070  00  00
071  00  00
072  83  83
073  53  (
074  82  HIR
075  18  18
076  55  ÷
077  01  1
078  00  0
079  00  0
080  54  )
081  59  INT
082  76  LBL
083  02  2
084  85  +
085  01  1
086  07  7
087  02  2
088  01  1
089  01  1
090  01  1
091  05  5
092  95  =
093  59  INT
094  22  INV
095  58  FIX
096  91  R/S
097  76  LBL
098  14  D
099  85  +
100  01  1
101  75  -
102  53  (
103  24  CE
104  55  ÷
105  07  7
106  54  )
107  59  INT
108  65  ×
109  07  7
110  95  =
111  91  R/S
112  76  LBL
113  13  C
114  22  INV
115  76  LBL
116  12  B
117  86  STF
118  00  00
119  75  -
120  01  1
121  07  7
122  02  2
123  01  1
124  01  1
125  01  1
126  09  9
127  93  .
128  02  2
129  95  =
130  82  HIR
131  08  08
132  87  IFF
133  00  00
134  01  01
135  56  56
136  55  ÷
137  03  3
138  06  6
139  05  5
140  02  2
141  04  4
142  93  .
143  02  2
144  05  5
145  95  =
146  59  INT
147  75  -
148  53  (
149  24  CE
150  55  ÷
151  04  4
152  54  )
153  59  INT
154  95  =
155  76  LBL
156  02  2
157  82  HIR
158  38  38
159  82  HIR
160  18  18
161  55  ÷
162  03  3
163  06  6
164  05  5
165  93  .
166  02  2
167  05  5
168  95  =
169  59  INT
170  82  HIR
171  07  07
172  65  ×
173  03  3
174  06  6
175  05  5
176  93  .
177  02  2
178  05  5
179  95  =
180  59  INT
181  82  HIR
182  58  58
183  82  HIR
184  18  18
185  75  -
186  93  .
187  05  5
188  95  =
189  55  ÷
190  03  3
191  00  0
192  93  .
193  06  6
194  95  =
195  59  INT
196  82  HIR
197  06  06
198  65  ×
199  03  3
200  00  0
201  93  .
202  06  6
203  94  +/-
204  85  +
205  82  HIR
206  18  18
207  85  +
208  93  .
209  05  5
210  95  =
211  59  INT
212  52  EE
213  04  4
214  94  +/-
215  82  HIR
216  37  37
217  25  CLR
218  03  3
219  82  HIR
220  36  36
221  01  1
222  03  3
223  32  X:T
224  82  HIR
225  16  16
226  22  INV
227  77  GE
228  02  02
229  37  37
230  01  1
231  02  2
232  82  HIR
233  56  56
234  01  1
235  82  HIR
236  37  37
237  82  HIR
238  16  16
239  52  EE
240  02  2
241  94  +/-
242  82  HIR
243  37  37
244  25  CLR
245  82  HIR
246  17  17
247  58  FIX
248  04  04
249  91  R/S
```

TI59
HALVLEDARFYSIK

MED TILLÄMPNINGAR PÅ METALL-HALVLEDARÖVERGÅNGAR

Jorge de Sousa Pires

Efter att ha läst Programbiten i alla dessa år som den funnits - med stor glädje! - har jag nu kommit mig för att skicka in ett bidrag. Länge har jag tänkt att det är lite väl skralt med program med vetenskaplig anknytning, dvs program som kan användas i en forskningssituation eller dyl. Mest har det varit spel eller fantastiska slumpgeneratorer, testprogram till nämnda generatorer etc.** I längden är det inte så intressant, men här skall det inte klagas på redaktörerna, ni väljer säkert bland inkomna bidrag....

Nåväl, här är ett litet annorlunda program. Det kan användas av alla, men mest intresserade blir de som sysslar med halvledarelektronik till vardags (forskning, utbildning eller i industrin). Det är väl inte så många människor i hela landet men alla publicerade program är inte heller intressanta för alla människor... varför skulle man plotta de nordiska flaggorna i svart(?)-vit när det går att rita dem färgglada... Varje civilingenjör träffar någon gång på en kurs i halvledarfysik och det skall här tilläggas att månget problem kan lösas genom att trycka på de rätta knapparna... inom några sekunder!

En annan iaktagelse som jag gjort är att andra föreningar erbjuder en hel del vetenskapliga program, att döma av listorna som ni presenterat på sistone. Vi skall väl inte vara sämre! Alltså, till verket.

Manuskriptet inkommet 1982-11-05 ** / CSCH.

PROGRAMMET HALVLEDARFYSIK

Programmet skrevs direkt på engelska, eftersom det är elektronikens språk (och många av termerna har inte heller någon bra översättning). Flödesdiagrammen och vissa andra instruktioner är också skrivna på engelska; de som är intresserade av programmet behärskar säkert engelska, varför jag inte bryr mig om att översätta just detta.

Programmet heter "Halvledarfysik" i största allmänhet, eftersom det ger så många storheter som är helt oberoende av huvudsyftet med programmet, nämligen beräkningar i anslutning till en metall-halvledarövergång, vanligen kallad Schottkydiod. Denna krets består i enklaste formen av en halvledarplatta belagd med en metall, se Fig. 1. I övergången mellan metallen och halvledaren bildas ofta en barriär som hindrar strömmen (flödet av laddningsbärare) i ena riktningen (därav namnet "diod").

Programmet är mycket allmänt och flexibelt men inlagrade materialkonstanter är för KISEL. Om materialkonstanter för andra halvledare matas in, kan programmet självfallet användas för andra halvledare.

Vilken metall som helst kan användas emedan beräkningarna är "kopplade" till metallen endast via ovannämnda barriär. Du måste alltid ange något värde; känner du inte till barriräran, ge ett rimligt värde (ex:vis 0.7 eV).

En bild över en sådan övergång visas i Fig. 1, där alla (gängse) beteckningar är definierade.

METALL-HALVLEDARÖVERGÅNG

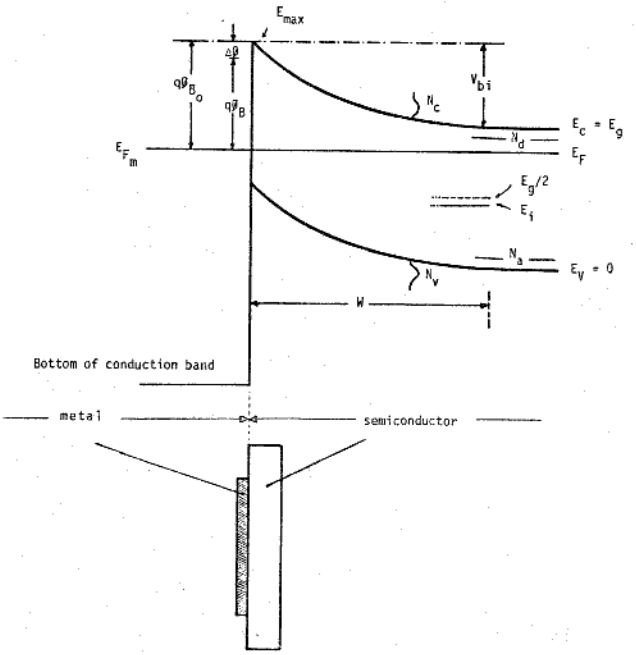


Fig. 1 Metall-halvledarövergång

Flödesdiagram 1 visar den största subrutinen. Där beräknas alla storheter som har med kontakten att göra. Först lagras alla fysikaliska konstanter (enl. standardverk) i minnen R7-R11. Alla (utom den i R11) är oberoende av halvledarmaterial (se flödesdiagrammet).

Den första körningen (initieringen) har temperaturen 293 Kelvin som defaultvärde (realistisk rumstemperatur).

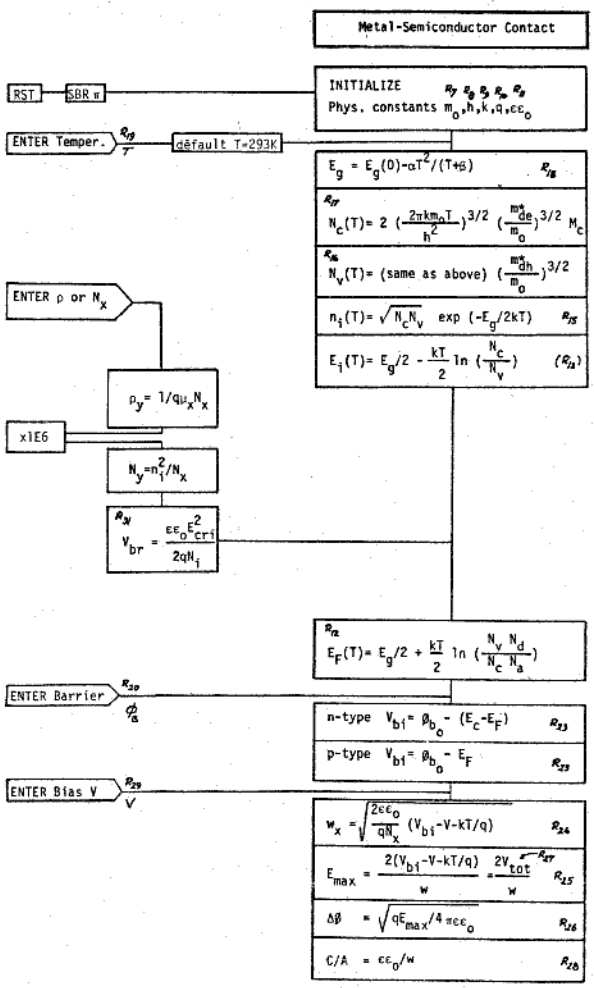
Att temperaturen är den första variabeln som matas in är pedagogiskt mycket riktigt: halvledarnas största svaghet är ju deras temperaturberoende!

Här beräknas sedan, i tur och ordning: bandgapet E_g, effektiva tillståndstätheter (N_c och N_v), koncentrationen av laddningsbärare om inga störatomer funnes (n_i) samt resulterande (egenledningens) Ferminivå (E_i).

Är skrivaren ansluten, skrivs alla storheter ut.

TI59

FLÖDESDIAGRAM 1



DOPNINGEN

Beträffande dopningen, har man antingen resistiviteten eller dopningskoncentrationen. Det här är, av praktiska skäl, det enda stället i programmet där enheten "centimeter" används, eftersom resistiviteten ofta anges i Ohmcm; den skall matas in via tangenten D eller D', beroende på huruvida dopningen är av n eller p-typ.

Beträffande detta med tangenterna, se skissen över texten på magnetkortet (Fig. 2).

cm^{-1} ⇄ μm	ϕ_B↓	N_x↓...E_F	N_a ⇄ ρ_p	CV...ϕ_{Bo}
μm ⇄ eV	V↓...C/A	T↓...E_i	N_d ⇄ ρ_n	C(pF)↓

Fig. 2 Magnetkort nr 1 (slavkortet nr 2 innehåller bl a konstanter).

Om resistiviteten anges och D trycks, fås dopningskoncentrationen i cm^{-3} och tvärtom. Programmet använder, för denna beräkningen, de materialkonstanter som bäst matchar publicerade kurvor (ref 1, sid 43) och kan användas för resistiviteter från 2 Ωcm (och högre).

Defaulttyp för dopningen är n-typ. För att tala om att det handlar om p-typ skall tangenten D' tryckas in, minst en gång; är dopningskoncentrationen känd skulle Du egentligen inte behöva trycka på D' men det skall ändå göras därför att det ställer flagga 1; finns dopningskoncentrationen i fönstret, kan D' tryckas två gånger, vilket då ställer flagga 1 och lämnar åter dopningskoncentrationen i fönstret.

Dopningskoncentrationen skall omvandlas till m^{-3} (dvs multipliceras med 1E6) INNAN den matas in via C'. Efter inmatningen, fås breakdown-spänningen (V_{br}), resistiviteten (ρ), dopningen (N_x=N_a eller N_d), storheten n_i^2/N_x, samt Ferminivån för denna temperatur och dopning.

SCHOTTKYDIODEN

Hittills har alltså själva Schottkydioden inte nämnts därför att den inte har kommit in i beräkningarna.

Har man en sådan diod, matas barriären Φ_{Bo} via B' (den matas in i electronvolt, eV).

Utsätts övergången för en spänning V, bildas det ett utarmningsområde (w) som "ger" diverse andra storheter, såsom den maximala fältet vid övergången (E_{max}), barriärsänkningen ($\Delta\Phi$) samt kapacitans per ytenhet (C/A). Dessa värden beräknas omedelbart så snart spänningen matats in via B.

Alla dessa värden lagras i olika register, vilket visas i flödesdiagrammet. För en cirkulär diod, kan ytan beräknas enkelt: om diameter finns i fönstret, används SBR P→R, varpå ytan lagras i R32 och visas i fönstret; ytan kan också lagras in på vanlig väg, naturligtvis.

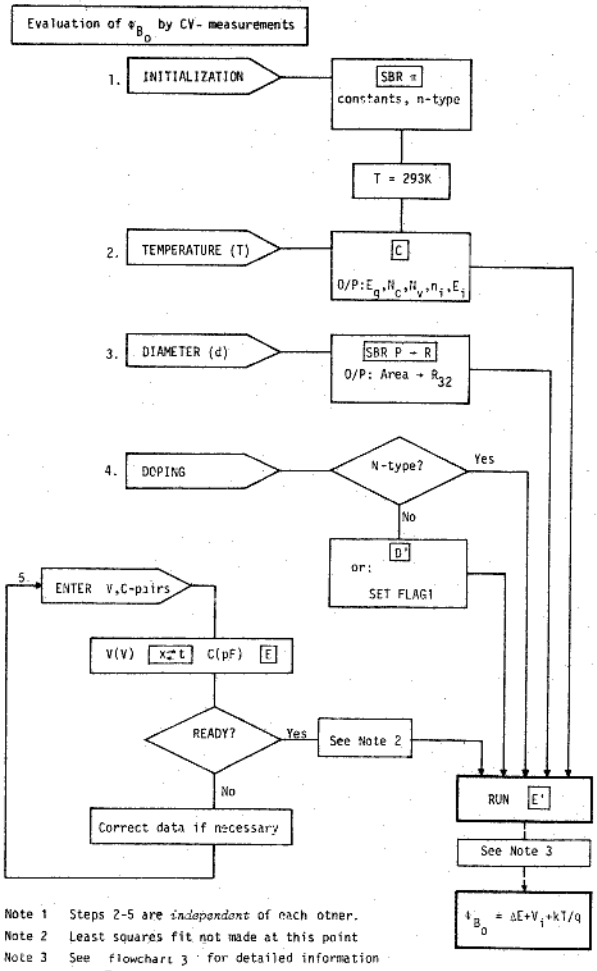
KAPACITANSMÄTNINGAR

En mycket vanlig typ av mätning (på laborationer bl annat!) på en sådan diod är den s k CV-mätningen, en metod att uppskatta barriären genom att mäta den resulterande kapacitansen (C) som en viss spänning (V) ger. Dessa datapar plottas sedan på formen $1/C^2$ kontra spänningen. Lutningen ger dopningskoncentrationen, och interceptet (V_i) ger småningom barriären (se figuren bredvid flödesdiagram 3).

I programmet finns ett antal mycket kraftfulla subrutiner som beräknar barriären så snart två par uppmätts (men flera par måste tas för att förbättra noggranheten i mätningen). Normalt tar detta ganska lång tid om det görs för hand, här är det fråga om sekunder!

TI59

FLÖDESDIAGRAM 2



Note 1 Steps 2-5 are *independent* of each other.
Note 2 Least squares fit not made at this point
Note 3 See flowchart 3 for detailed information

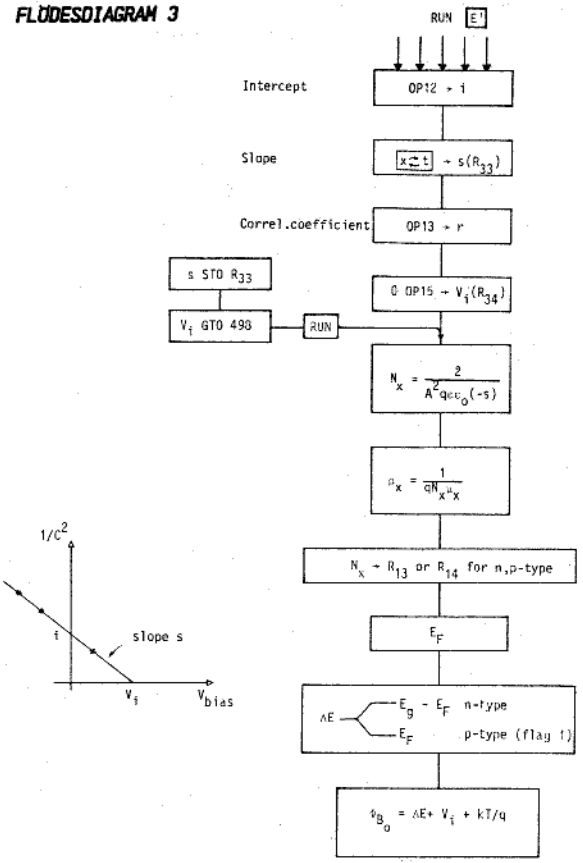
Betrakta flödesdiagram nr 2.
Punkterna 1, 2, och 3 tarvar inte närmare förklaring emedan de diskuterats tidigare. Utför Du en CV-mätning känner Du alltså inte till dopningskoncentrationen - det är ett av utgångsvärden som söks! Därför skall Du, under punkt 4 INTE ge dopningskoncentrationen (det gör dock inget om Du ändå skulle ge något värde, programmet skippar det!). Det enda som skall göras är att ställa/nollställa flagga 1 för p/n-typ (n-typ är default, som sagt).

Under punkt 5 skall de olika dataparen ges. Flödesdiagrammet är självinstruerande. Så snart två par matats in, kan en minsta kvadratanpassning göras. Beräkningen startar så snart E' tryckts in.

Flera datapar kan användas allteftersom. De kan också strykas eller rättas till (se manualen för TI59, det är standard OP-funktioner som används).

Om lutningen (s) samt interceptet är kända, kan programmet köras från steg 498 och framåt. Lutningen lagras i R33 och med interceptet i fönstret startas programkörningen i steg 498 (se flödesdiagrammet). Det kan t ex användas för att testa gamla mätningar, eller mätningar som gjorts av andra författare.

FLÖDESDIAGRAM 3



KONSISTENSTEST

Med de utgångsstorheter som nyss fåtts genom denna subrutin, kan mätningen testas för konsistens.

Den framräknade barriären matas in via B'. Programmet körs sedan som vanligt, kapacitansen per ytenhet (C/A i R28) multipliceras med arean (R32) för att få kapacitansen C, som sedan inverteras och kvadreras, varpå värdet plottas bredvid de experimentella värdena...

Ibland får man riktiga överraskningar!

ALLMÄNT

1. En mycket ofta använd storhet är termen kT/q som lagras direkt i R21 så snart temperaturen angetts. Den jämte alla andra storheter kan sedan användas i fortsatta beräkningar.

2. Konstanterna har hämtats ur ref 2. Alla värden skall ges i SI-systemet utom de som ges i samband med resistiviteten eller dopningskoncentrationen, som nämnts i texten.

3. Tangenterna A och A' används här för att omvandla mellan vissa optiska storheter såsom cm^{-1}, eV och μm. Dessa steg kan användas till något annat eller lämnas tomma för att programmera in andra beräkningar. Författaren arbetar med foto-elektriska mätningar varför det visat sig mycket användbart att ha med dessa mycket ofta använda omvandlingssubrutiner i anslutning till detta program.

4. Programmet har "översatts" till en Apple II-variant (SEMICON-9). Det är menystyrt och har en del andra finesser som man givetvis inte kan ha i en fickkalkylator. Intresserade att få ta del av detta BASIC program höra av sig till Programbiten.

5. Det finns en utökning av programmet som beräknar alla "möjliga" strömmar i samband med sådana dioder (generations-, mättnads-, strömmen etc). Detta har dock inte medtagits här.

REFERENSER

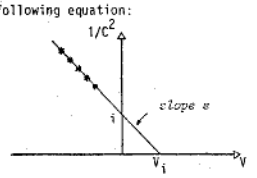
1. S.M. Sze, "Physics of semiconductor devices", Wiley-interscience, New York.

2. Rev. Mod. Phys. 41 (1969).

APPENDIX

The programme is based upon the following equation:

$$C = A\sqrt{\frac{qN_x\varepsilon\varepsilon_o}{2(V_i - V - kT/q)}}$$



and hence:

$$\frac{1}{C^2} = \left(\frac{-2}{A^2 q\varepsilon\varepsilon_o N_x}\right) V + \left(\frac{2}{A^2 q\varepsilon\varepsilon_o N_x}\right)(V_i - kT/q)$$

slope s - slope s

$$N_x = C\,\frac{1}{sd^4} \qquad N_x = \frac{-1.956E31}{sd^4} \quad \textit{for silicon}$$

★ ★ ★

Quick formulae (cm, Ωcm, V)

$w \approx 0.4\sqrt{\rho V}$ [µm]

$C \approx \frac{20d^2}{\sqrt{\rho V}}$ [nF] *slight oversetimate*

R05	används	R21	kT/q
R06	används	R22	$(2\pi km_o T/h^2)^{3/2}$
R07	m_o	R23	
R08	h	R24	w
R09	k	R25	E_{max}
R10	q	R26	Δφ
R11	$\varepsilon_o\varepsilon_r$	R27	$V_{tot}=V_{bi}-V-kT/q$
R12	E_i eller E_F	R28	C/A
R13	n_i^2/N_d eller n_i^2/N_a	R29	V
R14	N_d eller N_a	R30	$q/2\varepsilon_o\varepsilon_r$
R15	n_i	R31	V_{br}
R16	N_v	R32	A
R17	N_c	R33	lutning s (CV-mätning)
R18	E_g	R34	intercept V_i (CV-mätning)
R19	T	R35	används
R20	ϕ_B	R36	används
		R37	används

For the sake of future comparisons, here are the standard values of the physical constants, as well as two examples of MS-junctions, one on n- and the other on p-type SILICON. Most decimals are IRRELEVANT, but they are given here for the purpose of debugging programs.

Physical constants taken from Rev. Mod. Phys. 41 (1969).

m_o = 9.10956 E-31 kg
h = 6.62620 E-34 Js
k = 1.38062 E-23 JK^{-1}
q = 1.60219 E-19 C
$\varepsilon\varepsilon_o$ = 1.03594 E-10 Fm^{-1} ($=4\pi qm_o k^2/h^3$)
A* = 1.20164 E6 $Am^{-2}K^{-2}$
ε_o = 8.85418782E-14 Fcm^{-1}
ε_r = 11.7 (for silicon)

For T = 293 K kT/q = 25.2480 meV

N_c = 2.71979 E25 m^{-3}
N_v = 1.01129 E25 m^{-3}
E_g = 1.110 eV
n_i = 4.71226 E15 m^{-3}
E_i = 0.54250 eV

Parameter	n-silicon	p-silicon	units
Resistivity	10		Ωcm
Dop. N	4.62330	13.0030	E20 m^{-3}
n_i^2/N	4.80292	1.70771	E10 m^{-3}
E_F (Fermi)	0.83270	0.22620	eV
V(Breakdown)	518	279	V
Barrier ϕ_{B_o}	0.79000		eV
V_{bi}	0.51272	0.56380	V
V_{bias}	-kT/q		V
E_{max}	8.56287	1.50588	E5 Vm^{-1}
w	1.19753	0.74880	µm
Δφ	10.2658	13.6138	meV
C/A	0.86506	1.38346	E-4 Fm^{-2}
$qn_iw/2\tau$ (τ=1s)	4.52100	2.82692	E-10 Am^{-2}
Barrier ϕ_B	0.77973	0.77639	eV
J_o (A**, ϕ_B)	3.72101	1.21390	E-3 A
J (A**, ϕ_B)	2.35213	0.76733	E-3 A
J_o (A*, ϕ_B)	3.99227	4.55834	E-3
J (A , ϕ_B)	2.52359	2.88142	E-3

According to J.M. Andrews and M.P. Lepselter, Solid-State Electronics, Vol. 13, 1970, practical values for the modified Richardson constant are: 1.12 E6 for electrons and 0.32 E6 for holes in silicon.

Minnesinnehåll efter initiering med **RST SBR π**

RST SBR π
ger nedan utskrift

	0. 00 00
	0. 00 01
	0. 00 02
	0. 00 03
	0. 00 04
	0. 00 05
	0. 00 06
	9.10956-31 07
	6.6262-34 08
	1.38062-23 09
1.60219-19	1.60219-19 10
1.03594-10	1.03594-10 11
9.10956-31	5.4250244-01 12
1.38062-23	0. 00 13
6.62620-34	0. 00 14
	4.7122563 15 15
	1.0112875 25 16
2.93000 02	2.7197948 25 17
1.10998 00	1.1099836 00 18
2.71979 25	2.93 02 19
1.01129 25	0. 00 20
4.71226 15	2.5248045-02 21
5.42502-01	2.4219488 25 22
	0. 00 23
	0. 00 24
	0. 00 25
	0. 00 26
	0. 00 27
	0. 00 28
	0. 00 29
	7.7330251-10 30
	0. 00 31
	0. 00 32
	0. 00 33
	0. 00 34
	0. 00 35
	0. 00 36
	0. 00 37
	0. 00 38
	0. 00 39

PROGRAMLISTNING NÄSTA SIDA

HALVLEDARFYSIK TI59

```
HALVLEDARFYSIK

004  13  C
133  18  C'
180  97  DSZ
206  98  ADV
216  69  DP
240  19  D'
251  14  D
265  89  π
349  17  B'
354  12  B
373  68  NOP
419  57  ENG
428  78  Σ+
447  11  A
461  16  A'
474  10  E'
533  48  EXC
552  79  Σ
584  38  SIN
614  37  P/R
625  15  E

HALVLEDARFYSIK

000  02  2
001  09  9
002  03  3
003  76  LBL
004  13  C
005  98  ADV
006  99  PRT
007  42  STO
008  19  19
009  33  X²
010  65  ×
011  07  7
012  93  .
013  00  0
014  02  2
015  52  EE
016  94  +/-
017  04  4
018  55  ÷
019  53  (
020  43  RCL
021  19  19
022  85  +
023  01  1
024  01  1
025  00  0
026  08  8
027  95  =
028  94  +/-
029  85  +
030  01  1
031  93  .
032  01  1
033  05  5
034  03  3
035  95  =
036  42  STO
037  18  18
038  99  PRT
039  02  2
040  65  ×
041  89  π
042  65  ×
043  43  RCL
044  09  09
045  65  ×
046  43  RCL
047  07  07
048  65  ×
049  43  RCL
050  19  19
051  55  ÷
052  43  RCL
053  08  08
054  33  X²
055  95  =
056  45  Y^x
057  01  1
058  93  .
059  05  5
060  65  ×
061  02  2
062  95  =
063  42  STO
064  22  22
065  65  ×
066  06  6
067  65  ×
068  93  .
069  03  3
070  02  2
071  07  7
072  02  2
073  45  Y^x
074  01  1
075  93  .
076  05  5
077  95  =
078  42  STO
079  17  17
080  99  PRT
081  43  RCL
082  22  22
083  65  ×
084  93  .
085  05  5
086  05  5
087  08  8
088  06  6
089  05  5
090  45  Y^x
091  01  1
092  93  .
093  05  5
094  95  =
095  42  STO
096  16  16
097  99  PRT
098  65  ×
099  43  RCL
100  17  17
101  95  =
102  34  ΓX
103  65  ×
104  53  (
105  43  RCL
106  18  18
107  55  ÷
108  02  2
109  55  ÷
110  53  (
111  43  RCL
112  09  09
113  65  ×
114  43  RCL
115  19  19
116  55  ÷
117  43  RCL
118  10  10
119  54  )
120  42  STO
121  21  21
122  54  )
123  94  +/-
124  22  INV
125  23  LNX
126  95  =
127  42  STO
128  15  15
129  99  PRT
130  61  GTO
131  69  DP
132  76  LBL
133  18  C'
134  98  ADV
135  42  STO
136  14  14
137  45  Y^x
138  93  .
139  05  5
140  09  9
141  08  8
142  94  +/-
143  65  ×
144  01  1
145  93  .
146  01  1
147  08  8
148  52  EE
149  01  1
150  05  5
151  95  =
152  99  PRT
153  42  STO
154  31  31
155  87  IFF
156  01  01
157  97  DSZ
158  43  RCL
159  14  14
160  55  ÷
161  01  1
162  52  EE
163  06  6
164  95  =
165  71  SBR
166  14  D
167  43  RCL
168  14  14
169  35  1/X
170  65  ×
171  43  RCL
172  15  15
173  33  X²
174  95  =
175  42  STO
176  13  13
177  61  GTO
178  98  ADV
179  76  LBL
180  97  DSZ
181  43  RCL
182  14  14
183  55  ÷
184  01  1
185  52  EE
186  06  6
187  95  =
188  71  SBR
189  19  D'
190  43  RCL
191  14  14
192  42  STO
193  13  13
194  35  1/X
195  65  ×
196  43  RCL
197  15  15
198  33  X²
199  95  =
200  42  STO
201  14  14
202  71  SBR
203  98  ADV
204  92  RTN
205  76  LBL
206  98  ADV
207  43  RCL
208  14  14
209  99  PRT
210  55  ÷
211  43  RCL
212  13  13
213  99  PRT
214  65  ×
215  76  LBL
216  69  DP
217  43  RCL
218  16  16
219  55  ÷
220  43  RCL
221  17  17
222  95  =
223  23  LNX
224  65  ×
225  43  RCL
226  21  21
227  55  ÷
228  02  2
229  85  +
230  43  RCL
231  18  18
232  55  ÷
233  02  2
234  95  =
235  42  STO
236  12  12
237  99  PRT
238  92  RTN
239  76  LBL
240  19  D'
241  86  STF
242  01  01
243  55  ÷
244  02  2
245  93  .
246  08  8
247  01  1
248  02  2
249  05  5
250  76  LBL
251  14  D
252  65  ×
253  43  RCL
254  10  10
255  65  ×
256  01  1
257  03  3
258  05  5
259  00  0
260  95  =
261  35  1/X
262  99  PRT
263  92  RTN
264  76  LBL
265  89  π
266  98  ADV
267  47  CMS
268  25  CLR
269  58  FIX
270  05  05
271  01  1
272  93  .
273  06  6
274  00  0
275  02  2
276  01  1
277  09  9
278  52  EE
279  94  +/-
280  01  1
281  09  9
282  42  STO
283  10  10
284  99  PRT
285  55  ÷
286  01  1
287  93  .
288  00  0
289  03  3
290  05  5
291  09  9
292  04  4
293  52  EE
294  94  +/-
295  01  1
296  00  0
297  42  STO
298  11  11
299  99  PRT
300  55  ÷
301  02  2
302  95  =
303  42  STO
304  30  30
305  09  9
306  93  .
307  01  1
308  00  0
309  09  9
310  05  5
311  06  6
312  52  EE
313  94  +/-
314  03  3
315  01  1
316  42  STO
317  07  07
318  99  PRT
319  01  1
320  93  .
321  03  3
322  08  8
323  00  0
324  06  6
325  02  2
326  52  EE
327  94  +/-
328  02  2
329  03  3
330  42  STO
331  09  09
332  99  PRT
333  06  6
334  93  .
335  06  6
336  02  2
337  06  6
338  02  2
339  00  0
340  52  EE
341  94  +/-
342  03  3
343  04  4
344  42  STO
345  08  08
346  99  PRT
347  81  RST
348  76  LBL
349  17  B'
350  42  STO
351  20  20
352  91  R/S
353  76  LBL
354  12  B
355  42  STO
356  29  29
357  43  RCL
358  20  20
359  98  ADV
360  99  PRT
361  87  IFF
362  01  01
363  57  ENG
364  75  -
365  43  RCL
366  18  18
367  85  +
368  71  SBR
369  78  Σ+
370  43  RCL
371  14  14
372  76  LBL
373  68  NOP
374  65  ×
375  43  RCL
376  30  30
377  55  ÷
378  43  RCL
379  27  27
380  95  =
381  35  1/X
382  34  ΓX
383  42  STO
384  24  24
385  99  PRT
386  35  1/X
387  65  ×
388  02  2
389  65  ×
390  43  RCL
391  27  27
392  95  =
393  42  STO
394  25  25
395  99  PRT
396  65  ×
397  43  RCL
398  30  30
399  55  ÷
400  02  2
401  55  ÷
402  89  π
403  95  =
404  34  ΓX
405  42  STO
406  26  26
407  99  PRT
408  43  RCL
409  11  11
410  55  ÷
411  43  RCL
412  24  24
413  95  =
414  99  PRT
415  42  STO
416  28  28
417  91  R/S
418  76  LBL
419  57  ENG
420  75  -
421  71  SBR
422  78  Σ+
423  43  RCL
424  13  13
425  61  GTO
426  68  NOP
427  76  LBL
428  78  Σ+
429  43  RCL
430  12  12
431  95  =
432  42  STO
433  23  23
434  99  PRT
435  75  -
436  43  RCL
437  21  21
438  75  -
439  43  RCL
440  29  29
441  99  PRT
442  95  =
443  42  STO
444  27  27
445  92  RTN
446  76  LBL
447  11  A
448  55  ÷
449  01  1
450  93  .
451  02  2
452  03  3
453  09  9
454  08  8
455  01  1
456  04  4
457  52  EE
458  94  +/-
459  04  4
460  76  LBL
461  16  A'
462  55  ÷
463  01  1
464  52  EE
465  04  4
466  95  =
467  35  1/X
468  22  INV
469  52  EE
470  58  FIX
471  03  03
472  91  R/S
473  76  LBL
474  10  E'
475  98  ADV
476  22  INV
477  57  ENG
478  52  EE
479  58  FIX
480  05  05
481  43  RCL
482  32  32
483  99  PRT
484  69  DP
485  12  12
486  99  PRT
487  32  X:T
488  99  PRT
489  66  PAU
490  42  STO
491  33  33
492  69  DP
493  13  13
494  99  PRT
495  00  0
496  69  DP
497  15  15
498  42  STO
499  34  34
500  66  PAU
501  99  PRT
502  43  RCL
503  32  32
504  33  X²
505  65  ×
506  43  RCL
507  10  10
508  65  ×
509  43  RCL
510  11  11
511  65  ×
512  43  RCL
513  33  33
514  55  ÷
515  02  2
516  95  =
517  94  +/-
518  35  1/X
519  66  PAU
520  66  PAU
521  99  PRT
522  18  C'
523  66  PAU
524  66  PAU
525  87  IFF
526  01  01
527  48  EXC
528  94  +/-
529  85  +
530  43  RCL
531  18  18
532  76  LBL
533  48  EXC
534  85  +
535  98  ADV
536  66  PAU
537  99  PRT
538  43  RCL
539  34  34
540  85  +
541  43  RCL
542  21  21
543  99  PRT
544  95  =
545  99  PRT
546  98  ADV
547  98  ADV
548  98  ADV
549  98  ADV
550  91  R/S
551  76  LBL
552  79  Σ
553  98  ADV
554  43  RCL
555  36  36
556  99  PRT
557  55  ÷
558  43  RCL
559  21  21
560  94  +/-
561  95  =
562  22  INV
563  23  LNX
564  65  ×
565  43  RCL
566  13  13
567  42  STO
568  05  05
569  87  IFF
570  01  01
571  38  SIN
572  43  RCL
573  17  17
574  99  PRT
575  65  ×
576  43  RCL
577  16  16
578  55  ÷
579  43  RCL
580  14  14
581  42  STO
582  05  05
583  76  LBL
584  38  SIN
585  43  RCL
586  16  16
587  99  PRT
588  95  =
589  42  STO
590  37  37
591  65  ×
592  04  4
593  65  ×
594  43  RCL
595  05  05
596  99  PRT
597  85  +
598  43  RCL
599  37  37
600  33  X²
601  95  =
602  34  ΓX
603  75  -
604  43  RCL
605  37  37
606  95  =
607  55  ÷
608  02  2
609  95  =
610  99  PRT
611  18  C'
612  92  RTN
613  76  LBL
614  37  P/R
615  33  X²
616  65  ×
617  89  π
618  55  ÷
619  04  4
620  95  =
621  42  STD
622  32  32
623  92  RTN
624  76  LBL
625  15  E
626  52  EE
627  94  +/-
628  01  1
629  02  2
630  57  ENG
631  58  FIX
632  02  02
633  99  PRT
634  33  X²
635  35  1/X
636  78  Σ+
637  91  R/S
638  00  0
639  00  0
```

RÄTTELSE TILL PROGRAMBITEN 83-4

»KRYPTAN«

Från Conny Bonnet har vi fått följande brev om en rättelse till programmet "Kryptan":

"Jag upptäckte i går att det fanns en liten miss i mitt program Kryptan. Missen består i att om du stöter på ett monster, som anfaller först (7-12) och du lyckas fly, så kommer nästa gång du träffar ett monster, som du får anfalla först att i stället anfalla dig först. Följande ändring skall göras för att motverka detta."

Ändringen medför att steg 433 - 443 i stället ska se ut på följande sätt:

```
433  77  GE
434  04  04
435  41  41
436  01  1
437  02  2
438  77  GE
439  04  04
440  42  42
441  22  INV
442  86  STF
443  02  02
```

Å medans jag står och monterar sidnummer går jag till skrivmaskinen å författar dessa rader.
Ni som fått vänta på beställt material.
Tidningen, medlemsregistret, mycket jobb på jobbet mm mm är boven.
Skäll inte..
Skriv uppmuntrande i stället, som t ex 'vi väntar gärna bara vi får pryttlarna'.
Jag lovar 'Vilket år som helst skickar jag det beställda.

Skämt å sido, men misströsta inte.

claes schibler
Materialförvaltare
+ allt-i-allo

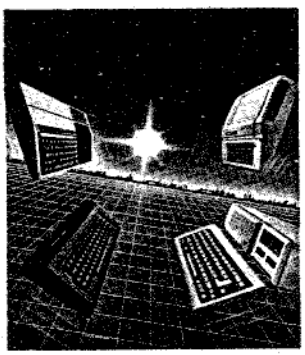
1984-05-17 kl 02:24
å innehållsförteckningen kvar...

HOME COMPUTER™ magazine

99'er Magazine.....HCM

Vad har hänt med 99'er Magazine? Varför kom det inget nummer i december? Är tidningen nedlagd? Dessa frågor ställer nog många 99-ägare sig. Jag kan ge lugnande besked på denna punkt. Tidningen lever vidare dock i något ändrad skepnad. Namnet på tidningen är numer Home Computer Magazine el. HCM. HCM kommer i fortsättningen att skriva om följande fyra fabrikat:

TEXAS INSTRUMENTS	(TI-99/4A)
COMMODORE	(VIC-20 o VIC-64)
IBM	(PCjr)
APPLE	(APPLE II)



Tidningen har blivit tjockare (195 sidor Vol 4 No 1). 99-materialet är dock i stort sett oförändrat till mängd och innehåll.
Vad innehåller ett nummer av HCM?

"ON SCREEN"	Chefredaktörens syn på saker och ting.
"GROUP GRAPEVINE"	Användarklubbarnas sidor.
"FEATURES"	Div. artiklar och program om tex. ordbehandling, musik, grafik, hårdvara, assembler, ekonomi etc.
"LOGO TIMES"	En tidning i tidningen om programspråket LOGO.
"PRODUKT REWIEVS"	En stor avdelning med recensioner av programvara och hårdvara.
"GAMEWARE BUFFET"	2-4 spelprogram med listningar. Både TI-basic och Extended Basic finns representerat.
"LETTERS TO THE EDITOR"	Frågor och svar samt korta program och subrutiner.
"HC TECH NOTES"	Om programmeringsteknik.
"HOME COMPUTER DIGEST"	En inbunden bilaga om nyheter på hemdatormarknaden.
"DEBUGS ON DISPLAY"	Avdelningen för fel och förbättringar.

SAMT en mängd annonser ur vilka jag saxat följande för er som går och misströstar och inte får tag på tillbehör:

32k RAM till Utbyggnadsenheten	$119.95
RS 232 till -"-	$79.95
Disk-kontrollkort till -"-	(kommer snart)
TI-forth	$38.95

Säljs av:
TEX-COMP Inc, P.O.Box 33084, Granada Hills, CA 91344.

Expansionssystem med 32k RAM, Diskkontrollkort, RS 232, Speech-synt. och diskdrive.

1 Drive	$796.00
2 Drivar	$995.00

Det här expansionssystemet är inte det samma som Texas eget.

Säljs av TEX MICRO , P:O:Box 5366, Titusville, FL

SST BASIC COMPILER
Översätter ditt BASIC-program till maskinkod samtidigt som du kan editera och avlusa med hjälp av TI Basic.
Det behövs Mini Memory och kassettbandspelare.

Kostnad: US$ 50.00
Säljs av **SST Software, Inc.**
P.O. Box 26
Cedarburg, WI 53012
(414) 771-8415
USA

(OBS! Jag kan inte garantera kvalité eller priser. Skriv och fråga!)
Om du har importerat något från U.S.A så skriv gärna till föreningen och berätta vad det kostade och hur det fungerar.

TILL SLUT...
Ni är inte ensamma om att äga en 99'a. Över 2 miljone datorer är sålda över hela världen. Flera USA-företag kommer att fortsätta sälja tillbehör till 99:an, och TI har sålt tillverkningsrätten av tillbehör och programvara from 31 mars till ett U.S.A-företag.

Adressen till HCM är följande:

HOME COMPUTER™ magazine
P.O. Box 5537
Eugene, OR 97405
USA

Ett års prenumeration by 'Surface mail'
är US$ 43.00
leveranstid ungefär 3 månader.

Bengt Fahlgren

On Screen

ASSEMBLER-SKOLAN

Det här är den första delen av assembler-skolan. Vi kommer att presentera assemblerspråket för TMS 9900 och visa hur det används. Intressanta och nyttiga rutiner skrivna i assembler kommer också att medfölja varje del, dock ej denna första del. Inte bara själva språket (mjukvaran) utan också maskinen (hårdvaran) kommer att redogöras och diskuteras, t.ex TMS 9929A eller VDP (video-processorn) och TMS 9919 (ljud-processorn) o.s.v.

Eftersom det första steget är det svåraste, skall vi hjälpa er lite på traven genom att visa de mest grundläggande när det gäller lågnivå hos datorer.

TALSYSTEM

De flesta av läsarna har säkert redan kommit i kontakt med det binära talsystemet, och om ni känner att ni behärskar det till fullo kan ni hoppa över detta avsnitt.

Det binära talsystemet

Binär står för TVÅ vilket innebär att det binära talsystemet endast har två siffror, nämligen 0 och 1. Det decimala talsystemet som vi vanligen räknar med har tio siffror eftersom deci betyder TIO. Dvs det innefattar siffrorna 0 till 9. När man skall beräkna vad ett tal i ett speciellt talsystem blir i ett annat så är siffrans position viktig, detta gäller alla talsystem, även det decimala. Basen upphöjt till siffrans position (med start från noll) gånger siffran självt är talet representerat i decimalt. (OBS man numrerar siffrorna bakifrån)

Exempel

Binärt	1	0	1	0	1
	*	*	*	*	*
Basen	2^4	2^3	2^2	2^1	2^0

Decimalt 16+ 0 + 4 + 0 + 1=21

Varför använder man sig av basen två, m.a.o det binära talsystemet? Svaret är att eftersom en dator är en maskin och drivs på spänning, så har man låtit spänning (ibland +5 volt) motsvara en etta och ingen spänning en nolla. Det medför att det binara talsystemet direkt kan tillämpas på en dator.

Det hexadecimala talsystemet

Hexadecimal står för SEXTON. Läsaren kanske nu tänker, som sig bör: Hur får man ihop sexton siffror när vi bara har tio? Svar: Man använder sig av bokstäverna A-F, där A är lika med tio o.s.v.

För att räkna ut vad ett tal i det hexadecimala talsystemet blir decimalt gör man på samma sätt som i fallet ovan, men med skillnaden att man byter ut 2:an mot 16.

Exempel

Hexadecimalt	2	A	4	F	
	*	*	*	*	
Basen	16^3	16^2	16^1	16^0	

Decimalt 8192+2560+64+15 =10831

Det Hexadecimala talsystemet används ofta inom programmering, främst därför att det är lättare att omvandla från det binära talsystemet till det hexadecimala talsystem än till det decimala talsystemet. Jag tror att de flesta som programmerat BASIC (på 99:an) känner igen nästa exempel, det används för att definiera tecken med CALL CHAR.

Exempel

Binär	0010	1010	0100	1111
Hexadecimal	2	A	4	F

I detta fall är det bara att att binda ihop de fyra siffrorna i den binära koden (brukar kallas en "nybble"), och omvandla dessa till det hexadecimala talsystemet. (Du kan ta tabell 1 till hjälp) Sedan sätter man ihop de hexadecimala siffrorna och man har talet. Exemplet visar också varför man inte använder det binära talsystemet särskilt ofta. Det binära talet består av fyra gånger så många siffror som det hexadecimala.

I slutet av denna artikel hittar du en nyttig rutin skriven i BASIC, för omvandling mellan valfria baser, som kan vara till hjälp.

MINNET

Minnet består av bitar som är organiserade i grupper. En bit kan man likna vid en kontakt som antingen släpper igenom ström (etta) eller är avstängd (nolla). Dessa grupper kan vara av varierande storlek. Vanligtvis brukar en grupp av åtta bitar kallas en "byte". Två bytes eller 16 bitar brukar i mikrodatorsammanhang kallas ett ord. (På större datorer kan ett ord bestå av t.ex. 32 bitar.)

I ett ord kallar man byten som ligger till vänster eller har högst värde för den mest signifikanta byten och den andra för den minst signifikanta byten. Man får ej blanda ihop detta med MSB och LSB, som står för den mest signigikanta biten resp. den minst signifikanta biten. MSB brukar refereras till som tecken bit (eng. sign bit). MSB har till uppgift att representera negativa tal, alltså när MSB är satt (lika med en etta) så är talet negativt. Detta brukar kallas två-komplement (eng. two's complement).

Exempel

	MSB			LSB
Binärt	1001	1010	1100	0010
Hexadecimalt	9	A	C	2
	*	*	*	*
Basen	16^3	16^2	16^1	16^0

Decimalt 36864+2560+ 192+ 2 =39618

Om man tittar på MSB så ser man att den är satt, alltså har talet också ett motsvarande negativt tal. Det negativa talet får man fram genom att dra i från 65536 (decimalt), alltså 39618-65536=-25918 . Att se om tal är representerat av ett negativt tal (för sexton bitar) eller inte kan man gör främst genom att se om teckenbiten är satt eller ej. Men om du har talet i hex eller i decimal kan du lätt se det om talet är större än 32767 (decimalt) resp. 7FFF (hex).

Minnets uppläggning hos TI 99/4(A) .

```
>0000 ------------------------------
      I    ROM inbyggt i datorn     I
      I            8K               I
>2000 I-------------------------------I
      I      Minnesexpansion         I
      I      (lägre delen) 8K        I
>4000 I-------------------------------I
      I Area för DSR (device service  I
      I routine ) yttre enheters ROM  I
      I t.ex. RS232, disk-kontroller  I
      I            8K                 I
>6000 I-------------------------------I
      I    ROM eller RAM i modul      I
      I            8K                 I
>8000 I-------------------------------I
      I RAM (scratch pad) eller       I
      I kladdblocksminnet 256 bytes   I
>8400 I-------------------------------I
      I Addresser för minneskartade   I
      I enheter: VDP,GROM,LJUD        I
      I och TAL.                      I
>A000 I-------------------------------I
      I      Minnesexpansions         I
      I (högre delen) 24K             I
      I                               I
>FFFF ------------------------------
```

Varje byte i datorn har en adress från 0 upp till)FFFF (tecknet ")" betyder att talet är hexadecimalt). Med hjälp av adressen kan man hitta den information man vill komma åt. Eftersom 9900 är en 16-bitar processor kan den hantera ord bestående av 16 bitar eller 2 bytes. Den första byten måste alltid ligga på en jämn adress. Den är alltid mest signifikant (har högst värde). Den minst signifikanta byten i ordet ligger på nästa adress (som alltså är udda).

I regel anger man minnesstorleken i K bytes som uttalas "kilo bytes", men det betyder inte 1000 bytes utan 1024 bytes (kilo i vanlig bemärkelse skrivs med litet k). 1024 eller)400 är ett jämnt och bra tal hexadecimalt sett, och är dessutom mycket nära 1000, varför man alltså låtit 1024 motsvara ett K bytes.

Minnet kan antingen var ett ROM (eng. read only memory) eller RAM (eng. random access memory). Ett ROM minne kan man endast läsa och i ett sådant lagrar man program som skall vara permanenta (bestående). Ett RAM (eller bättre RWM = read and write memory), kan däremot både läsas från och skrivas till, men så fort strömmen slås av går innehållet förlorat. ROM används för program som alltid måste finnas i datorn, t.ex. BASIC-tolken i 99:an. BASIC-program lagras däremot i RAM eftersom man själv vill kunna ändra i dem.

Den första minnesarean i 99:ans minne är ett ROM och innehåller bl.a. GPL-tolken (GPL är assemblerliknade som TI har uppfunnit och är TI:s favoritspråk. GPL-program måste ligga i ett special-ROM (GROM) som TI har patent på. TI BASIC är skriven i detta språk), matematiska- och omvandlingsrutiner.

Den andra delen är en expansionsdel kan endast användas om man har ett minneskort, fristående eller i expansionslådan.

Del tre används som area för ROM som ligger inbyggda i yttre enhter som RS232, diskkontrollkortet och är styrprogram för dessa.

Del fyra är adresserad till modulporten. Den här delen används av vissa moduler som har ett ROM på den adressen. De flesta modulprogrammen är dock skrivna i GPL och innehåller endast GROM.

Kladdblocksminnet är ett RAM bestående av 256 bytes och används internt av ROM och GROM för olika uppgifter.

Del sex används bl.a för att skriva till och läsa från VDP och GROM. Hur detta görs kommer att diskuteras i kommande delar av assembler-skolan.

Den sista delen är som den andra delen.

En intressant iakttagelse är att 99:an i grundutförande saknar egentligt RAM (utom kladdblocksminnets 256 bytes). I stället använder den RAM från VDP som endast tillsammans med lämpligt GROM kan brukas som programlagringsarea.

PROCESSORN TMS 9900

Processorn är datorns hjärna, som styr själva arbetet. Processorns eget språk kallas maskinspråk (även maskinkod). I processorn har man tre stycken sextonbitars hårdvaruregister, som har olika uppgifter och är av stor betydelse för att förstå en dators arbete och i programmerings-hänseende. Ett register är ungefär det samma som en minnescell (ord), men det har speciella upgifter att fylla för processorns egna arbete.

De tre hårdvaruregistrena är :
1. programräknaren (PC)
2. mjukvaruregisterpekaren (WP)
3. statusregistret (ST)

PC

Programräknaren pekar på den adress där maskinkoden finns som processorn skall tolka och utföra. Efter en instruktion har blivit utförd ökas automatiskt PC och pekar på nästa adress o.s.v.

WP

Mjukvaruregisterpekaren pekar på en area i minnet där register som programmeraren kan använda ligger. Det finns sexton sådana register och används till att mellanlagra värden under beräkningar o.dyl.

ST

I Status registret sätts olika bitar som har olika funktioner beroende på vilken räkneoperation som just utförts. Bitarna benämnes med olika namn (se nedan).

Not: två komp=två komplement.

Namn	bit	
L)	0	logiskt större än (ej två komp)
A)	1	aritmetiskt större än (i två komp)
EQ	2	lika med
C	3	minnessiffra
OV	4	överspill
OP	5	udda paritet
X	6	utökad operation
-	7-11	reseverade
Int-mask	12-15	interrupt bitar

Bitarnas praktiska användning kommer att förklaras i programexempel för att enklare förstå dem.

ASSEMBLER SPRÅKET

Att programmera i ren maskinkod är inte den roligaste sysselsättning man kan ha, då det skulle innebära att programmeraren skulle få sitta med en tabell över maskininstruktionerna och "knacka hex". Alltså gav man varje maskinkodsinstruktion ett namn som påminner om vad instruktionen gör. JMP betyder t.ex. JUMP (hoppa) och MOV betyder MOVE (flytta) o.s.v. När man har skrivt in desso instruktionsnamn i t.ex en editor, måste man köra ett program som omvandlar dessa till maskinkod. Programmet kallas assemblator.

I assemblerprogrammering går det mest ut på att flytta, ladda, addera och subbtrahera värden. Det kan till en början förefalla rent meningslöst, men när man ser dem i sitt sammanhang faller bitarna på plats i pusslet.

```
100 REM BAS KONVERTRERING
110 CALL CLEAR
120 INPUT "VILKEN BAS?":A
130 INPUT "  TILL BAS?":B
140 INPUT "TAL-)":C$
150 FOR I=LEN(C$)TO 1 STEP -1
160 D$=SEG$(C$,I,1)
170 IF ASC(D$)>57 THEN 200
180 E=VAL(D$)
190 GOTO 210
200 E=ASC(D$)-55
210 F=F+(E*A^(ABS(I-LEN(C$))))
220 NEXT I
230 FOR J=INT(LOG(F+0.5)/LOG(B))TO 0 STEP -1
240 G=INT(F/B^J)
250 F=F-G*B^J
260 IF G>9 THEN 290
270 H$=H$&STR$(G)
280 GOTO 300
290 H$=H$&CHR$(G+55)
300 NEXT J
310 PRINT "SVAR:";H$
320 H$=""
330 GOTO 120
```

Tabell 1

decimalt	binärt	hexadecimalt
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Snabbare GRAFIK i TI-Basic ?

Av Micael Dahlquist.

De flesta som har programmerat i TI BASIC har väl tyckt att TI-99:an inte är marknadens snabbaste maskin.Men man kan (på en del ställen) snabba upp BASIC-programmen.Det hela bygger på att det ska se ut som om datorn gör en massa saker, medans den i själva verket nästan inte gör någonting. Genom att använda följande så snabbas programmen upp:

1. DOLD GRAFIK
2. TECKEN-DEFINITIONER

Jag börjar med att försöka förklara begreppet DOLD GRAFIK; när grafiken ritas upp på skärmen låter man den vara 'osynlig', sedan när man får lust att visa den så ändrar man bara färgen på den genom några CALL COLOR satser (det hela kommer då fram till synes väldigt snabbt, fastän det legat där redan innan man ändrat färgerna).

Exempel 1

```
100 CALL SCREEN(2)
110 CALL CLEAR
120 FOR I=1 TO 11
130 READ A,B,C
140 CALL HCHAR(A,B,96,C)
150 NEXT I
160 CALL COLOR(9,5,5)
170 GOTO 170
180 DATA 10,19,2,11,16,1,11,19,2,12,16,8,13,
    18,4,13,23,1,14,18,4,15,18,4,16,16,3,16,
    21,1,17,21,1
```

Kommentarer till programmet:

100 svart skärmfärg, 110 rensar skärmen, 120-150 loop som ritar vår demonstrationsfigur med tecknet 96, 160 visa figuren, genom att ändra färggrupp 9 (som teknet 96 tillhör), 170 stanna (inte stoppa) programmet, 180-200 data för figuren.

Exempel 2

```
100 CALL SCREEN(2)
110 CALL CLEAR
120 FOR I=14 TO 30
130 LET F=F+1+(F=8)*8
140 CALL HCHAR(4,I,F*8+88)
150 CALL HCHAR(20,44-I,F*8+88)
160 NEXT I
170 FOR I=5 TO 19
180 LET F=F+1+(F=8)*8
190 CALL HCHAR(24-I,14,F*8+88)
200 CALL HCHAR(I,30,F*8+88)
210 NEXT I
220 FOR I=4 TO 11
230 CALL VCHAR(I,I,(I-3)*8+88,2)
240 CALL VCHAR(23-I,I,(I-3)*8+88,2)
250 NEXT I
260 CALL HCHAR(12,12,152)
270 LET C=INT(RND*14+3)
280 FOR I=9 TO 16
290 CALL COLOR(I,C,C)
300 IF FLAGGA THEN 320
310 CALL COLOR(I,2,2)
320 NEXT I
330 CALL KEY(0,K,S)
340 IF S=0 THEN 270
350 LET C=INT(RND*14+3)
360 FOR I=16 TO 9 STEP -1
370 CALL COLOR(I,C,C)
380 IF FLAGGA THEN 400
390 CALL COLOR(I,2,2)
400 NEXT I
410 CALL KEY(0,K,S)
420 IF S=0 THEN 350
430 LET FLAGGA=(FLAGGA=0)
440 GOTO 270
```

Kommentarer till programmet:

100 svart skärmfärg, 110 rensar skärmen, 120-160 loop som sätter ut de två horizontala linjerna, 170-210 loop som sätter ut de två vertikala linjerna, 220-260 loop som sätter ut de två diagonala linjerna, 270-340 loop som ställer färggrupperna 9 till 16 med en och samma färg (om v liabeln FLAGGA innehåller värdet -1 så ställs sedan färggrupperna till svart på svart), 350-420 samma som förra men från färggrupp 16 till 9, 430 variabeln FLAGGA blir 0 om FLAGGA var -1 (och vice versa), 440 går tillbaka till rad 270

Genom att ändra teckendefinitionerna su kan man få fram vackra effekter och då samma tecken finns på många olika ställen på skärmen så tycker man att datorn gjort en massa saker s 5Qidigt (vilket den egentligen gör, men inte i BASIC-fart).

Exempel 3

```
100 CALL CLEAR
110 LET A$="FFFF000000000000"
120 CALL CHAR(32,A$)
130 LET A$=SEG$(A$,3,14)&SEG$(A$,1,2)
140 GOTO 120
```

Kommentarer till programmet:

100 rensar s-rmen, 110 låter A$ få en viss teckendefinition (testa gärna med andra definitioner, ex LET A$&"0102040810204080"), 120 definierar tecknet 32 (mellanslag) med teckenkoden i A$, 130 placerar om teckenordningen i A$ (så att de två första tecknena hamnar sist), 140 går tillbaka till 120.

Exempel 4

```
100 CALL CLEAR
110 LET  A$="00000000000000001038107C54542844
    FFFFFFFFFFFFFFFF"
120 FOR I=0 TO 23
130 CALL CHAR(I+128,"0")
140 CALL HCHAR(I+1,14,I+128,5)
150 CALL
SOUND(-100,930-I*30,I,940-I*30,I,950-I*30,I)
160 NEXT I
170 FOR I1=150 TO 128 STEP -1
180 FOR I2=1 TO 17 STEP 2
190 CALL CHAR(SEG$(A$,I2,16))
200 CALL CHAR(SEG$(A$,I2+16,16))
210 CALL SOUND(-1000,-4,0)
220 NEXT I2
230 NEXT I1
240 GOTO 120
```

Kommentarer till programmet:

100 rensar skärmen, 110 låter A$ få en viss teckendefinition, 120-160 loop som sätter ut tecknena 128 till 151 i en stapel som är fem tecken bred under vilda (?) ljudeffekter, 170-230 två loopar som tillsammans definierar om tecknena 128-151 samt knastrar ut ljud till TV:n, 240 går tillbaka till 120

Om man sedan kombinerar de här olika sätten så kan man kanske få fram riktigt roliga (sevärda) program. Skriv gärna om du kommer på något kort (men intressant) program.

Adressen är:

MICAEL DAHLQUIST
S:A SKOGSRUNDAN 5
184 00 ÅKERSBERGA

Och märk kuveret med texten: INTRESSANT

PROGRAM för kodning av SPRITES

av Seppo Huikuri

PROGRAM FÖR KODNING AV SPRITES.

Som de flesta vet, vilka har arbetat med Sprites. Är det ganska knöligt att planera Sprites, eftersom det på papperet planerade mönstret oftast inte ser ut som den färdiga Spriten på skärmen.

Därför har jag gjort ett enkelt program, med vars hjälp man
1) kan rita Sprites
2) får koden för Spriten
3) kan prova olika färger
4) kan exprimentera med MAGNIFY-effekten

Hela ritningsprocessen sker med hjälp av Joystick 1. Man kan både fylla och tömma en ruta genom att trycka på Joysticks-knappen. Om man trycker på knappen utanför rutorna, går programmet till kodningsprocessen samt visar Spriten i naturlig storlek. Därefter kan man välja önskad rutin genom tangentbordsval. Rutnätet på skärmen är delat i fyra delar 1, 3, 2, 4.
Kod 1 ger koden för figuren på fält 1, kod 2 för figur på fält 2 osv.
Fält 2, 3, och 4 används endast i samband med MAGNIFY-faktor 3 och 4.
Med rotationsrutinen kan man rotera mönstret på fält 1 med 90 grader moturs.
Programmet är skrivet i Extended Basic och kräver Joysticks.

```
100 REM ******************
110 REM *                *
120 REM * SPRITE-MAKER    *
130 REM * AV S.HUIKURI    *
140 REM *                *
150 REM ******************
160 REM
170 REM
180 DIM R2(64),S2(64)
190 CALL CLEAR :: CALL SCREEN(12)
200 CALL CHAR(91,"2800384444C444424003C424242423C
    ")
210 CALL CHAR(96,"FF818181818181FF"):: CALL COLOR(
    9,6,16)
220 CALL CHAR(104,"FFFFFFFFFFFFFFFF"):: CALL COLOR
    (10,2,1)
230 CALL CHAR(105,"0101010101010101808080808080808
    0000000000000FFFF")
240 CALL CHAR(112,"00003C3C3C3C"&RPT$("0",37)):: C
    ALL CHAR(120,"FFFFFFFFFFFFFFFF
    "):: CALL COLOR(11,10,16)
250 CALL CHAR(116,"00606000"&RPT$("00",25))
260 CALL SOUND(100,880,2)
270 DISPLAY AT(12,10):"ALPHA-LOCK!" :: FOR N=1 TO
    200 :: NEXT N :: CALL CLEAR
280 GOSUB 1050 :: GOSUB 1090 :: CH=112 :: F=2
290 REM --------------------
300 REM     ** JOYST **
310 REM --------------------
320 RP=49 :: SP=145 :: MF=1 :: CALL SPRITE(#1,CH,1
    0,RP,SP)
330 CALL JOYST(1,Y,X):: RP1=RP :: SP1=SP :: RP=RP-
    2*X :: SP=SP+2*Y
340 IF (RP)169)+(RP(9)+(SP(100)+(SP)249)THEN RP=RP
    I :: SP=SP1
350 R=INT((RP-1)/8+1):: S=INT((SP-1)/8+1):: CALL L
    OCATE(#1,RP,SP)
360 CALL GCHAR(R,S,X):: CALL KEY(1,KEY,ST):: IF KE
    Y()18 THEN 330
370 IF X=96 THEN CALL HCHAR(R,S,104):: GOTO 400
380 IF X=104 THEN CALL HCHAR(R,S,96):: GOTO 400
390 GOTO 440
400 FOR N=1 TO 20 :: NEXT N :: GOTO 330
410 REM --------------------
420 REM  ** BILDA KODER **
430 REM --------------------
440 CALL DELSPRITE(#1):: DISPLAY AT(24,1):"VÄNTA E
    N STUND!"
450 R=4 :: S=16
460 FOR Z=1 TO 4 :: E2=S+7 :: E1=S+3
470 FOR RR=R TO R+7 :: FOR SR=E2 TO S STEP -1
480 CALL GCHAR(RR,SR,X):: IF SR)E1 THEN 510
490 IF X=104 THEN L1=L1+2^(E1-SR)
500 GOTO 520
510 IF X=104 THEN L2=L2+2^(E2-SR)
520 NEXT SR
530 IF L1)9 THEN C1$=CHR$(L1+55)ELSE C1$=STR$(L1)
540 IF L2)9 THEN C2$=CHR$(L2+55)ELSE C2$=STR$(L2)
550 CODE$=CODE$&C1$&C2$ :: L1,L2=0
560 NEXT RR :: R=R+8 :: IF R)13 THEN R=4 :: S=24
570 KOD$(Z)=CODE$ :: CODE$=""
580 NEXT Z
590 REM --------------------
600 FOR N=1 TO 4 :: DISPLAY AT(20+N,1):"KOD";N;"=
    ";KOD$(N):: NEXT N
610 REM --------------------
620 CALL CHAR(128,KOD$(1)&KOD$(2)&KOD$(3)&KOD$(4))
    :: CALL SPRITE(#2,128,F,81,49)
630 GOSUB 1140 :: GOSUB 1150
640 REM --------------------
650 REM * VAELJ FUNKTION **
660 REM --------------------
670 CALL KEY(0,KE,ST)
680 IF (KE(49)+(KE)53)+(ST=0)THEN 670
690 ON KE-48 GOTO 770,840,710,740,890
700 REM --------------------
710 GOSUB 1250 :: GOSUB 1200
720 GOSUB 1110 :: CALL MAGNIFY(MF):: GOTO 320
730 REM --------------------
740 CALL DELSPRITE(#2):: GOSUB 1090 :: GOSUB 1050
    :: GOSUB 1200 :: CH=112 :: F=2
     :: CALL MAGNIFY(1)
750 GOSUB 1250 :: GOTO 320
760 REM --------------------
770 GOSUB 1200 :: DISPLAY AT(1,1):"FÖRGRUND?"
780 ACCEPT AT(1,11)SIZE(2)VALIDATE(DIGIT)BEEP:F ::
     IF F(1 OR F)16 THEN 770
790 DISPLAY AT(3,1)SIZE(9):"BAKGRUND?"
800 ACCEPT AT(3,11)SIZE(2)VALIDATE(DIGIT)BEEP:B ::
     IF B(1 OR B)16 THEN 790
810 CALL COLOR(12,B,1):: CALL COLOR(#2,F)
820 GOSUB 1140 :: GOSUB 1150 :: GOTO 670
830 REM --------------------
840 GOSUB 1200 :: DISPLAY AT(1,1):"MAGNIFY-FAKTOR?
    " :: ACCEPT AT(1,17)SIZE(1)VAL
    IDATE("1234")BEEP:MF
850 GOSUB 1110
860 CALL MAGNIFY(MF)
870 GOSUB 1140 :: GOSUB 1150 :: GOTO 670
880 REM --------------------
890 GOSUB 1200 :: GOSUB 1250
900 W=0 :: FOR N=4 TO 11 :: FOR M=16 TO 23
910 CALL GCHAR(N,M,X):: IF X()104 THEN 930
920 W=W+1 :: R2(N)=27-M :: S2(W)=12+N
930 NEXT M :: NEXT N
940 CALL COLOR(12,16,1):: CALL DELSPRITE(#2):: F=2
     :: GOSUB 1050
950 FOR N=1 TO 64 :: IF R2(N)=0 THEN 980
960 CALL HCHAR(R2(N),S2(N),104)
970 NEXT N
980 FOR N=1 TO 64 :: R2(N),S2(N)=0 :: NEXT N
990 GOSUB 1150 :: GOSUB 1180
1000 REM --------------------
1010 CALL KEY(0,KE,ST):: IF (KE(51)+(KE)54)+(ST=0)T
     HEN 1010
1020 GOSUB 1200
1030 ON KE-50 GOTO 710,740,890,440
1040 REM --------------------
1050 FOR N=4 TO 19 :: CALL HCHAR(N,16,96,16):: NEXT
     N
1060 CALL HCHAR(3,23,105):: CALL HCHAR(3,24,106)::
     CALL HCHAR(20,23,105):: CALL
     HCHAR(20,24,106)
1070 CALL HCHAR(11,15,107):: CALL HCHAR(11,32,107):
     : CALL HCHAR(12,15,108):: CAL
     L HCHAR(12,32,108)
1080 REM --------------------
1090 CALL COLOR(12,16,1):: FOR N=8 TO 15 :: CALL HC
     HAR(N,4,120,8):: NEXT N :: RE
     TURN
1100 REM --------------------
1110 IF MF=2 OR MF=4 THEN CH=116 ELSE CH=112
1120 CALL PATTERN(#1,CH):: RETURN
1130 REM --------------------
1140 DISPLAY AT(1,1):"1 NYA FÄRGER":"2 MAGNIFY" ::
     RETURN
1150 DISPLAY AT(3,1)SIZE(12):"3 ÄNDRA"
1160 DISPLAY AT(4,1)SIZE(12):"4 NY SPRITE"
1170 DISPLAY AT(5,1)SIZE(9):"5 ROTERA" :: RETURN
1180 DISPLAY AT(6,1)SIZE(12):"6 KOD+SPRITE" :: RETU
     RN
1190 REM --------------------
1200 FOR N=1 TO 6 :: IF N)2 THEN 1220
1210 CALL HCHAR(N,1,32,32):: GOTO 1230
1220 CALL HCHAR(N,1,32,15)
1230 NEXT N :: RETURN
1240 REM --------------------
1250 CALL HCHAR(21,1,32,128):: RETURN
```

TERRORISTSPELET

av Joakim Söderberg

Spelet är lite våldsfixerat, vilket namnet också antyder. Spelet går ut på att skjuta ner "skateboardmannen" och gubben med en attacheväska. Samtidigt måste man akta sig för en bomb som i de minst anade stunder dyker upp i nedre högra hörnet.
Skulle den dyka upp måste man skjuta ner den.
När man nu har skjutit gubben och klarat av bomben så börjar ibland en polishelikopter att jaga en. Gissa vad man då gör som en god terrorist... Jo, man skjuter ner den.
När polisen ändå tagit fast dig har du chansen att fortsätta men poängen minskar drastiskt. Vid slut av spelet kommer betyget upp. Spelet kräver joystick nr:1 och i grundutförande SPEECH SYNT. har du ingen SPEECH så sätt REMarks där CALL SAY står ensamt (Obs ! ta inte bort raden. En del GOTO satser kanske hänvisar till den raden) där det står i sk multiplestatement (fler kommandon på samma rad) stryk hela CALL SAY kommandot (här får du inte sätta REMarks, då resten av raden skall användas).

HEADER

100-160	REMarks
170-200	PRE-SCAN (snabbare uppstart)
210-260	definitioner
270-320	Instruktioner del 1 text
330	Lägg ut sprites
340-400	Melodi
410-520	Instruktioner del 2 frågor
530-950	Mer Definitioner och skriv på skärm
960-1410	Huvudprogram
1420-1600	Underrutiner
1610-1690	DATA betyget

```
100 !*********************
110 !*  TERRORIST SPELET *
120 !*  by calle & kimme *
130 !*    date 83.04.08  *
140 !*      ORIGINAL     *
150 !*********************
160 !***** PRE-SCAN ******
170 GOTO 210 :: CALL COLOR :: CALL MAGNIFY :: CALL
     CLEAR :: CALL SCREEN :: CALL
    SPRITE :: CALL HCHAR :: CALL VCHAR :: CALL POS
    ITION :: CALL SOUND :: CALL COINC
180 CALL SAY :: CALL PEEK :: CALL CHAR :: CALL KEY
     :: CALL MOTION :: CALL PATTER
    N :: CALL DELSPRITE :: CALL CHARSET :: CALL LO
    CATE :: CALL JOYST
190 X,Y,I,T,F,K,S,D,C,M,A,P,R,TD,T$,PO,NM$,TI,MH,M
    G,BT,CK,FG,BG,TR
200 !éP-
210 CALL CLEAR :: CALL SCREEN(12):: CALL MAGNIFY(3
    ):: GOTO 240
220 CALL CHAR(91,"002800384470444",92,"0028003844
    444438",93,"00100038447C4444",
    64,"3C4299A1A199423C")
230 RETURN
240 GOSUB 220
250 CALL CHAR(36,"0000000000341FF0808040000000000
    0000000000F020FF7F0F0000000000
    00")
260 CALL CHAR(100,"0000000000101F020000000000000000
    0000000000000C0C04000000000000
    000")
270 DISPLAY AT(1,7):"INSTRUKTIONER"
280 DISPLAY AT(3,1):"DU SKA SKJUTA NED SÅ MÅNGA  G
    UBBAR SOM MÖJLIGT PÅ SÅ     KOR
    T TID SOM MÖJLIGT."
290 DISPLAY AT(6,1):"NÄR DU TRÄFFAT EN GUBBE KAN P
    OLISEN KOMMA. OM DU INTE   LYC
    KAS SKJUTA NED DEM FÅR   DU SÄR FÄNGELSE TID."
300 DISPLAY AT(10,1):"DU KAN BLI BOMBAD NÄR SOM
    HELST,OCH DU MÅSTE DÅ SKJUTABO
    MBEN,ANNARS ÄR DU FÖRLORAD"
310 DISPLAY AT(13,1):"OM DU VÄLJER REVOLVER FÅR DU
    BONUS POÄNG."
320 DISPLAY AT(16,1):"1. KULSPRUTA" :: DISPLAY AT(
    18,1):"2. REVOLVER" :: DISPLAY
     AT(23,20):"© JOAKIM","                SÖDE
    RBERG"
330 CALL SPRITE(#1,36,13,118,130):: CALL SPRITE(#2
    ,100,2,134,130)
340 RESTORE 360
350 READ T,F :: IF F=0 THEN 410 :: CALL SOUND(T,F,
    0,F+1,1,F+2,2):: CALL KEY(0,K,
    S):: IF S THEN 410 ELSE 350
360 DATA 400,233,200,220,200,233,400,262,200,233,2
    00,156,200,156,200,147,200,147
    ,200,139
370 DATA 600,147,200,262,200,262,200,262,600,294,2
    00,262,200,262,200,233,200,233
    ,200,220,600,233,200,233,400,311
380 DATA 200,233,200,196,600,156,200,233,300,262,1
    00,294,200,311,200,262,600,233
    ,200,196,600,233,100,262,200,233
390 DATA 200,196,300,233,100,262,200,233,200,196,4
    00,208,400,147,400,156,400,400
    00,400,156,200,176,200,147,600,156
400 DATA 200,196,400,233,400,147,400,156,100,0
410 CALL SAY("WHAT+IS+YOUR+CHOICE")
420 DISPLAY AT(21,1):"VAPEN? " :: CALL KEY(0,K,S):
    : DISPLAY AT(21,1):""
430 IF S=0 THEN 420 :: IF K=49 THEN P=-300 :: GOTO
     450
440 IF K=50 THEN P=300 ELSE DISPLAY AT(21,3):"SVAR
    A 1 ELLER 2 !!" :: FOR D=1 TO
300 :: NEXT D :: GOTO 420
450 CALL SAY("AND+YOUR+NAME")
460 DISPLAY AT(21,1):"NAMN:" :: ACCEPT AT(21,7)VAL
    IDATE(UALPHA):NM$
470 IF P=-300 THEN TI=50 ELSE TI=200
480 DISPLAY AT(21,1):"SVÅRHETS":"GRAD? (1-3)" :: A
    CCEPT AT(22,13)VALIDATE("123")
    :S :: ON S GOTO 490,500,510
490 MH=8 :: BT=25 :: MG=3 :: GOTO 520
500 MH=10 :: BT=20 :: MG=4 :: GOTO 510
510 MH=15 :: BT=17 :: MG=5
520 CALL DELSPRITE(ALL):: CALL CLEAR :: CALL SCREE
    N(2)
530 CALL CHAR(124,"000000000004081010387CFEFEFE7C3
    8000000000000000000000000000")
540 CALL CHAR(36,"0101010F0F0101010101010081A0A0B
    8080B0F0F0B0B0B0B0B0B000B8A8AB
    B8")
550 CALL CHAR(116,"0000000000000201000100000000000
    0000000000040808000000000000
    0")
560 CALL CHAR(48,"0000000000000001030301010100000
    0000000000000000080C000E0800000
    0")
570 CALL CHAR(52,"1313121309070303030302020202060
    C8C848C890E0C0C0C0C0C040404040
    60")
580 CALL CHAR(68,"00000000000007FF8700000000000000
    000000000000EFFDEF222018000000
    00")
590 CALL CHAR(72,"000107091111113F1111110907010000
    0000C020101010F810101020C00000
    00")
600 CALL CHAR(76,"030703010708132307080402063F0C0C
    8080800080402040008040C0FC18
    18")
610 CALL CHAR(80,"0303030301070B137B7B7B0201020406
    80C0008000C0A0908080806010180C
    00")
620 CALL CHAR(140,"0303030301070B1303030302020C080
    080C0808000C0A090BCBCBC4020102
    030")
630 CALL CHAR(100,"0303030301070B07030303010107050
    180C0808000808080C0C0C08040800
    080")
640 CALL CHAR(135,"FFFEFCF8F0E0C080",134,"0103070F
    1F3F7FFF")
650 CALL CHAR(114,"0000AAAAAAAAFFFF",113,"FFFFFFFF
    FFFFFFFF",130,"FFFFFB7B5751000
    0")
660 CALL CHAR(61,"3F3F3F3F3F3F3F3F",62,"FCFCFCFCFC
    FCFCFC")
670 CALL CHAR(131,"0303031F1F1FFFFF",63,"FFFF00FFF
    F00FFFF",132,"C0C0C0F8F8F8FFFF
    ")
```

```
680 CALL CHAR(88,"000000031F3F7FFF",89,"FFFF7F3F3F
    1F1F03",90,"071FFFFFFFFFFFFF",
    91,"FFFFFFFFFFFC380")
690 CALL CHAR(92,"E0F0F3FFFFFFFFFF",93,"FFFFFFFFFF
    FFFF71",94,"00E0F0F0F8FEFEFE",
    95,"FEFFFFFFFFFFCF0")
700 CALL CHAR(96,"00070F1F3F3F7FFF",97,"00C4FCFCFC
    FEFFFF",122,"FEFEFCF8F8FCF060"
    ,123,"3F7E7C7F3F3F3F3F")
710 CALL CHAR(99,"FFFFFFFFFFFFFFFE",98,"E0F8FCFCFC
    FCFEFE",120,"FF7F7F7F3F1F0701"
    ,121,"FFFEFEFEFEFEF8F0")
720 CALL CHAR(129,"1C1C1C1D1E1C1C1C",133,"7722FF22
    22FF2222")
730 CALL CHAR(40,"0103070F1F3F7FFF",41,"FFFFFFFFFF
    FCFCFC",42,"FFFFFFFFFF3C3C3C",
    43,"FFFFFFFFFF3F3F3F")
740 CALL CHAR(44,"80C0E0F0F8FCFEFF",45,"FF3F3F3838
    383838",46,"FFFFFF3838383838",
    47,"FFFCFC3C3C3C3C3C")
750 CALL CHAR(87,"FFFFFFFFC9B6D5FF",64,"7F3E1C1C1C
    1C1C1C",65,"1C1C1C1C1C1C3E77F"
    ,60,"3F202F2F2F2F2F")
760 CALL CHAR(57,"7E027A7A7A7A7A7A",58,"2F2E2F2F2F
    2F2F2F",59,"7A3A7A7A7A7A7A7A")
770 RESTORE 780 :: FOR C=1 TO 14 :: READ CK,FG,BG
    :: CALL COLOR(CK,FG,BG):: NEXT
     C
780 DATA 10,2,1,9,13,6,8,16,6,11,4,6,14,7,4,4,2,16
    ,2,16,6,5,15,16,3,12,6,6,2,2,7
    ,16,12
790 DATA 1,6,6,12,13,4,13,2,4
800 FOR I=1 TO 24 :: CALL HCHAR(I,31,75,4):: NEXT
    I
810 DISPLAY AT(1,12):"XZ8Ö" :: DISPLAY AT(2,3):"XZ
    8Ü     YÄÄ_" :: DISPLAY AT(3,3
    ):"YÄÄ_"
820 DISPLAY AT(5,19):"()*****," :: DISPLAY AT(6,19
    ):"-....../" :: DISPLAY AT(7,1
    9):"-....../"
830 DISPLAY AT(8,19):"WWWWWWW" :: DISPLAY AT(9,19
    ):"ééé(9ééé" :: DISPLAY AT(10,
    5):"@a             ÄÄÄ:ÄÄÄ"
840 DISPLAY AT(11,1):"rrrrröbrrrrrrrrrr"&CHR$(131)
    &"????????"&CHR$(132)&"r":"qqq
    qxyzqqqqqqqqq"&CHR$(134)&"HHHHHHHHH"&CHR$(135)
    &"q"
850 DISPLAY AT(13,1):"qqqqq"&CHR$(129)&"qqqqqqqqq"
    &CHR$(134)&"HHHHHHHHH"&CHR$(13
    5)&"qq"
860 DISPLAY AT(14,1):"qqqqq"&CHR$(129)&"qqqqqqqq"&
    CHR$(134)&"HHHHHHHHH"&CHR$(135
    )&"qqq"
870 DISPLAY AT(15,1):"qqqqq"&CHR$(129)&"qqqqqqq"&C
    HR$(134)&"HHHHHHHHH"&CHR$(135)
    &"qqqq":RPT$(CHR$(133),12)&"=HHHHHHHHH)"&RPT$(
    CHR$(133),5)
880 DISPLAY AT(17,1):"HHHHHHHHHHHHHHHHHHHHHHHHHHH
    " :: DISPLAY AT(18,1):"??HHHH?
    ?HHHH??HHHH??HHHH??HH"
890 DISPLAY AT(19,1):"HHHHHHHHHHHHHHHHHHHHHHHHHH
    ":RPT$(CHR$(130),28)
900 DISPLAY AT(21,1):"qqqqqqqqqqqqqqqqqqqqqqqqqqq
    qqqqqqqqqqqqqqqqqqqqqqqqqqqqq
    qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
    qqqqqqqq"
910 CALL SAY("$READY TO START$"):: CALL SCREEN(2)
920 CALL CHAR(104,"0000000000000000003FFF0000000
    0000700000000000007FC0FF07000
    00")
930 CALL CHAR(108,"00FF02071F7F7070F0F0F0F0FBFF3F0
    000FF0000C0E0781C1C00070707FCF
    000")
940 CALL CHAR(136,"00183C3C180C0F1F274321100000030
    20000000081OE00080C0E0E090880
    503")
950 CALL CHAR(32,"0000000000000000000000000003F0C
    0000000000000000000000000004F8
    60")
960 CALL SAY("USE+JOYSTICK+NUMBER+ONE")
970 CALL DELSPRITE(#10):: CALL SPRITE(#1,76,8,120,
    100,0,-MH,#2,80,10,140,1,#3,72
    ,16,100,150)
980 CALL MOTION(#2,0,MG):: CALL PATTERN(#2,80)
990 TD=TD+1 :: CALL JOYST(1,X,Y):: CALL MOTION(#3,
    -Y*3,X*3)
1000 RANDOMIZE :: CALL PEEK(-31880,R):: IF R<5 THEN
     1100
1010 CALL PATTERN(#2,140):: CALL DELSPRITE(#4)
1020 CALL KEY(1,K,S):: IF K=18 THEN 1200
1030 CALL JOYST(1,X,Y):: CALL MOTION(#3,-Y*3,X*3)
1040 CALL KEY(1,K,S):: IF K=18 THEN 1200
1050 CALL PATTERN(#2,100):: CALL JOYST(1,X,Y):: CAL
    L MOTION(#3,-Y*3,X*3)
1060 CALL KEY(1,K,S):: IF K=18 THEN 1200
1070 CALL PATTERN(#2,80):: CALL JOYST(1,X,Y):: CALL
     MOTION(#3,-Y*3,X*3)
1080 CALL KEY(1,K,S):: IF K=18 THEN 1200
1090 GOTO 990
1100 CALL PATTERN(#2,52):: CALL MOTION(#2,0,0):: CA
    LL SPRITE(#10,124,15,100,249,
    15,-10)
1110 FOR D=1 TO 135 :: NEXT D :: CALL SOUND(50,-7,1
    )
1120 CALL MOTION(#10,-30,-10):: CALL JOYST(1,X,Y)::
     CALL MOTION(#3,-Y*3,X*3):: C
    ALL MOTION(#10,35,-20):: CALL SOUND(35,40000,1
    )
1130 CALL SOUND(50,-7,1):: CALL MOTION(#10,0,0):: I
    =0
1140 FOR I=1 TO BT :: CALL KEY(1,K,S):: IF K=18 THE
    N 1170
1150 CALL JOYST(1,X,Y):: CALL MOTION(#3,-Y*4,X*4)
1160 CALL MOTION(#3,-Y*3,X*3):: CALL SOUND(50,400,1
    ):: NEXT I :: GOTO 1180
1170 I=I+1 :: CALL SOUND(-50,-6,1,400,1,400,1):: CA
    LL COINC(#3,#10,10,TR):: IF T
    R=-1 THEN 970 ELSE 1150
1180 CALL SOUND(500,-6,1):: CALL CLEAR :: CALL CHAR
    SET :: CALL DELSPRITE(ALL)::
    FOR I=1 TO 20 :: CALL SCREEN(7):: CALL SCREEN(
    12)
1190 CALL SOUND(50,-7,I):: CALL SCREEN(16):: NEXT I
     :: CALL SAY("YOU+ARE+FINISHE
    D"):: GOSUB 220 :: GOTO 1590
1200 CALL PATTERN(#2,52):: CALL MOTION(#2,0,0)
1210 CALL JOYST(1,X,Y):: CALL MOTION(#3,-Y*3,X*3)
1220 CALL POSITION(#3,R,K):: CALL SPRITE(#4,116,16,
    R,K):: CALL SOUND(-TI,-6,1)
1230 CALL COINC(#2,#3,2,TR):: CALL COINC(#1,#3,2,T)
1240 IF TR=-1 THEN 1270 :: IF T=-1 THEN 1330 :: IF
    TI=200 THEN 1250 ELSE 1260
1250 FOR D=1 TO 7 :: CALL JOYST(1,X,Y):: CALL MOTIO
    N(#3,-Y*3,X*3):: NEXT D
1260 CALL KEY(1,K,S):: IF S=0 THEN 980 ELSE 1210
1270 FOR I=1 TO 20 STEP 5 :: CALL SOUND(-1,2000-I*9
    ,I):: NEXT I
1280 CALL SAY("UHOH+VERY+GOOD")
1290 CALL DELSPRITE(#4):: CALL PATTERN(#2,68)
1300 CALL POSITION(#2,R,K):: CALL SPRITE(#5,36,15,R
    ,K+10)
1310 FOR D=1 TO 300 :: NEXT D
1320 P=P+200 :: CALL DELSPRITE(#5):: GOTO 1360
1330 CALL SAY("OH"):: CALL PATTERN(#1,136):: CALL S
    OUND(-70,-5,1):: CALL MOTION(
    #1,0,0,#3,0,0):: CALL COLOR(#1,10)
1340 CALL DELSPRITE(#4):: CALL POSITION(#1,R,K):: C
    ALL SPRITE(#9,32,8,R,K,0,-MH)
1350 FOR D=1 TO 500 :: NEXT D :: P=P+50 :: CALL DEL
    SPRITE(#9):: GOTO 1360
1360 RANDOMIZE :: CALL PEEK(-31880,R):: IF R>50 THE
    N 1370 ELSE 970
1370 CALL SAY("LOOK+OUT"):: CALL SPRITE(#6,104,2,30
    ,1,#7,108,2,30,18,#8,48,16,30
    ,21)
1380 CALL POSITION(#8,R,C):: IF R>30 THEN M=-10 ELS
    E M=10 :: IF C>230 THEN 1460
1390 TD=TD+1 :: CALL SOUND(-10,700,1):: CALL MOTION
    (#6,M,MH,#7,M,MH,#8,M,MH)
1400 CALL JOYST(1,X,Y):: CALL MOTION(#3,-Y*3,X*3)::
     CALL KEY(1,K,S):: IF K=18 TH
    EN 1410 :: GOTO 1380
1410 CALL SOUND(-50,-7,0):: CALL COINC(#8,#3,3,T)::
     IF T=-1 THEN 1420 ELSE 1380
1420 CALL DELSPRITE(#8):: CALL MOTION(#6,10,5,#7,10
    ,5)
1430 FOR I=1 TO 600 STEP 40 :: CALL SOUND(-5,2600-I
    ,0):: NEXT I
1440 CALL SAY("#GOOD WORK$")
1450 CALL SOUND(1000,-7,0,120,0):: P=P+200 :: A=A-2
     :: CALL DELSPRITE(#6,#7,#8,#
    9):: GOTO 970
1460 CALL DELSPRITE(ALL):: CALL SAY("$NICE TRY$")
1470 CALL CLEAR :: CALL SCREEN(15):: CALL CHARSET :
    : GOSUB 220
1480 CALL SAY("YOU+ARE+FINISHED ")
1490 A=A+5 :: P=P-100 :: IF A<0 THEN A=0
1500 DISPLAY AT(3,7):"MISS! HE,HE,HEE"
1510 DISPLAY AT(5,5):"OK,"&NM$&", DU KAN NU VALJA":
    "ATT TJÄNA AV DITT FÄNGELSE"
1520 DISPLAY AT(8,1):"STRAFF PÅ "&STR$(A)&"ÅR ELLER
     FORT-   SÄTTA SPELET"
1530 DISPLAY AT(10,5):"OCH OM DU DÅ LYCKAS SKJUTA
    NED NÅGON POLIS REDUCERAS   D
    IN FÄNGELSE TID."
1540 DISPLAY AT(14,5):"MEN OM DU MISSAR EN HELI-
    KOPTER TILL ÖKAR DIN        F
    ÄNGELSE TID MED 5 ÅR!"
1550 DISPLAY AT(18,6):"DU HAR NU "&STR$(P)&" POÄNG.
                 VAD VÄLJER DU??"
1560 DISPLAY AT(22,3):"1. FORTSÄTTA  2. SLUTA"
1570 CALL KEY(0,K,S):: IF S=0 THEN 1570 :: IF K=49
    THEN CALL SAY("$THAT IS RIGHT
    $"):: GOTO 520 :: IF K=50 THEN 1580
1580 CALL SAY("YOUR+CHOICE+ARE+COMPLETE+MEAN+LESS")
    :: CALL CLEAR :: DISPLAY AT(6
    ,1):NM$&", DU ÄR EN RIKTIG"&"FEGIS! "
1590 DISPLAY AT(10,1):"DIN POÄNG ÄR "&STR$(P)&" OCH
     DET":"TOG DIG "&STR$(TD)&" M
    INUTER."
1600 T=P-TD-A*30+1500 :: IF T>2000 THEN 1640 ELSE 1
    610
1610 RESTORE 1630
1620 READ T$,PO :: IF T<PO THEN 1620 ELSE 1670
1630 DATA GÄNSKA SÄ DÅLIG,1800,DÅLIG,1600,VÄLDIGT D
    ÅLIG,1400,USEL,1200,VÄRDELÖS,
    1000,TOTALT VÄRDELÖS,500
1640 RESTORE 1660
1650 READ T$,PO :: IF T>PO THEN 1650 ELSE 1670
1660 DATA GODKÄND,2400,HYFSAD,2500,KLART BRA,3000,V
    ÄLDIGT BRA,3200,SUVERÄN,3600,
    FANTASTISKT BRA,4000,ÖVERLÄGSET BÄST,4200
1670 DISPLAY AT(15,5):"DET GÅR DIG TILL EN
    "&T$&" TERRORIST !" :: DISPLA
    Y AT(23,20):"© JOAKIM","                    SÖD
    ERBERG"
1680 CALL SAY("$TEXAS INSTRUMENTS$+THINK+YOU+ARE+A+
    VERY+GOOD+PARTNER+AFTER+ALL")
1690 CALL SAY("GOODBYE"):: FOR D=1 TO 2000 :: NEXT
    D :: CALL CLEAR
```

BARNVAKTEN

av Björn Gustavsson

Om du inte har råd att ha en mänsklig barnvakt som håller barnen lugna när du är ute och springer på stan, är det här programmet något för dig. Helt automatiskt skriver det ut "sagor" bestående av grammatiskt riktiga meningar. Ordförrådet kan lätt bytas ut eller utökas.

Adjektiv finns i DATA-satser på rad 780 och 790. Fler adjektiv kan sättas in, under förutsättning att det sista adjektivet följs av en asterisk ("*"). Varje adjektiv måste finnas i tre former i följande ordning:

En hårig ko.
Ett hårigt hus.
Några håriga kor.

Substantiv finns från rad 800 med en asterisk efter det sista substantivet. Substantiven finns i sex former (i nu nämnd ordning)::

En ko.
Ett ägg.
Den kon.
Det ägget.
Några ägg.
De äggen.

Former som inte finns måste lämnas tomma. "Ägg" skrivs t ex så här:

DATA ,ägg,,ägget,ägg,äggen

Från rad 820 finns satser. (Den sista satsen måste följas av en asterisk.) Nummertecken ("#") anger att en plats där ett slumpmässigt valt substantiv och eventuellt adjektiv kan sättas in.
Exempel: Om vi har angivit satsen "# lagar ett staket" kan datorn av detta skapa bl a dessa satser:

En hårig ko lagar ett staket.
Ett flitigt ägg lagar ett staket.
En sömnig greve lagar ett staket.

(Understrykning anger nummertecknets position.)

Slutligen finns hela meningar från rad 840. (Den sista meningen måste följas av en asterisk.) En ampersand ("&") anger en hel sats.

Exempel: Om vi har meningen

&, men &

kan vi få bl a följande meningar:

En hårig ko lagar ett staket, men några vackra monster dricker mjölk.

Ett sömnigt ägg sjunger för en myra, men någon greve sjunger.

(Understrykning anger en ampersands position.)

Se till att ALPHA LOCK är uppsläppt när du matar in DATA-satserna 770 till 840.

```
100 REM  BARNVAKTEN
110 REM  Ett program av
120 REM  Bjirn Gustavsson
130 REM
140 DEF FIND(A$)=POS(FRAS2$,A$,
    1)
150 DEF SUBSTITUTE$(A$)=SEG$(FR
    AS2$,1,Q-1)&A$&SEG$(FRAS2$,
    Q+1,LEN(FRAS2$)-Q)
160 DEF INTRND(X)=INT(RND*X)
170 DIM ADJ$(30,2),SUBST$(30,5)
    ,BEST$(5,2),VERB$(30),MENIN
    G$(30)
180 RANDOMIZE
190 CALL CLEAR
200 GOSUB 860
210 RESTORE
220 FOR I=0 TO 5
230 READ BEST$(I,0),BEST$(I,1)
240 NEXT I
250 NA=0
260 NS=0
270 NV=0
280 READ TEMP$
290 IF TEMP$="*" THEN 340
300 ADJ$(NA,0)=TEMP$
310 READ ADJ$(NA,1),ADJ$(NA,2)
320 NA=NA+1
330 GOTO 280
340 READ TEMP$
350 IF TEMP$="*" THEN 420
360 SUBST$(NS,0)=TEMP$
370 FOR I=1 TO 5
380 READ SUBST$(NS,I)
390 NEXT I
400 NS=NS+1
410 GOTO 340
420 READ VERB$(NV)
430 IF VERB$(NV)="*" THEN 460
440 NV=NV+1
450 GOTO 420
460 NM=0
470 READ MENING$(NM)
480 IF MENING$(NM)="*" THEN 520
490 NM=NM+1
500 GOTO 470
510 REM  HUVUDPROGRAM
520 FRAS2$=MENING$(INTRND(NM))&
    "."
530 FRAS1$=VERB$(INTRND(NV))
540 Q=FIND("&")
550 IF Q=0 THEN 580
560 FRAS2$=SUBSTITUTE$(FRAS1$)
570 GOTO 530
580 GOSUB 680
590 Q=FIND("#")
600 IF Q=0 THEN 630
610 FRAS2$=SUBSTITUTE$(FRAS1$)
620 GOTO 580
630 BIG$=CHR$(ASC(FRAS2$)-32)
640 FRAS2$=BIG$&SEG$(FRAS2$,2,L
    EN(FRAS2$)-1)
650 PRINT #UT:FRAS2$::
660 GOTO 520
670 REM  SLUMPAR FRAM SUBSTANTI
    V OCH ADJEKTIV
680 S=INTRND(NS)
690 F=INTRND(5)
700 IF SUBST$(S,F)="" THEN 690
710 A=-F*(F<2)-2*(F>1)
720 FRAS1$=BEST$(F,INTRND(2))&"
    "
730 IF RND>RND THEN 750
740 FRAS1$=FRAS1$&ADJ$(INTRND(N
    A),A)&" "
750 FRAS1$=FRAS1$&SUBST$(S,F)
760 RETURN
770 DATA en,n}gon,ett,n}got,den
    ,den,det,det,n}gra,hennes,d
    e,hans
780 DATA h}rig,h}rigt,h}riga,gl
    upsk,glupskt,glupska,s!mnig
    ,s!mnigt,s!mniga,vacker,vac
    kert,vackra
790 DATA flitig,flitigt,flitiga
    ,*
800 DATA ,{gg,,{gget,{gg,{ggen,
    myra,,myran,,myror,myrorna,
    ko,,kon,,kor,korna,katt,,ka
    tten,,katter,katterna
810 DATA greve,,greven,,,,,mons
    ter,,monstret,monster,,*
820 DATA # sjunger,# sjunger f!
    r #,# {ter #,# {ter,# dansa
    r med #,# dricker mj!lk,hon
     dansar med #
830 DATA # sover,# lagar ett st
    aket,# och # dansar,*
840 DATA &,"&, men &","&, och &
    ",&,"och &",*
850 REM  Subrutin som definiera
    r om sm} bokst{ver.
860 CALL CHAR(48,"0038444C54644
    438")
870 CALL CHAR(64,"0008107C40784
    07C")
880 CALL CHAR(79,"0038444444444
    438")
890 RESTORE 940
900 FOR I=91 TO 126
910 READ A$
920 CALL CHAR(I,A$)
930 NEXT I
940 DATA 00280038447C4444
950 DATA 0028007C4444447C
960 DATA 00382838447C4444
970 DATA 0028004444444438
980 DATA 3C4299A1A199423C
990 DATA 0008103844784038
1000 DATA 0000003843C443C
1010 DATA 0040407844444478
1020 DATA 0000003C4040403C
1030 DATA 0004043C4444443C
1040 DATA 0000003844784038
1050 DATA 000C101010381010
1060 DATA 0000003C443C0438
1070 DATA 0040405864444444
1080 DATA 0010003010101038
1090 DATA 0008001808084830
1100 DATA 0020202428302824
1110 DATA 0030101010101038
1120 DATA 0000007854545454
1130 DATA 0000007844444444
1140 DATA 0000003844444438
1150 DATA 0000007844784040
1160 DATA 0000003C443C0404
1170 DATA 0000005864404040
1180 DATA 0000003C40380478
1190 DATA 0010103810101000
1200 DATA 000000444444443C
1210 DATA 0000004444282810
1220 DATA 0000004444545428
1230 DATA 0000004428102844
1240 DATA 0000004428101010
1250 DATA 0000007C0810207C
1260 DATA 0028003843C443C
1270 DATA 0028003844444438
1280 DATA 0010003843C443C
1290 DATA 0000280044444438
1300 RETURN
```

Tankar om årets MikrodatorMässa

Jag var på mässan för andra året i rad. Gick omkring för att se vad nytt tillverkarna skapat på ett år. För mig var inget av hårdvaran revolutionerande, men delvis nytt. Alla maskiner man sett och hört om fanns där, plus kopior på dessa varianter. IBM, APPLE,DEC, CANON, COMMODORE, HEWLETT-PACKARD, etc. etc. Nyheten låg snarare i mjukvaran, de imponerade på mig ordentligt.

Apples nya dator Macintosh med sitt program Mac_Write, för ordbehandling. Ett storartat genombrott av de nya programvarutankar vilket mjukvaru-tillverkarna tydligen har insett. Macintosh program fick folk att nästan gapa av förundran över vad den nya tekniken kan åstadkomma. Tankarna att låta olika program samverka med varandra kommer från Lotus 1-2-3. Ett integrerat kalkyl, ord och bokföringsprogram. På Macintosh kunde man arbeta med text, teckned bild och lägga in kalkyl i texten. Man har en textfil och en "skissblock"-fil samt en kalkylfil. Vilka alla tre gick att sammanlänka med ordbehandlingen. Du bläddrade - det var verkligen ett bläddrande - i ditt ritblock och valde den teckningen du tyckte att du lyckats bäst med och lade in den i texten var helst du ville ha den och nästan hur som helst. Jag stod där och stirrade i förundran när Apple-folkets suveräna demonstration visades. Min tanke sade mig det vore någonting att imitera på min Texas dator. Kanske inte så stort men ideen med sammanlänkade program. En utmaning för oss som har en TI99/4A.

På tal om Texas datorer. Texas Instrument visade givetvis sin Professional Computer. Jag tittade och hörde på deras program med deras röst och taligenkännigsenhet. Fantastisk, återigen programvara som överträffade ryktena. Du talade till datorn ledigt och obesvärat, vissa nyckelord behövde den givetvis. Exempelvis kalkyl, månads-rapport, fortsätt och liknande ord. Denna programvara kan bli någonting för människor med nedsatt rörelseförmåga. Mer program av den sorten Texas Instrument! En framtidvision blixtrade till inom mig att handikappade får ta del av den nya tekniken. Tänk att kunna tala med sina kompisar genom en dator, en prator, taligenkänningsenhet och modem. Hisnande.

Texas Instruments CAD/CAM program visade sig vara professionellt, med en mycket fin grafisk upplösning. Möjlighet att kraftigt uppförstora delar av konstruktionen. Jag hoppas Texas Instrument tänker till ordentligt den här gången när det gäller lansering och uppbackning av programvara och liknande så att det inte går som med vår lilla 99:a. TI har alltid haft en inneboende drivkraft att komma med nya ideer när det gäller program till såväl hårdvara som mjukvara. Fortsätt med det TI och lägg på ett extra kol med mjukvaran.

IBM visade sitt nya program ExecuVision att användas till IBMs PC. Underbart! Helt enkelt underbart för en gammal tavelmålare som jag. Du kan rita dina egna bilder och färglägga dem som du vill i god högupplösning. Men det smartaste av allt, du har möjlighet att simulera animering i 10 olika hastigheter. Rörliga bilder på en persondator, som i stort sett endast förr gick att göra med stordatorer. En mycket givande utveckling. Var bär det hän i framtiden ?

Göran Nygren
En datorsmittad 34-åring.

Säljes:
Shugart drive 40 spår.
Dubbelsidig,virrande ny.
pris :2800 kr
tel :08/216278
(fråga efter Love)

KÖPES

TI:s Text-To-Speech diskett, PHD 5075

Peter Odelryd

SÄLJES

Republic Software Utilities I
på kassett med manual (original)

Peter Odelryd
Middagsvägen 6
146 00 TULLINGE

SÄLJES

Household Budget Management, modul
(planera din egen hembudget)
Säljes på grund av inköp av Multiplan

Göran Nygren
Uppingegränd 10
163 61 SPÅNGA

Nyheter i omlopp!!!

Extended Basic-modulen tillverkas fortfarande.
På moduler som fanns att köpa i slutet av
april -84 stod det LTA 0684 under. Borde betyda
att de är tillverkade under vecka 6 i år.
Vet någon om fler moduler fortfarande tillverkas?

Peter Odelryd

Utdrag ur brev från en av våra
"Internationella kontakter"

MAURICE E. T. SWINNEN
Texas Instruments Software Consultant

Dear Claes,
Thanks for sending the nice letter and the continued sending of Nittinian. I always enjoy reading it. Even if it took you a long time to put it together, it is still a fantastic piece of reading.
Many thanks again.

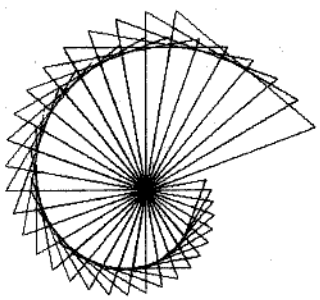
I am for the moment engaged in helping to write a local 99-er newsletter.

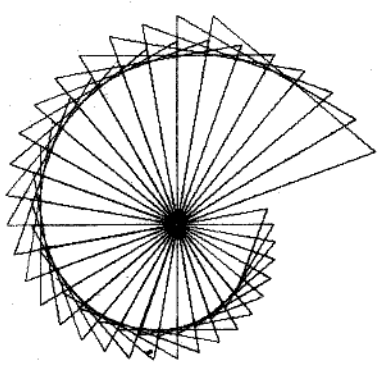
Now we are forming some special interest groups, also to meet each month. These groups will deal with assembly language, CC-40 and a last one with a special plotterprinter that Radio-Shack had on sale.....
...It plots very nice pie charts, sine curves, and even 3-D pictures. I have enclosed a few sample with this letter, to make your mouth water.....

In a separate envelope (a large one) I send you all the documentation plus a diskette with FORTH version 3.0. I think you will know what to do with it.....

It is in public domain, so not copyrighted.

(Kuväret vägde för US$ 23. flyg, å kom dagen före brevet ovan) csch.



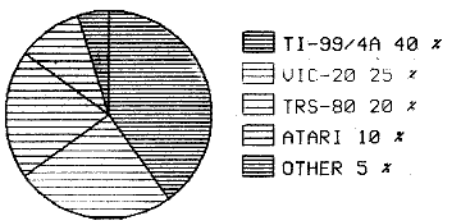


```
10 REM spiral pattern,creative computing
,feb 84,p.22,maurice swinnen
20 CALL CLEAR :: OPEN #1:"RS232/2.BA=600
" :: PRINT #1:CHR$(18):"M240,-240":"I"
30 D=10 :: R=210 :: F=PI/3
40 FOR J=1 TO 30
45 C=J-INT(J/3)*3+1 :: PRINT #1:"C";C
50 D=D+10
60 R=R-4 :: K=D*PI/180 :: Y1=R*SIN(K)::
X1=R*COS(K)
70 Y2=R*SIN(K+F):: X2=R*COS(K+F)
80 PRINT #1:"D";X1;",";Y1;",";X2;",";Y2;
",0,0"
90 NEXT J
100 PRINT #1:"HM0,-200":"C0":"A" :: CLOS
E #1 :: END
```

TI-99/4A

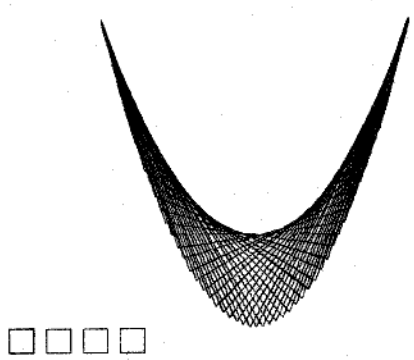
COLORGRAPHIC PRINTER

----------PIE GRAPH----------

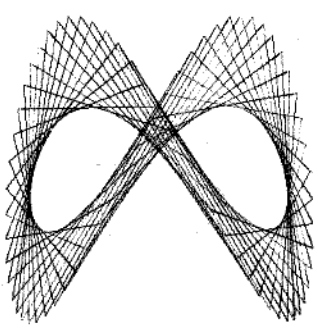


--------SINE AND COSINE CURVES--------

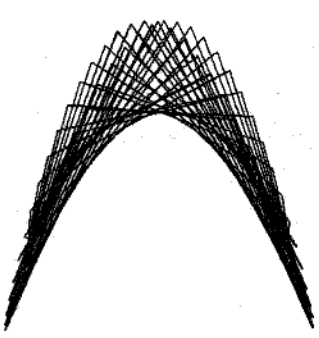
20 10 40 31

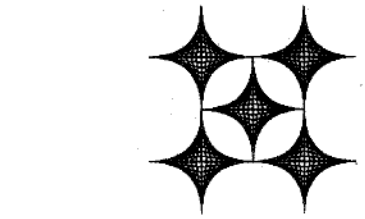


20 10 40 29

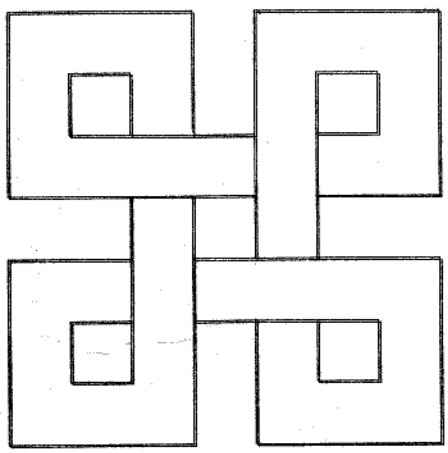


20 10 40 28

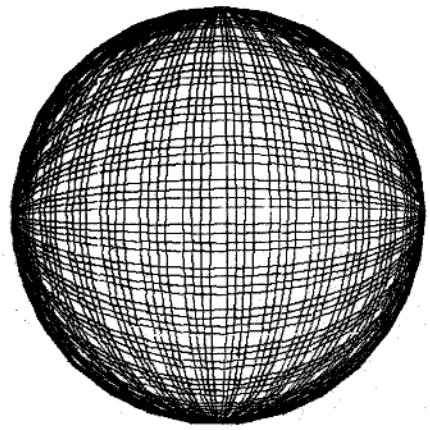


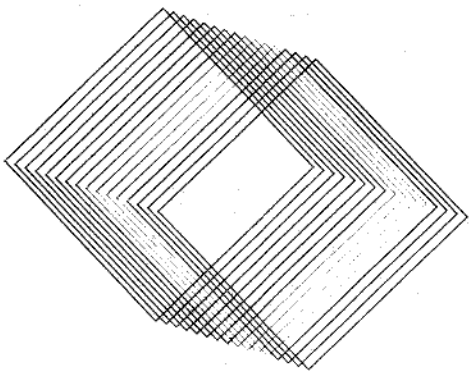


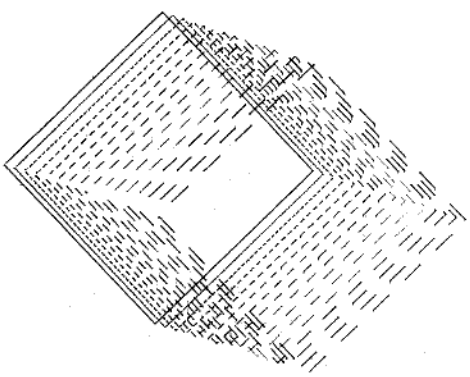
```
10 REM LISTING 1C,P.155,CREATIVE COMPUTI
NG,FEB 84,XBASIC,MAURICE SWINNEN
15 CALL CLEAR :: OPEN #1:"RS232/2.BA=600
" :: PRINT #1:CHR$(18):"M240,-240":"I"
20 FOR N=1 TO 5 :: READ X,Y
30 CX=X :: CY=Y :: GOSUB 70
40 NEXT N
50 PRINT #1:"M0,-240":"A" :: CLOSE #1 ::
 END
60 DATA 0,0,-50,-50,50,-50,-50,50,50,50
70 FOR C=1 TO 4
75 PRINT #1:"C";C-1
80 FOR X2=0 TO 50 STEP 5
90 Y=50-X2 :: X1=X2
100 IF C=2 THEN Y=-Y
110 IF C=3 THEN Y=-Y :: X1=-X2
120 IF C=4 THEN X1=-X2
130 X=CX :: Y=CY+Y :: GOSUB 1000
140 X=CX+X1 :: Y=CY :: GOSUB 2000
150 NEXT X2
160 NEXT C :: RETURN
1000 PRINT #1:"M";X;",";Y :: RETURN
2000 PRINT #1:"D";X;",";Y :: RETURN
```

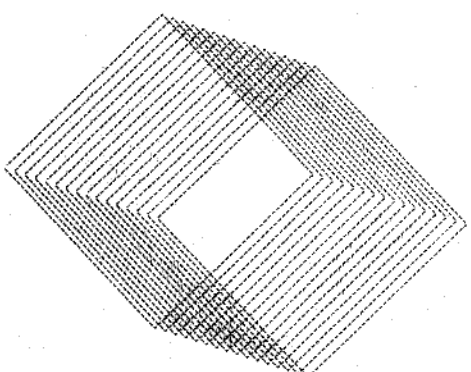


```
!00 REM   DRAW INTERLACE ON PLOTTER, TI BASIC, LARRY HUGHES,2/2/84
110 OPEN #1:"RS232/2.BA=600",OUTPUT
120 PRINT #1:CHR$(18):"M30,-150":"I"
130 FOR G=1 TO 2
140 FOR I=1 TO 333
150 READ A$
160 PRINT A$
170 IF A$="STOP" THEN 230
180 REM  INPUT B$
190 IF B$="" THEN 210
200 A$=B$
210 PRINT #1:A$
220 NEXT I
230 PRINT #1:"R2,-2":"I"
240 RESTORE
250 NEXT G
260 PRINT #1:"M0,-550":"A"
270 CLOSE #1
280 END
290 DATA "R188,-128"
300 DATA "J0,128"
310 DATA "J-188,0"
320 DATA "J0,-188"
330 DATA "J248,0"
340 DATA "R60,0"
350 DATA "J128,0"
360 DATA "J0,188"
370 DATA "J-188,0"
380 DATA "J0,-248"
390 DATA "R8,-60"
400 DATA "J0,-128"
410 DATA "J188,0"
420 DATA "J0,188"
430 DATA "J-248,0"
440 DATA "R-60,0"
450 DATA "J-128,0"
460 DATA "J0,-188"
470 DATA "J188,0"
480 DATA "J0,248"
490 DATA "R60,0"
500 DATA "R0,60"
510 DATA "J-188,0"
520 DATA "J0,60"
530 DATA "J60,0"
540 DATA "J0,-60"
550 DATA "R8,-60"
560 DATA "J0,-188"
570 DATA "J-60,0"
580 DATA "J0,60"
590 DATA "J60,0"
600 DATA "R60,0"
610 DATA "J188,0"
620 DATA "J0,-60"
630 DATA "J-60,0"
640 DATA "J0,60"
650 DATA "R8,60"
660 DATA "J0,188"
670 DATA "J60,0"
680 DATA "J0,-60"
690 DATA "J-60,0"
700 DATA "H"
710 DATA "STOP"
```





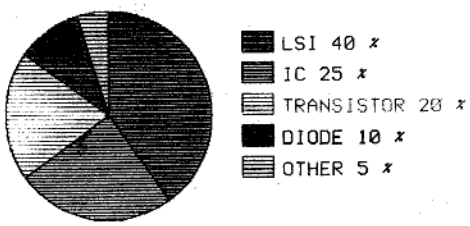




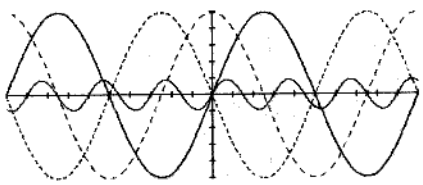
TI-99/4A

COLORGRAPHIC PRINTER

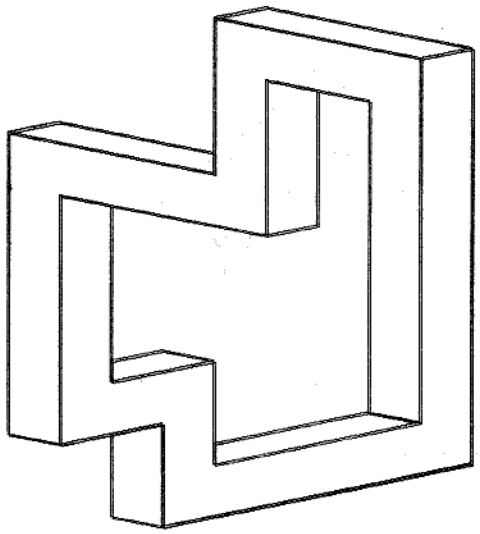
----------PIE GRAPH----------



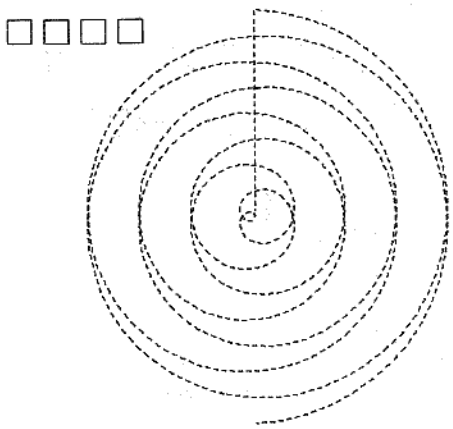
--------SINE AND COSINE CURVES--------

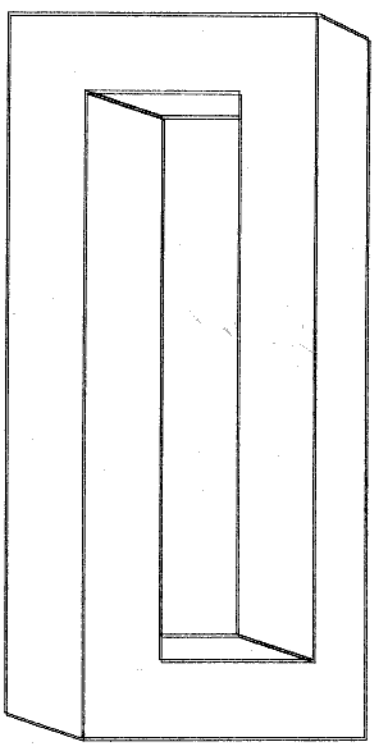


---THE END---



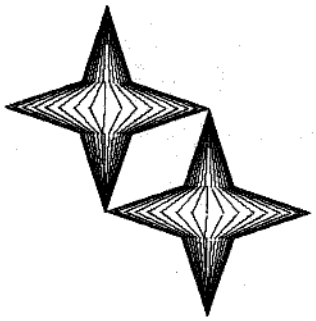
```
100 REM ESCHER DRAWING B,XBASIC,MAURICE SWINNEN,23 FEB 84
110 OPTION BASE 1
120 DIM X(50),Y(50)
130 FOR N=1 TO 50 :: READ X(N),Y(N):: NEXT N
140 CALL CLEAR :: OPEN #1:"RS232/2.BA=600" :: PRINT #1:CHR$(18);"CR";"M5,-500";"
I"
150 FOR Q=1 TO 2 :: FOR N=1 TO 50 :: PRINT #1;"D";X(N);",";Y(N):: NEXT N :: PRIN
T #1:"M";"R2,-2";"I" :: NEXT Q
160 PRINT #1:"M0,-200":"A" :: CLOSE #1 :: END
170 DATA 0,230,200,250,200,270,50,300,0,230,200,250,200,400
180 DATA 400,360,450,320,250,410,200,400,400,360,400,10,350,20
190 DATA 200,-10,200,-30,400,10,350,20,350,320,250,340,250,190
200 DATA 300,200,300,330,250,340,250,190,50,230,50,-10,0,0
210 DATA 30,-10,150,10,150,-30,100,-60,100,0,150,10,150,-30
220 DATA 450,-30,450,270,450,-30,150,-30,150,10,50,-10,50,230
230 DATA 100,220,100,50,200,70,150,80,100,70,100,50,200,70,200,-30
```



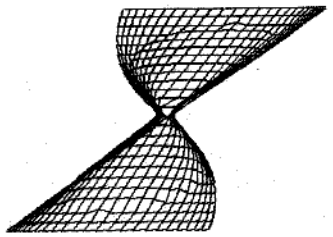


```
100 REM DOUBLE SPIRAL
110 REM TI-BASIC MAURICE SWINNEN FEB 4 1984
120 CALL CLEAR
130 OPEN #1:"RS232/2.BA=600"
140 R=200
150 PRINT #1:CHR$(18)
160 PRINT #1:"L4"
170 PRINT #1:"C8"
180 PRINT #1:"M240,-240"
190 PRINT #1:"I"
200 PI=3.14159
210 P2=2*PI
220 FOR I=0 TO 800 STEP 2
230 S=I/100*P2
240 X=INT(SIN(S)*R)
250 Y=INT(COS(S)*R)
260 PRINT #1:"D";X;",";Y
270 R=R-1
280 NEXT I
290 PRINT #1:"M0,-240"
300 PRINT #1:"A"
310 CLOSE #1
320 END
```

```
100 REM ESCHER DRAWING C,XBASIC,MAURICE SWINNEN,MARCH 1984
110 OPTION BASE 1
120 DIM X(21),Y(21)
130 FOR N=1 TO 21 :: READ X(N),Y(N):: NEXT N
140 CALL CLEAR :: OPEN #1:"RS232/2.BA=600" :: PRINT #1:CHR$(18);"M0,-500";"I";"M
350,350"
150 FOR Q=1 TO 2 :: FOR N=1 TO 21 :: PRINT #1:"D";X(N);",";Y(N):: NEXT N :: PRIN
T #1:"M";"M2,-3";"I";"M350,350" :: NEXT Q
160 PRINT #1:"M0-350,-500":"A" :: CLOSE #1 :: END
170 DATA 400,325,400,-350,125,-350,50,-325,50,350,350,350,350,-275
180 DATA 200,-275,200,-250,275,-250,350,-275,275,-250,275,275,125,275
190 DATA 200,250,275,250,200,250,200,-250,200,250,125,275,125,-350
```

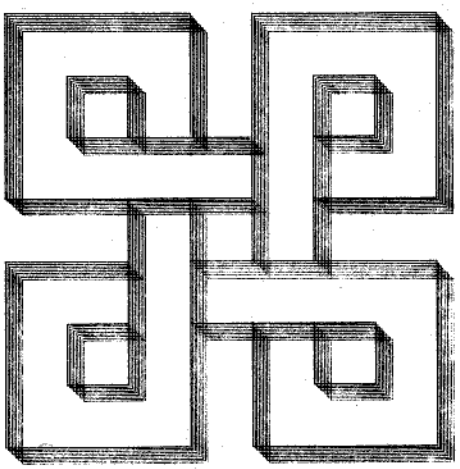


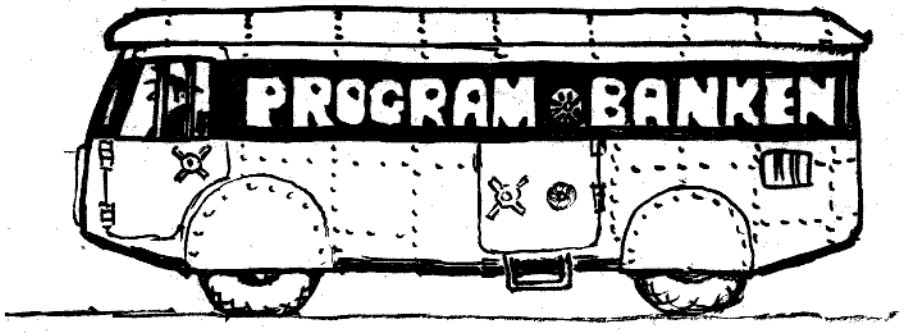
□□□□



```
100 REM FOUR-STAR,LISTING 3,CREATIVE COMPUTING,FEB 84,P.155,XBASIC,MAURICE SWINN
EN
110 CALL CLEAR :: OPEN #1: "RS232/2.BA=600" :: PRINT #1:CHR$(155);"W
"C0"
120 S=PI/20
130 RESTORE
140 GOSUB 270
150 FOR AN=0 TO PI STEP S
160 PRINT #1:"C";INT(AN)+1
170 FOR N=1 TO 5
180 U=X(N)*COS(AN):: W=Y(N)
190 IF N=1 THEN 210
200 GOSUB 300
210 U=INT(U):: W=INT(W):: GOSUB 290
220 NEXT N
230 NEXT AN
240 IF F1=1 THEN 260
250 F1=1 :: PRINT #1:"M";-100,-100";" :: GOTO 130
260 PRINT #1:"M0,-100";"C0";" :: CLOSE #1 :: END
270 FOR N=1 TO 5 :: READ X(N),Y(N):: NEXT N :: RETURN
280 DATA 0,100,25,25,100,0,25,-25,0,-100
290 PRINT #1:"D";U;",";W :: RETURN
300 PRINT #1:"D";U;",";W :: RETURN
```

□□□□





Här visas en lista på de tillgängliga programmen i Programbanken. Programmen kan numera köpas. Priset är uppdelat i två delar, dels en startkostnad som täcker kostnaden för porto och mediet, dels en kopieringsavgift för varje program.

Startkostnad, kassett	**35:-**
Startkostnad, diskett	**55:-**
Kopieringsavgift, kassett	**15:-**
Kopieringsavgift, diskett	**10:-**

Som förut får du tre program i utbyte när du skickar in ett BASIC-program.

De två första siffrorna i programnumret anger vilken diskett programmet ligger på.
De nästa två siffrorna anger programkategori se nedan
Femte siffran anger ursprungsland:
1 = USA, 2 = Holland, 7 = Sverige.
De tre sista siffrorna är ett löpnummer och bokstaven som finns sist anger vilket språk som används i programmet: E = Eng., H = Holl., F = Fra., S = Svenska.
Bokstäverna efter numret talar om vilken utrustning som krävs: X = Ext. Basic, D = Diskett, P = Printer E = Minnesexp., J = Joystick, C = Call Files (1) måste användas om man har disk. Ett + anger att programmet är delat i flera delar.

Programkategori:
14 = Ekonomi
20 = Regression, Kurvanpassning
21 = Statistik, Varians
29 = Statistik, Sannolikhet
30 = Linjär algebra
39 = Allmän matamatik
65 = Elektronik
78 = Astronomi
90 = Praktiska programhjälpmedel
91 = Spel
92 = Utbildning
96 = Musik
97 = Demo och Musik
99 = Övrigt

01201001E ___ LINEAR REGRESSION
01201002E ___ TREND LINES ANALY
01391001E ___ 8 MATH PROGRAMS
01391002E ___ SIMULT. EQUATIONS
01651001E ___ ELECTRONICS
01781001E ___ GEOSTAT. SATELITE
01901001E __P WORD-PROCESSOR 1
01901002E ___ MAILING LIST
01911028H ___ CAR DRIVER
01911029H ___ ANIMALS
01911030H ___ SPACE INVADER
01921006H __X GENETICS
01971001H __C 99/4A DEMO
01991001E __X ANCESTRAL FILER
01991002H ___ CALENDAR W. PRINT
01992001E ___ MOON CYCLE
02911001E ___ MASTERMIND
02911002E __X SCHMOO
02911003E ___ STARTRECK
02911004E ___ AIR - SEA BATTLE
02911005E ___ QUEENS
02911006E ___ OTHELLO 2-SPELARE
02911007E ___ CHASE
02911008E ___ PUZZLE 15
02911009E ___ NUMBERS AWAY
02911010E ___ AWARI
02911011E ___ KENO
02911012E _+D STARGUARD
03911013E ___ MINER
03911014E __C YAHTZEE
03911015E ___ BACKGAMMON
03911016E ___ 3D TIC-TAC-TOE
03911017E __X BLACKJACK
03911018E __X NOT ONE
03911019E __X POKER
03911021E ___ DONKEY'S TAIL
03911023E __X SPACE GEMS
03911024E ___ SCORE FOUR

04911025E ___ STARTREK 2
04911026E __X CHICKEN HELPER
04911027E ___ CATAPULT
04911031E ___ ISOLA
04912001E ___ SQUARES
04912002E ___ ECHO CHAMBER
04921001E ___ COLORMATCH
04921002E ___ TEST TUBE
04921003E __X RELATIVE IQ TEST
04921004E ___ ALGEBRA
04921005E ___ PROJECTILE PROBL.
05912003 __X SEA AIR MISSILES
05962001 __C A STRAUSS WALTZ
05962002 ___ NEVER ON A SUNDAY
05962003 __X BERCEUSE
05962004 _XC BEETHOVEN #5
05962005 __X SERENADE
05962006 +DX AMAZING GRACE
05962007 ___ BEETHOVEN OPUS 27
05962008 +DX TIME IN A BOOTLE
05962009 +DX LIGHT UP MY LIFE
05962010 __N BUMBLE-BOOGIE
06211001E __C GEN. STATISTICS
06301001E ___ MATRIS INV/MULT.
06901003E __X TOTAL DISK KAT.
06901004E ___ DISK KAT M PRINT
06911032E ___ LAST ROBOT
06911033E ___ ANTI AIR GUN
06911034H __X L-GAME
06912004E ___ SLALOM
06912005E ___ KILLER
06962011 _+D FIDDLER ON ROOF
06962012 ___ BACH #3 MINUET
06971002E ___ KALEIDOSCOPE
06972001 ___ THE PINK PANTHER
06972002 ___ SINE WAVE
06972003 __P LOVE!
06972005 __X SPRITE DEMO #1
06972006 ___ SWEETHEART

07141001H __X ANNUITIES
07901005H ___ MORSE
07902002 __P BANNER (BIG TEXT)
07911035H __X WARI
07911036H ___ LABYRINTH
07911037H __X ELIZA
07911038H ___ SECRET THOUGHTS
07911039H __X CATCH THE CHAR.
07911040H __X BLACKJACK
07911041H __X SPACE BATTLE
07911042H __X MAGIC WORD
07911043H ___ JACKPOT
07911044H ___ LABYRINTH 2
07921007E ___ PAST TENSE
07962013 ___ LOOKING THROUGH U
07971003H ___ BIG CHARACTERS
08901006E _XE UNIV. TITLE PAGE
08902003 ___ LARGE CHARACTERS
08911039E __X SPRITE CHASE
08911046E __X CHECKERS
08911049E __X SWORDS & SORCERY
08911051E __X EGGWARS
08961001E __X KILLING ME SOFTLY
08962014 __X CHOPIN OP 10 #3
09291001H ___ PROBABILITY
09391003H ___ DIFFERENTIAL EQ.
09911052E ___ DEEP SPACE
09911053F ___ ATOMES
09911054H ___ PLANETARY LANDER
09911056H __X LUNAR LANDER
09911057F ___ TOWERS OF HANOI
09911058E ___ BREAKOUT
09911059H __J DRAWING ON SCREEN
09911060H _JX TREASURE HUNT
09912008F __X ASTEROID
09912009 ___ BATTLE STAR
09991003F ___ BIORYTHM
10392002 ___ PRIME NUMBERS
10901007H ___ LOTTO (HOLL.)
10901008E ___ VERY LARGE CHAR.
10901009E ___ KEY DEFINER 99/4
10901010E __C TEXTPRO (99/4)
10911061H __X KERMIT
10911062H _JX FIND THE GUN
10911063E ___ CHASE A TARGET
10911064H ___ RUSSIAN ROULETTE
10911065E ___ CAMEL
10921008E ___ COLOR FRACTIONS
11901011S ___ STAPELDIAGRAM
11911055S __X HANGNING
11917001S _JX INKRAEKTARNA
11917002S _JX PAERON