LEDAREN

ABC-KLUBBEN ÖPPNAR SIG MOT MS-DOS KOMPATIBLA DATORER

FULL FART FRAMÅT ! ! !

ABC-klubben har vind i seglen! Jag talade just med kansliet och hörde att medlemsinbetalningarna strömmar in för fullt nu, så om Du glömt sända in årsavgiften så skynda på och skicka in den. Det är ännu för tidigt att se om vi även i år kommer att ligga kvar på samma nivå som under föregående år, men eftersom det är just nu som medlemmarna verkligen behöver klubben så är jag optimistisk. När vi i styrelsen utformade budgeten för 1987, tog vi hänsyn till en viss stagnering i tillväxten och vi räknade även med den möjligheten att vi kunde minska en del. Budgeten blev därför tämligen stram och utan några större utsvävningar. För att behålla medlemsförmånerna oförändrade och mot bakgrund att en minskning av medlemsantalet kan vara möjlig, var det därför nödvändigt med en höjning av medlemsavgiften till 190:- för detta år. Vi drabbades ju av en del oförutsedda kostnader under 1986, i form av moms på kanslikostnaderna, samt även en kraftig portohöjning på föreningspost. Vi kunde dock, tack vare en bra försäljningsökning på flexskivorna klara dessa ökade kostnader inom budgeten. För det kommande året beräknar jag en lungare kostnadsutveckling, och vi har redan nu beslutat om oförändrad årsavgift för 1988. Skulle öppningen mot MS-DOS-världen visa sig lyckosam och medlemsanslutningen öka, så finns det möjligheter till positiva överraskningar när det gäller medlemsförmåner redan detta år.

ÅRSMÖTET ÄNDRAR STADGARNA.

Årsmötet gav styrelsen i uppdrag att till kommande årsmöte komma med ett stadgeändringsförslag. Det innebär att det även i klubbens stadgar skall stå i klartext att klubben även är öppen för andra datorer. I de livliga debatter som pågått i klubben det senaste året, i MSG och bladet, i styrelsen, på styrelsens ideologimöte i höstas, och nu senast på ABC-klubbens årsmöte har ett stående tema framkommit, nämligen att öppna klubben för MS-DOS. Debatten har dock endast försökt slå in öppna dörrar, eftersom klubben redan är öppen för medlemmar med andra datorer och de allra flesta har den uppfattningen att vi skall satsa på MS-DOS. Nu är det så att vi i styrelsen inte satsar på någon speciell dator, utan vi försöker efter bästa förmåga möjliggöra för medlemmarna att utbyta erfarenheter av skilda slag. Men vi har konstaterat att många medlemmar har både MS-DOS och ABC och många säljer sina ABC och skaffar IBM. Detta har inneburit att vi redan fått in en hel del erfarenheter i form av artiklar och program avseende IBM/PC och IBM/PC-kompatibler med MS-DOS från våra medlemmar. Detta medför genast att vi får en skyldighet att inom ramen för våra resurser göra dessa erfarenheter tillgängliga för alla medlemmar i form av artiklar i bladet eller via programbanken. Jag vill också i detta sammanhang påpeka att detta inte sker på bekostnad av utskick och förmåner som avser ABC, utan som jag ser det i dag kommer vi under många år framåt att se det som klubbens främsta och mest ansvarsfyllda uppgift att underhålla erfarenhetsutbytet för ABC, ABC datorernas framtid är - ABC-klubben - Jag instämmer också i den uppfattning som flertalet medlemmar har i MS-DOS-frågan. Vi måste nämligen vara öppna för andra datorer annars kommer klubben snabbt att minska i storlek, få sämre ekonomi och då skulle vi få svårigheter att serva de medlemmar som blir kvar.

VÄLKOMMEN MED MS-DOS!

Gå ej ur klubben för att Du bytt dator! Sänd oss en rad till bladet eller en programsnutt, vi är väl rustade. Jag har under året fått flera personliga brev från medlemmar som skrivit och tackat för den trevliga tid som de varit medlemmar och menat att det är synd att de måste lämna klubben nu när de skaffat en annan dator. Jag har då kunnat lämna det glädjande beskedet att vi redan har många program att bjuda på för MS-DOS så de kan lungt stanna kvar som medlemmar.

FÖRDUBBLAD PROGRAMBANK !!!

Programverksamheten har varit livlig under året. Programbanken har för ABC-80 och ABC-800 ökat kraftigt med minst ca 500 filer till ca 12-15 Mb, och inte nog med det, dessutom har tillkommit ca: 18 Mb IBM/MS-DOS fria programvaror av hög klass. Detta har till stor del skett genom Kent Berggrens försorg. I syfte att ta fram en komplett programkatalog har, i samarbete med ABC-Stockholm, ett stort katalogiseringsarbete påbörjats för samtliga program i programbanken. Meningen är att dumpa alla bibliotek på disketter och på detta sätt göra programbanken åtkomlig för alla medlemmar, även de som inte har modem. Programdisketterna skall därefter kunna beställas till självkostnadspris, troligen 50:- per diskett. Klubben har dessutom fått och bytt till sig flera professoinella programpaket för både ABC-80 och ABC-800 dels från medlemmar och dels från DIAB, SKANDIA METRIC, CAT m fl. Programmen säljs till självkostnadspris med manualer. (se klubbens annons på omslagets sida 3)

ABC-DISK

På årsmötet hade jag förmånen att meddela styrelsens gratifikation till medlem <1019> Anders Olsson för hans magnifika insats med programmet "ABC-DISK" som verkligen är en nödvändig länk mellan ABC och MS-DOS. Man får verkligen förundra sig över varför ingen fabrikant varit tillräckligt ansvarsmedveten eller affärsmässig att lösa problemet, se artikeln intill! Jag vill också tacka Anders Olsson för den rätt som medlemmarna i klubben fått, att för egen del fritt disponera ABC-DISK Ver: 0.97. Denna versionen är en utveckingsversion med i stort sett samma prestanda som det färdiga programet 1.0 som annonseras på sidan 13 i detta blad. Priset för ABC-medlem är 875:- Som jag har erfarit, är skillnaden i huvudsak den, att 1.0 även klarar Q formatet eller 80 spårs 832 Luxorformat. Vidare så går inte 0.97 köra på alla IBM-kompatibla datorer, vilket 1.0 gör. Eftersom klubbens medlemmar fritt får använda (inte sälja) 0.97 rekomenderar jag Dig först beställa ABC-DISK från klubben för 50:- (då följer också KERMIT för MS-DOS med) prova den på din IBM kopia, om programmet inte fungerar så har Du i alla fall fått en KERMIT. Sen kan Du ju beställa Ver:1.0 enligt annonsen.

ABC-KASSETT/DISKETT

Kassett 20 och 21 är på väg att kopieras nu, de har varit färdigproducerade en tid men vår masterbandspelare har inte fungerat. Vi har därför tvingats bygga om en Revoxbandspelare för ändamålet och det har tyvärr tagit lite tid, men den skall fungera nu.

VÄRVA NYA MEDLEMMAR !!!

För att vår förening skall utvecklas och vara stark är det viktigt att Du känner ansvar som föreningsmedlem och värvar nya medlemmar. Du får snart en värvningsbrochyr som vi vill att Du delar ut till någon som har nytta av ett medlemskap. Kansliet hjälper Dig om Du behöver fler.

<872> Stig Löfgren

Kallelse

till extra föreningsmöte i ABC-klubben

Tid: Onsdagen den 87-04-29 Klubblokalen Vidängsvägen 1, ALVIK. 161 33 BROMMA

Förslag till dagordning

1. Val av mötesordförande.
2. Val av mötessekreterare.
3. Val av två justeringsmän, tillika rösträknare att jämte mötesordföranden justera protokollet.
4. Fråga om mötet är behörigen utlyst.
5. Fastställande av dagordning.
6. Föredragning av de sedan årmötet 1987-02-28 bordlagda redovisningshandlingarna.
7. Föredragning och godkännande av revisionsberättelse.
8. Fråga om ansvarsfrihet för styrelsens medlemmar.
9. Fastställande av balansräkning.
10. Avslutning.

ABCDISK

En länk mellan ABC och PC

Så sent som 1984 gick Luxor Datorer ut med annonser där man erbjöd sig att tjänstgöra som tolk närhelst olika maskiner från de stora datortillverkarna behövde utväxla information. Därför är det smått ironiskt att det bara två och ett halvt år senare inte är alldeles självklart att två olika modeller från samma tillverkare smärtfritt ska kunna utbyta data.

1985 kom EriBasic ut på marknaden, och Ericsson var därmed först på plan med en "ABC-kompatibel" BASIC-interpretator för PC-datorer. Nu skulle det gå att byta upp sig till en modernare dator samtidigt som man bevarade värdet av sin gamla programvara. Men hur skulle det gå till att fysiskt flytta programmen till MS-DOS miljö? Den sidan av saken togs över huvud taget inte upp i bruksanvisningen till EriBasic. Inte heller DIAB hade någon lösning i sikte vid utgivningen av BasicII/PC.

En möjlighet är naturligtvis att koppla samman de två datorerna med ett nollmodem, dvs en klurigt kopplad kabel mellan serieportarna, och att därefter föra över data med något protokoll som finns för båda maskinerna. Det blir i praktiken bara Kermit man kan använda. Men metoden tilltalade mig inte särskilt mycket. Dessutom har jag själv en ABC80 som det (vad jag vet) inte finns någon Kermit till. Då föreföll det mig betydligt attraktivare att med ett program läsa ABC-maskinernas disketter direkt i en PC. Jag var smått förvånad över att ett sådant program inte levererades med EriBasic, och ännu mer förvånad över att det vid den tidpunkten (våren 1986) inte fanns något sådant program på marknaden över huvud taget. Är det kanske inte tekniskt möjligt att hantera ABC-format i en PC? Det måste givetvis undersökas!

Några veckor senare var prototypen till programmet "ABCdisk" (skriven i EriBasic) färdig. Ganska snart kom jag emellertid underfund med att EriBasic - och även BasicII/PC - lämpar sig mycket väl för gamla program som ska köras i en PC men inte så väl för nyutveckling av program. Därmed inte sagt att det skulle vara något fel på EriBasic eller BasicII/PC, men båda dessa språk begränsar möjligheten att distribuera det egna programmet genom att de kräver licensierade run-time moduler till ett styckepris av 500 kronor. Om användaren själv har tillgång till respektive programspråk är detta naturligtvis inte något problem. Men EriBasic och BasicII/PC är tyvärr inte heller helt kompatibla med varandra, så ett val av antingen EriBasic eller BasicII/PC skulle i båda fallen utesluta möjligheten för användare av det andra språket att köra mitt program. Det var tydligt att det krävdes ett kompilerande programspråk, och mitt val föll på Turbo Pascal som är helt unikt när det gäller pris/prestanda.

Hårdvaran i en PC är faktiskt ganska flexibel när det gäller att hantera olika diskettformat. Till skillnad från en ABC-dator, där kontroller-kretsen hanteras av en egen mikroprocessor, är floppy-kontrollern i en PC direkt ansluten till maskinens buss. Detta öppnar stora möjligheter att både skapa och läsa diskettformat som ser ut nästan hur som helst. (Vill man göra motsvarande i en ABC-dator måste man ladda upp ett eget program i kontroller-kortets minimala RAM-minne - en synnerligen besvärlig procedur.)

Men man behöver faktiskt inte vara så brutal att man går direkt på hårdvaran. PC:ns BIOS är någorlunda väl förberedd för att hantera främmande diskettformat. Information om gällande diskettformat finns i en elva bytes lång tabell som kallas "Disk Base Table". Till denna tabell pekar interruptvektor 1EH. Så grundprincipen för att läsa ett främmande diskettformat är enkel:

Skapa en egen Disk Base Table som beskriver det diskettformat du vill arbeta med.

Ställ sedan om interruptvektor 1EH så att den pekar på din tabell.

Anropa BIOS (interrupt 13H) med begäran om läsning av önskad sektor.

Ställ slutligen tillbaka interruptvektorn till det ursprungliga värdet igen så att det åter går att läsa normala (DOS-formaterade) disketter.

Tekniken är välkänd och används bl a av otaliga kopieringsskydd.

Om man vill läsa disketter genom anrop till BIOS finns det en detalj man bör känna till, en fälla som det är lätt att trilla i beroende på PC:ns hårdvarukonfiguration. DMA-kretsen (8237) är egentligen avsedd för 8-bitars datorer och kan bara adressera minnet i jämna 64-kilobytes block. Med lite otur kan programmets buffert hamna kring en 64-kilobytes gräns. Då går det inte att läsa disketten. Om man ändå försöker kommer BIOS att rapportera felkod 9. En lösning på detta är att använda två buffertar i omedelbar följd. Minst en av dessa kommer alltid att kunna användas.

Om det går att läsa en sektor går det naturligtvis att läsa hela disketten, men den litteratur som går att uppbringa om ABC-datorernas filformat är ytterst sparsam och består ofta av vad olika personer helt enkelt har gissat sig fram till genom flitiga studier av hur det ser ut på disketterna. Min utgångspunkt var två artiklar i ABC-bladet nr 4, 1980 och nr 1, 1982 av Magnus Lundberg respektive Lars Eric Sörensen. Programmet FILSTAT.BAS (av Bo Kullmar och Kristoffer Eriksson) som finns på klubbens monitor gav en del värdefulla tips på de förändringar som introducerats i UFD-DOS. Men det fanns trots allt några detaljer i filsystemet som inte framgick av ovanstående, och där hjälpte inget annat än personliga kontakter med klubbens medlemmar.

Vilka ABC-format kan man läsa i en PC? Det bestäms av PC-datorns flexskiveenheter. Den i särklass vanligaste driven i PC- och PC/XT-datorer har under MSDOS en kapacitet av 360 kilobytes och arbetar med 40-spår per sida. I en sådan drive kan disketter med "ABC830-format" (160 kB) läsas. Vill man ha större flexibilitet måste man gå upp till en IBM PC/AT eller en med denna kompatibel maskin. I en AT-dator är standardfloppyn på 1,2 megabytes. Och eftersom den räknar till 80 spår per sida kan man läsa filer i "ABC832-format" (320 och 640 kB). Genom dubbelstegning kan även 40-spårs disketter av ABC830-typ hanteras.

Säkerligen täcker detta 90-95 procent av behovet. Kvar har vi åttatumsdisketter och det äldsta ABC80-formatet med enkel packningstäthet. I båda fallen sätter hårdvaran spärrar. Åttatumsdisketter får man kämpa rejält med om man ska få in i en 5 1/4" drive, och enkel packningstäthet kan helt enkelt inte hanteras av dataseparatorn på kontrollerkortet.

Nyttan med att kunna överföra filer mellan de två maskintyperna är naturligtvis inte begränsad till konvertering av BASIC-program. Alla som sysslar med textbehandling, databashantering, bokföring och mycket annat måste ha precis lika stort behov av att enkelt kunna flytta över data. Vidareutvecklingen av programmet "ABCdisk" fortsätter. Min förhoppning är att många med mig ska tycka att det här är en mycket trevligare metod att föra över data än med sladd och kommunikationsprogram.

<1019>
Anders M Olsson
Formansgatan 11
216 21 Malmö

Varning för blåbär!

Det ödesdigra misstaget

I decembernumret av ABC-bladet gjorde Sven Wickberg ett ödesdigert misstag i sin blåbärsartikel. Han uppmanade nämligen andra medlemmar att komma på bättre versioner av de blåbärsfunktioner han presenterade. Detta torde inte bli några som helst problem. En del av de funktioner Sven visade har fått mig att vakna genomsvettig mitt i natten och högt skrika: "Nej! Så får man INTE göra!" Efter att ha övervägt läkarhjälp har jag kommit fram till att den enda lösningen på mina sömnproblem är att prata ut; att få de lömska mardrömmarna ur mitt sinne och reda ut spagettikoden i min hjärna.

Lokalt och globalt

Låt mig börja med begreppen lokala och globala variabler. En av fördelarna med BASIC II är möjligheten att ha flerradiga funktioner med lokala variabler (det finns dock en del omständigheter som hindrar effektivt utnyttjande av funktioner, jag återkommer till detta senare). Man kan då bygga upp ett litet bibliotek med nyttiga funktioner som, genom att de använder lokala variabler, inte kan krocka med andra funktioner eller programdelar. Om vi t ex har funktionerna:

```
FNSkriv(X,Y) LOCAL I,K
```

och

```
FNRäkna(Antal) LOCAL I
```

som vi kan låtsas skriver något viktigt på skärmen respektive räknar ut något intressant, och följande programsnutt (radnummer utelämnade):

```
WHILE I<10
  Z=FNSkriv(Xpos,Ypos) : Xpos=Xpos+1

  Z=FNRäkna(R) : R=R-1
  I=I+1
WEND
```

så finns det ingen risk att den globala variabeln I, som här används i villkoret för WHILE, ska få sitt värde förstört av funktionerna FNSkriv och FNRäkna eftersom de senare deklarerar I som lokal variabel. Detta betyder naturligtvis också att ingen sammanblandning kan ske mellan det I som används i FNSkriv och det som används i FNRäkna. Variabeln är lokal i den funktion i vilken den är deklarerad.

Funktioner kontra subrutiner

När jag först såg Svens artikel trodde jag att det var detta som han tog upp - alltså att man kan bygga upp ett liten samling av nyttiga funktioner som man sedan kan MERGEa in i sitt program och att detta är mycket mer fördelaktigt än att använda sig av subrutiner. Vid närmare skärskådande av artikeln (jag ignorerade självklart varningen om att icke-blåbär skulle bläddra vidare) upptäcker jag till min oförställda förvåning att Sven i själva verket har tagit några föga avancerade subrutiner och gett dem namn genom att göra funktioner av dem. Den enda vinst man gör med detta är att man slipper skriva

```
GOSUB 1781
```

utan istället kan skriva

```
Z=FNLäspost
```

eller så.

NEJ! inte JA OCH NEJ

Håhå jaja. Låt mig ta funktionen FNJa som exempel på vad jag menar. Sven tilldelar här två globala variabler nya värden, något som man mycket sällan ska behöva göra om man använder funktioner på ett riktigt sätt. Så här såg Svens FNJa ut:

(I nedanstående och följande exempel har jag, liksom Sven, inte brytt mig om att göra INPUT idiotsäkert. I en slutgiltig applikation bör man naturligtvis inte använda BASIC-ordet INPUT över huvud taget, utan istället ha sin egen inmatningsrutin som kan klara av alla tänkbara underliga fel som kan uppstå. Så när ni kommenterar denna artikel kan ni undanta denna del i er svidande kritik.)

```
20 DEF FNJa
25   ON ERROR GOTO 25
30   INPUT Svar$
40   IF Svar$='J' OR Svar$='j' THEN
       Ja=-1 ELSE Ja=0
50   RETURN 0
60 FNEND
```

och så här borde den ha sett ut:

```
20 DEF FNJa LOCAL Svar$=20
30   INPUT Svar$
40   IF Svar$='J' OR Svar$='j' THEN
       RETURN -1 ELSE RETURN 0
60 FNEND
```

Här utnyttjar man två av funktionernas fördelar framför subrutiner, nämligen dels att man kan ha lokala variabler, dels att funktionen kan returnera ett värde. I vårt huvudprogram kan det se ut så här när man anropar FNJa (radnummer utelämnade):

```
WHILE NOT Klar
  ! Här ligger huvudprogrammet
  PRINT "En gång till (N)? ";
  IF FNJa THEN Klar=0 ELSE Klar=-1

WEND
```

De som nu inte förstått vitsen med att INTE ha globala variabler i funktioner bör ta sig en liten funderare. Tänk efter vad som antagligen händer när ens funktionsbibliotek kommer upp i tiotals, ja kanske hundratals, funktioner. Utan tvivel har man då nått den punkt då man näppeligen kan hålla reda på vilka variabler som använts var. Enligt lagen om alltings djävlighet kommer man säkerligen att använda variabeln Svar$ för att lagra undan något viktigt svar och därefter anropa Svens FNJa som obönhörligen förstör innehållet i Svar$.

Sven pekar faktiskt på möjligheten att deklarera Svar$ som lokal i slutet av sin artikel, vilket fick mig på något bättre humör. Detta spolierades dock ganska snabbt när jag fortsatte och såg att Sven åter igen talade om "ovanstående subrutiner". Ett GOSUB i ett program bör ses som ett nederlag och när funktioner endast får tjäna som namngivning av subrutiner är det ett ännu större nederlag. Det är som att använda en grävskopa att luta spåden mot.

Ännu bättre

Om man anammar konventionen att alltid markera logiskt sant med -1 och logiskt falskt med 0 (och det bör man göra eftersom det är så datorn lagrar dessa tillstånd och funktionen NOT fungerar då som man förväntar sig) kan man snygga upp ovanstående program lite. Vi definierar en ny funktion, FNSvar:

```
DEF FNSvar(stnd) LOCAL Svar$=20
  INPUT Svar$
  IF Svar$="J" OR Svar$="j" THEN
  RETURN -1
  IF Svar$="N" OR Svar$="n" THEN
  RETURN 0
  RETURN stnd
FNEND
```

stnd står för standardvärde och är det värde som ska returneras om varken J eller N skrivs in (man har då antagligen tryck enbart på RETURN; ovanstående rutin är som synes inte helt perfekt, men med en mer generell inmatningsrutin skulle den fungera tillfredställande).

Vårt huvudprogram kan nu skrivas så här (återigen utan radnummer; i princip bör man undvika alla former av radnummerreferenser):

```
Ja=-1 : Nej=0
Klar=Nej
WHILE NOT Klar
  ! Här ligger huvudprogrammet
  PRINT "En gång till (N)? ";
  Klar=NOT FNSvar(Nej)
WEND
```

I detta exempel fungerar Ja och Nej som konstanter och de bör därför deklareras för sig själv i början av programmet.

Mer funktioner

Sven presenterar andra funktioner än FNJa i sin artikel. Men även i FNNej, FNTöm$, FNHome, FNPaus$, FNPrinter och FNÖppna begår han samma misstag. Vad Sven visar är inte en samling funktioner utan en samling namnade subrutiner utan några lokala tillstånd. På detta sätt missar han totalt de fördelar som BASIC II:s flerradiga funktioner har framför subrutiner.

I FNNej sätter han de globala variablerna Svar$ och Nej. FNNej borde utformas analogt med min definition av FNJa. Eller ännu hellre skulle både FNJa och FNNej slopas och ersättas med FNSvar.

FNTöm$ är verkligen katastrofalt dålig. Här tilldelar Sven den globala variabeln Töm$ ett nytt värde, helt i onödan. Det hela är speciellt klantigt gjort då (eftersom man ändå använder en global variabel) kunde börja programmet med raden

```
Töm$=CHR$(12)
```

och sedan använda den i utskrifter i stil med

```
PRINT TÖm$ "Nu blankades skärmen!"
```

Att, som Sven rekommenderar, skriva

```
PRINT FNTöm$ "Hej hopp!"
```

innebär bara slöseri med utrymme och skrivarkraft.

De övriga funktionerna visar på samma mönster och det skär i en programmerares hjärta att se en sådan relativt bra BASIC-dialekt som BASIC II misshandlas på detta sätt. (Ja, Sven, jag sa ju att det var ett ödesdigert misstag att begära kommentarer; hoppas att du inte tar det alltför hårt!)

Anrop av funktioner

Funktioner i BASIC II kan man använda på två sätt: dels som vanliga funktioner (de tar ett eller flera värden och ger tillbaka ett resultat som sedan används till något efter anropet) och dels som procedurer. I det senare fallet är man inte inte intresserad av vad funktionen ger som resultat. Frågan är då vad man gör med detta "onödiga" resultat. Lämpligen lägger man det i en variabel som inte används, vanligen används Z men det kan lika gärna vara Dummy, Slask, A9 etc. Så här kan det se ut:

```
Z=FNÖppnafil("REGISTER.DAT")
```

Funktionen kan då avslutas med ett RETURN Z som markerar att den ska betraktas som en procedur.

Observera att procedurer inte är samma sak som subrutiner. De kan användas för likvärdiga uppgifter men man bör utnyttja möjligheten till lokala variabler för att undvika variabelkollision. Över huvud taget är globala variabler något man ska vara mycket försiktigt med.

Sven menar att ett alternativt sätt att anropa funktioner när de betraktas som procedurer är med PRINT. Detta är självklart helt förkastligt (i själva verket är det ju inte procedurer Sven visar utan namnade subrutiner, men det är i alla händelser fel). Detta leder den som studerar programmet att tro att funktionens resultat kommer att skrivas ut på skärmen och är av betydelse. PRINT ska användas då funktionens resultat ska skrivas ut (eller möjligen om man absolut vill att funktionen ska utföras mitt i en utskrift).

Variablers räckvidd

Betrakta följande programexempel och fundera över vad som skrivs ut:

```
X=10
Z=FNA
END
DEF FNA LOCAL X
  X=20
  Z=FNB
  RETURN Z
FNEND
DEF FNB
  PRINT X
  RETURN Z
FNEND
```

Är det 10 eller 20? X sätts först globalt till 10. Sedan anropas funktionen FNA där X deklareras som lokal och får värdet 20. FNB anropas sedan och där skrivs X:s värde ut. Svaret är tyvärr att 10 skrivs ut. När BASIC II skrevs resonerade man kanske som så att eftersom alla funktioner är definierade i toppomgivningen (det går ju som bekant inte att definiera en funktion inuti en annan; då hade man definierat denna i en lokal omgivning) så ska alla variabler som inte deklareras som lokala ha sitt globala värde, inte ett värde som en anropande funktion har bundit till variabeln.

Lexikalisk och dynamisk bindning

I BASIC II används alltså ett lexikaliskt räckviddsbegrepp för variabler. Vid anrop av funktioner tas det globala värdet av alla variabler som inte är formella parametrar eller definierade som lokala. Men man hindrar effektivt programmeraren att programmera lexikaliskt, dvs att placera funktioner inuti varandra och på så sätt få dessa att definieras och exekveras i en egen omgivning. Ett programspråk eller system som hindrar programmeraren eller användaren att utforma saker och ting på sitt sätt är naturligtvis inte bra. Man måste därför konstatera att trots att BASIC II är relativt kraftfull i jämförelse med andra BASIC-tolkar är det inte ett bra programmeringsspråk. Programmeringsomgivningen är dessutom ganska usel och man klarar sig i princip inte utan diverse hjälpsystem, (typ Smartaid).

Vad man självklart borde använt sig av är dynamisk bindning av variabler. I praktiken kan många sätt användas för att implementera detta men det är enklast att förklara om man resonerar kring "stackning". När en variabel ska användas som formell parameter eller deklareras som lokal tas dess aktuella värde (om variabeln redan finns och således har ett värde) och läggs på stacken. För en rutin uppträder alltså en lokal variabel precis som en global - enda skillnaden är att dess gamla värde finns undansparat någonstans i minnet, lämpligen på en stack. Vid uträde ur rutinen återhämtas värden från stacken och läggs tillbaka i variablerna.

Om således en rutin A där variabeln I är deklarerad som lokal anropar en rutin B så uppfattar B-rutinen variabeln I som vilken som helst. Man kan säga att rutin A har skapat en lokal omgivning där vissa variabler har nya värden. Om då rutin B används för att t ex sätta I till ett uträknat värde, kan olika rutiner deklarera I som lokal, anropa B och få "sitt" I satt på rätt sätt. Detta är inte möjligt i BASIC II.

Allt hopp är dock inte ute för BASIC II och det går att göra en hel del intressant. Särskilt om man anstränger sig och använder möjligheten till flerradiga funktioner och försöker strukturera sina program med WHILE och FOR.

Epilog

När man befinner sig i blåbärsstadiet är det mycket viktigt att man umgås med meloner, pumpor och andra erfarna läromästare som kan förklara för blåbäret vad som är rätt och riktigt. Det gäller då för melonerna och pumporna att inte sänka sig till blåbärets nivå utan att försöka förklara de svåra sakerna på ett lätt sätt. Steg för steg inser då blåbäret hur det ligger till och kan i sinom tid växa upp till en stolt kokospalm eller så.

Jag anser mig inte vara en fullvuxen melon och jag tror inte jag insett alla mysterier men jag måste helt enkelt protestera när man försöker vilseföra blåbär. Funktioner är funktioner och subrutiner subrutiner och de ska INTE blandas ihop. Då blir programmen en farlig soppa som är svår att torka ur skägget.

<2422>

Ulf Dahlen

Inga blåbärs-funktioner

Sven Wickberg skrev en artikel i ABC-bladet nr 4 1986 med rubriken "Blåbärsfunktioner". Enligt min uppfattning är det nödvändigt att förklara för blåbär hur man bör utnyttja flerradiga funktioner. Att som Sven enbart ersätta GOSUB rutiner med funktioner är ett typiskt nybörjarmisstag.

Finessen med flerradiga funktioner är att man kan använda lokala variabler. Det är variabler som bara finns just inom funktionen. Fördelen med detta är att dessa inte påverkar omgivningen och de tar inte heller upp något minne utom när funktionen exikveras.

Det är också möjligt att låta en funktion direkt returnera att värde utan att använda en mellanvariabel. Ibland behöver man dock en mellanvariabel för att felhanteringen med ON ERROR GOTO skall fungera som avsett.

Ett bra sätt är att låta en funktion returnera falskt eller sant dvs 0 eller -1. Anrop av en strängfunktion kan göras genom att skriva Dummy$=FNFunktion$ men då måste man komma ihåg att funktionen ej får returnera en så lång sträng att man får DIM-fel (fel 137).

Felrutinen är lokal inom en funktion.

```
10 DEF FNMedelvärde(Tal1.,Tal2.) LOCAL
   Temp
20   ON ERROR GOTO 40
30   Temp=(Tal1.+Tal2.)/2.
40   RETURN Temp
50 FNEND
```

I min version av FNMellanvärde använder jag en mellanvariabel som jag kallar för Temp. Den är definierad som lokal genom "LOCAL Temp". En ON ERROR GOTO i funktionen ser till att värdet noll returneras om fel uppstår. Detta beror på att den lokala variabeln då inte har tilldelats något värde. Alla lokala variabler initieras till 0 respektive tom sträng för strängvariabler. Denna initiering sker varje gång en funktion anropas om variabeln är lokal.

Jag har jag skrivit om Svens program INPUT. Problemet med PF-tangenter och INPUT löses enklast genom att skriva en egen FNInput$ funktion. Jag har gjort en mycket enkel sådan för detta exempel. I ABC-klubbens programbank finns det flera exempel på mera avancerade INPUT-funktioner.

```
10 ! SAVE INPUT.BAC
15 INTEGER : EXTEND
20 DEF FNJa LOCAL Svar$=160
30   Svar$=FNInput$
40   IF (ASCII(Svar$) AND 223)=74 RETURN
     -1 ELSE RETURN 0
50   !
60 FNEND
```

Funktionen FNInput$ returnerar svaret till en lokal variabel som jag kallar för Svar$. Denna lokala variabel är lokal och därför måste den dimensioneras. Funktionen FNInput$ är gjord så att den kan ta emot maximalt 160 tecken därför är Svar$ dimensionerad till detta. Svar$ tar enbart upp minnesutrymme i datorn så länge funktionen exikveras därför behöver man inte snåla med längden.

Om J returnerar funktionen FNJa sant värde, dvs -1 annars returneras falskt värde, dvs 0. Genom att ta ASCII-värdet som man matar in och skriva AND 223 maskar man mycket enkelt bort små bokstäver så att det fungerar om man matar in både "j" och "J".

I detta fallet testat bara på första bokstaven vilket brukar vara fullt tillräckligt.

```
   J (=ASCII  74) binärt 01001010
   j (=ASCII 106) binärt 01101010
   ==============================
   summa     223  binärt 11011111
```

AND 223 innebebär att bit 6 (räknat från höger till vänster med 0 som första) nollställes. Matar man in "j" som har ASCII-värdet 106 får man efter AND operationen ut 74. I början på Luxors BASIC-manual kan Du läsa lite mera om bithantering.

```
100 DEF FNNej LOCAL Svar$=160
110   Svar$=FNInput$
120   IF (ASCII(Svar$) AND 223)=78 RETUR
      N -1 ELSE RETURN 0
130   !
140 FNEND
```

FNNej följer samma mönster som FNJa.

```
180 DEF FNTöm$
190   RETURN CHR$(12)
200   !
210 FNEND
```

Någon mellanvariabel behövs inte för FNTöm$ utan man kan med fördel direkt returnera det aktuella värdet, dvs CHR$(12).

```
250 DEF FNHome$
260   RETURN CUR(0,0)
270   !
280 FNEND
```

Jag har här ändrat FNHome till FNHome$ dvs den blir en strängfunktion. Detta gör att man kan anropa den med ;FNHome$ vilket jag tycker är naturligare. Att detta går beror på att CUR i BASIC i verkligheten ger en sträng som ger den önskade effekten på markören på ABC:ens bildskärm. Det är i själva verket frågan om en markörstyrningssekvens för en ADM3A terminal.

```
310 DEF FNPaus$ LOCAL Dummy$=1
315   ; 'Tryck på valfri tangent för for
      tsättning! ';
320   GET Dummy$
330   RETURN CHR$(13)+SPACE$(50)+CHR$(13
      ,10)
340 FNEND
```

GET i FNPaus$ använder nu en lokal variabel. Eftersom själva värdet i variabeln inte används har den fått ett namn som antyder detta. Lokala strängvariabler måste definieras till sin längd, i detta fall ett tecken eftersom GET satsen här bara läser in ett tecken.

Jag har också lagt till en ledtext som förklarar att man skall trycka på en tangent för fortsättning. Detta för att göra funktionen mera användarvänlig. När man har tryckt på en tangent suddas ledtexten ut genom att funktionen returnerar en sträng som först flyttar markören till vänsterkanten på bildskärmen och sedan skriver över den gamla texten med blanktecken och avslutar med en ny rad, dvs CR för att flytta markören till vänstekanten och LF för ny rad.

```
370 DEF FNPrinter LOCAL Prnotok,Printer$
    =16,Svar$=160
375   Prnotok=-1
380   ; FNTöm$;
390   ; 'Skall printer användas (J) ';
400   !
410   IF FNNej THEN RETURN 0
420   !
430   Printer$='PR:VSA30C72.5'
440   !
450   WHILE Prnotok
460     ; : ; 'Vad heter printern (' Pri
        nter$ ') '; : Svar$=FNInput$
470     IF Svar$='' THEN Svar$=Printer$
472     Svar$=FNVersal$(Svar$)
474     IF INSTR(1,Svar$,'PR:')=0 Svar$=
        'PR:'+Svar$
480     Prnotok=FNPrinterparok(Svar$,9)
490   WEND
500   !
510   RETURN FNÖppna(Svar$,9)
520   !
530 FNEND
```

FNPrinter hade jag lite svårt för att förstå i Svens version. Jag har skrivit om den och utnyttjar där FNNej direkt i IF-satsen, dvs IF FNNej THEN RETURN 0. Detta är mycket bättre än att utnyttja en global variabel i FNNej för att testa på. En mellanvariant hade varit att skriva Nej=FNNej och sedan IF NEJ THEN ... Nej bör då också vara en lokal variabel i FNPrinter.

För att kolla om man har angett en giltig printerparameter om man inte använder den som programmet föreslår har jag skrivit en funktion FNPrinterparok. Denna funktion provöppnar printern och kollar om det blir fel. Funktionen returnerar falskt eller sant till den lokala variabeln Prnotok. Initialt är denna variabel satt till sann, -1, så att programmet kommer in i WHILE-satsen minst en gång.

När detta är klart sker uthopp hur funktionen direkt med RETURN FNÖppna(Svar$,9). Man behöver alltså inte anropa funktionen med Z=FNÖppna och sedan skriva RETURN 0. Notera också att parametrar skickas med anropet och därmed blir dessa lokala.

```
560 DEF FNÖppna(Pr$,Lu)
570   ; FNTöm$;
580   ; 'Är printern påslagen? ';
590   !
600   OPEN Pr$ AS FILE Lu
610   ; $Lu,CHR$(13,27,16) ! Sven har CH
      R$(13);
611   ! här men det fungerar ej på
612   ! FACIT4510 eftersom den skrivaren
      vill
613   ! ha CR LF för att skriva ut en ra
      d.
614   ! I detta fall skrivs alltså en to
      m rad
615   ! ut och sedan skickas en
616   ! sekvens till FACIT 4510 för att
      backa en rad.
620   ; CHR$(13) TAB(25) CHR$(13);
630   RETURN 0
640 FNEND
```

FNÖppna har jag inte ändrat alls, om man undantar mitt exempel på anpassning av funktionen till FACIT 4510. Svens funktion fungerar säkert bra på hans skrivare.

```
670 DEF FNPrinterparok(Pr$,Lu)
680   ON ERROR GOTO 700
690   OPEN Pr$ AS FILE Lu : CLOSE Lu :
      RETURN 0
700   RETURN -1
710 FNEND
```

Funktion för att kolla om printerparametern är ok. Om så ej är fallet återvänder fuktionen via ON ERROR GOTO 700 och RETURN -1 annars RETURN 0. Notera att Pr$ och Lu är lokala variabler utan att man behöver definiera dem som LOCAL. Inparametrar blir autmatiskt lokala.

```
740 DEF FNVersal$(In$) LOCAL Y,Ord$=160
750   Y=1 : Ord$=In$
760   WHILE Y<=LEN(Ord$)
770     IF ASCII(MID$(Ord$,Y,1))>95 AND
        ASCII(MID$(Ord$,Y,1))<127 THEN M
        ID$(Ord$,Y,1)=CHR$(ASCII(MID$(Or
        d$,Y,1)) AND 223)
780     Y=Y+1
790   WEND
800   RETURN Ord$
810 FNEND
```

Detta är en funktion för att göra om små bokstäver till stora dito. Samma AND 223 som beskrivs ovan används. En lokal sträng som är en inparameter kan ej ändras därför används en lokal sträng, här kallad Ord$. Detta beror på att det bara är referensen till värdet som skickas med anropet. Man kan under vissa omständigheter ändra en sträng som är inparameter i en funktion om man inte ändrar längden. Det är dock inte god programmerarsed att göra så.

```
830 DEF FNInput$ LOCAL Tkn$=1,Str$=160
835   GET Tkn$
840   WHILE Tkn$<>CHR$(13) ! Läs tills R
      ETURN matas in
845     WHILE Tkn$=CHR$(24) ! Blanka fäl
        tet om CE/CTRL-X
850       WHILE Str$<>''
855         Str$=LEFT$(Str$,LEN(Str$)-1)
            ; ; CHR$(8,32,8);
860       WEND
865     IF 0 WEND
870     WHILE Tkn$=CHR$(8) ! Bakåtpil,
        sudda ett tecken
875       Tkn$=''
880       Str$=LEFT$(Str$,LEN(Str$)-1) :
          ; CHR$(8,32,8);
885     IF 0 WEND
890     IF LEN(Str$)<160 IF Tkn$>=' ' AN
        D Tkn$<='Ö' Str$=Str$+Tkn$
          : ; Tkn$; ELSE ; CHR$(8,32,8);
895     GET Tkn$
900   WEND
905   ;
910   RETURN Str$
915 FNEND
```

Detta är ett enkelt exempel på en INPUT-funktion som fungerar på ungefär samma sätt som INPUT i BASIC med den skillnaden att den klarar PF-tangenter. Ofta skriver man mera avancerade INPUT-funktioner som utnyttjar vissa PF-tangenter.

```
920 ! ========================
1000 ! Demonstrationsprogram
1010 WHILE -1
```

En WHILE-loop som skrivs som WHILE -1 är alltid en sann och det är alltså en evighetsloop. Det är ett sätt att skriva ett program utan GOTO vilket gör programmet mera lättläst.

```
1020   ; 'Vill du tömma skärmen (N) ';
1030   !
1040   IF FNJa THEN ; FNTöm$;
1050   !
1060   ; FNPaus$
1070   !
1080   ; 'Vill du flytta "hem" markören
       (J) ';
1090   !
1100   IF FNNej THEN ; FNHome$;
1110   !
1120   ; FNPaus$
1130   !
1140   ; : ; 'Detta är en rutin som inte
       får utföras'
1150   ; 'av misstag. Därför skall inget
       annat '
1160   ; 'godkännas än J/j för Ja.'
1170   ; : ; 'Skall denna ödesdigra ruti
       n verkställas (N) ';
1180   !
1190   !
1200   WHILE FNJa
1210     !
1220     ; : ; 'Alldeles säker (N) ';
1230     !
1240     IF FNJa THEN ; FNTöm$;'Rutinen
         är verkställd'
1250   IF 0 WEND
```

IF 0 WEND är ett trick att avsluta en WHILE sats så att programmet alltid faller igenom. En sådan WHILE sats exikveras alltid en gång och det är egenligten en ersättning för en flerradig IF-sats av typen IF villkor, satser..., IFEND.

```
1260   ; 'Var det något mera (N) ';
1270 IF FNJa THEN WEND
```

IF FNJa THEN WEND innebär att funktionen går "runt" om FNJa returnerar ett sant värde.

```
1300 ; FNTöm$;
1310 !
1320 Dummy=FNPrinter
1360 !
1370 ; FNPaus$
2000 ; FNTöm$;
2010 END
```

I övrigt har jag ändrat vissa anrop. Jag använder ofta Dummy som anropsvariabel i stället för Z som Sven använder. Det spelar inte så stor roll vilket man använder eftersom de flesta förstår att Z har samma funktion som Dummy, dvs ingen funktion alls utom att returnera ett värde som man inte använder.

Notera att det inte finns några GOTO i programmet om man undantager felhanteringen. Utan GOTO blir programmet mera lättläst och det är lättare att ändra programmet. Det finns inte iheller så många globala variabler. Det är främst anropsvariabelen som är global.

Man bör när man har skrivit ett program använda ett program typ DIAB:s CROSS som ger information om ett programs variable[illegible]ROSS följer med om man köper SQUEZ, ett program för att komprimera BAC-kod. CROSS ger en lista över alla variabler som används såväl lokala som globala. Finns det några variabler som ej används eller ej tilldelas varnar CROSS.

Mitt program VARLIST som har gått ut på kassett/diskett kan användas för att lista globala variabler som finns i ett program. Målsättningen bör vara att man skall ha så få globala variabler som möjligt eftersom detta underlättar förändringar av programmet och tar mindre minne i anspråk.

<1789>
Bo Kullmar

Vad suckar häcken

eller

De Stora Grabbarnas Klagan

Tvenne lagar styra detta liv.
Förmågan att begära är den ena
tvånget att försaka är den andra.

Stagnelius

Nu har De Stora Grabbarna uttalat sig. Vi blåbär bugar och tackar.

Men man kan inte ha allt på en gång. Den enda undervisning jag tidigare fått i ärendet var Bo Kullmars artikel på sid 17 i Bladet 1984:1. Där talar han på 16 rader om hur bra det är med flerradiga funktioner i stället för GOSUB. DEN saken fattade jag i alla fall och har haft STOR nytta och glädje av. Men det må ursäktas mig om jag inte var i rätt sinnestillstånd för att kunna tillgodogöra mig alla andra finesser på samma gång.

I varje fall har Bosse nu fått tillfälle att utveckla pedagogiken, och jag bugar mig ännu en gång (extra djupt - syns det?) och utan alla ironiparenteser.

En del av de finurligheter som visas upp härintill är så finurliga att jag inte heller nu hänger med så där på en gång. Det problemet är jag kanske inte ensam om. Men hav tröst alla andra blåbär: Även om De Stora Grabbarna vaknar kallsvettiga över att vi inte begriper programmeringens alla finesser (för det gör vi ju inte - då hade vi varit en av dem, alltså!), är faktiskt även litet mindre smarta rutiner rätt användbara.

Om man bara skall gräva ned några tulpaner gör det ju inte så mycket om man använder en spade i stället för grävskopan som står bredvid.

De blåbärsrutiner jag visade i förra numret är pålitliga, fungerar och leder inte till några olyckor. Att de kan förbättras är en annan sak. Meningen var ju att ge blåbären några begripliga tips på vägen.

DET är kanske största fördelen med BASIC i olika varianter som programspråk: även blåbär kan få till fungerande program relativt lätt.

Därmed inte sagt att vi inte behöver förkovra oss. Det gör vi ju alla, inte minst genom att noga studera Bladet och ta till oss alla visdomsord från både STORA och små.

Med ytterligare en tacksam bugning till Bo, Kristoffer och alla andra Stora Grabbar som vinnlägger sig om vår förkovran och med vänlig hälsning till alla andra grabbar och tjejer.

<1384>
Sven Wickberg

ABC800 nedlagd - vad händer nu?

Göran Engelbo, VD för ABC DATA sedan 1979, skriver i denna artikel om hur situationen på mikrodatormarknaden i Sverige förändras och utvecklas nu när ABC800 lagts ned.

Bakgrund

Luxor och DIAB hade en gång ett järngrepp om mikrodatormarknaden i Sverige. Marknadsandelen låg som mest på nära 70%! Inget annat företag har någonsin varit uppe i liknande siffror. Förklaringen till att man den gången lyckades så enormt bra, är att man hade en överlägsen produkt som dessutom var svensk. ABC80 och ABC800 var marknadens bästa datorer till marknadens bästa priser! Under den här perioden bestod konkurrensen av föga imponerande konstruktioner från Commodore, Radio Shack och Apple.

Commodore PET var en gräslig historia med en långsam 6502-baserad dator, en liten halvtaskig bildskärm, ett långsamt kassettminne och tidernas sämsta tangentbord sammanbyggt till en enda klumpig enhet. En historisk bragd genomfördes av Mats Gabrielsson på Datatronic som faktiskt lyckades sälja några stycken!

Apple II var bättre. Den hade ingen bildskärm alls. Sådan fick man köpa separat. Endast versaler visades. Datorn var 6502-baserad och hopplöst primitiv. Stöd för flexskivehantering fanns inte från början, utan hade lagts till i efterhand som en klumpig halvmesyr.

Radio Shack hade kunnat bli ett tungt märke. Deras dator TRS80 hade en överlägsen processor, Z80, och en betydligt bättre konstruktion än PET och Apple II. I USA var också TRS80 den dator som sålde mest. I Sverige fanns dock ingen importör!

ABC80

ABC80 var totalt överlägsen. Den var billigare än alla konkurrenter, mycket snabbare än någon annan mikrodator, hade bättre tangentbord än någon konkurrent, och den hade överlägsna expansionsmöjligheter. Dessutom var den svensk. Succen var given!

Luxor och DIAB lutade sig tillbaka och njöt av framgången. 1981 hände emellertid något. 16-bitars datorer började dyka upp. Först var Victor med sin 9000 (Sirius.I hette den på den tiden). En vecka senare kom IBM med sin PC. Dessa datorer var de första konkurrenter till ABC80/800 som var tekniskt överlägsna. I början var de underlägsna i pris/prestandaförhållande, och LUXOR/DIAB tog ingen större notis om det ändrade läget, speciellt som det dröjde ända till 1983 innan IBM PC kom till Sverige.

Priset på IBM PC sjönk stadigt medan priset på ABC-maskinerna stod stilla. Dessutom började IBM:s dator alltmer att accepteras som standard på marknaden. PC-kompatibla datorer blev ett begrepp och satte ytterligare press på priserna. Den slutliga smällen utdelades av Mats Gabrielsson (ja, just han som lyckades sälja några PET!) när han köpte Victor i USA och började tillverka PC-kompatibla datorer till sensationella priser. Plötsligt var ABC800 inte bara tekniskt underlägsen, den hade dessutom ett sämre pris.

Resten vet vi. DIAB försökte kontra med att gå ett steg längre än IBM. ABC1600 var på nytt tekniskt överlägsen, men den var för dyr. Och dessutom var nu IBM:s standard helt dominerande. Luxor gick till botten.

ABC1200

Det sista Luxor Datorer gjorde var att presentera ABC1200. Denna dator är egentligen ingen ABC-dator. Den är helt IBM-kompatibel, och heter egentligen Nokia ASC. Den tillverkas i Finland av Nokia, Luxors ägare sedan ett år tillbaka.

När Luxor datorer nu lagts ned försvinner väl t o m Luxor-märket. Nokia ABC1200 är ett faktum.

ABC1200 är en relativt dyr högkvalitativ dator i högsta prestandaklass. Den kan p g a sitt pris verka mindre intressant, men efter en närmare bekantskap kan man bara konstatera: den är fantastisk! Den slår alla konkurrenter på så gott som samtliga punkter.

ABC1200 är en PC/AT-kompatibel dator, och vi förflyttar oss med den över i en delvis annorlunda omgivning än vår gamla trygga ABC-värld.

Hur ser denna värld ut?

Hårdvaran i PC-världen

1981 presenterades IBM PC. Denna dator är baserad på Intels mikroprocessor 8088 som är en förkrympt variant av deras första 16-bitars processor 8086. 8088 är, till skillnad från 8086, inte en äkta 16 bitars processor, eftersom databussen bara har 8 bitar. Anledningen till att man valt att använda en sådan halvmesyr, är att kringkretsarna runt mikroprocessorn då kunde plockas från 8-bitarsvärlden till väsentligt lägre priser än som skulle varit fallet med en äkta 16-bitars processor. Adressbussen på de här processorerna har 20 bitar och således kan 1 M byte adresseras. Detta skall jämföras med de 64 kB som 8 bitarsdatorerna, som PC skulle konkurrera med, klarade av. På den här tiden var primärminnen på 1 M byte otänkbara. Gränsen var teoretisk, inte praktisk. IBM PC levererades t ex i början med ett primärminne på 64 kB. Detta ansågs tillräckligt på den tiden!

I ABC80 kan primärminnet inte vara större än 32 kB, trots att processorn kan adressera 64 kB. Detta beror på att resterande 32 kB är upptagna av systemet (16 kB Basictolk, 8 kB reserverat som använts till 80-teckens bildminne och SuperSmartAid, 4 kB DOS, 1 kB IEC-option, 1 kB reserverat, 1 kB printerrutin och 1 kB 40-teckens bildminne).

En liknande situation gäller för IBM PC. Här är det maximala primärminnet 640 kB. Övriga 384 kB används för text och grafikminne, ROM, I/O, etc.

Mjukvaran i PC-världen

I början presenterades IBM PC med två konkurrerande operativsystem, CP/M-86 från Digital Research och MS-DOS från MicroSoft. Digital Research var vid den här tiden det överlägset största programvaruföretaget. CP/M-80, företagets operativsystem för 8-bitars datorer (8080, 8085 och Z80), var helt dominerande på marknaden. CP/M hade en liknande ställning som IBM PC/MS-DOS har idag. Det var naturligt att förvänta sig att CP/M-86 skulle vinna dragkampen om att bli det dominerande operativsystemet för IBM PC. Ganska snart visade det sig emellertid att MicroSoft skulle vinna. MS-DOS 1.0 hade segrat!

MS-DOS

MS-DOS var ett primitivt och enkelt operativsystem som dock betraktades som ganska bra när det kom. Anledningen var att man jämförde med CP/M 2.2 som drogs med ett jätteproblem: 64 kB-barriären som gjorde det omöjligt att få plats med ett vettigt operativsystem. MS-DOS kunde breda ut sig på ett helt annat sätt.

Endast flexskivor kunde i början hanteras. Till en början använde man enkelsidiga skivor på 160 kB. Ganska snart övergick man till dubbelsidig lagring och nådde då en kapacitet på 320 kB (samma format som DataDisc 84 för ABC80; 48 tpi, 8 sektorer om 512 byte per spår (16 sektorer om 256 byte på ABC80), dubbelsidig lagring med dubbel densitet). Alla filer på en skiva låg i samma bibliotek. Directorystrukturer kunde inte hanteras.

I olika omgångar växte såväl IBM PC som MS-DOS. IBM PC/XT gjorde hårddisken till ett vanligt sekundärminne. MS-DOS kom i en ny version som hanterade detta. MicroSoft kom på att man kunde öka kapaciteten på flexskivorna genom att pressa in 9 i stället för 8 sektorer på varje spår, utan att göra några hårdvaruändringar. En ny version av MS-DOS fixade detta. Resultatet blev 360 K per flexskiva, det vanligaste formatet idag. En annan viktig förbättring var att directorystrukturer tillfördes operativsystemet.

MS-DOS har med tiden utvecklats till ett fullvärdigt operativsystem för IBM PC och kompatibla 8088-baserade datorer.

Utvecklingen har emellertid sprungit ifrån dessa datorer, och idag betraktar man IBM PC som långsam, och 640 K primärminne är för litet!

IBM PC/AT

1984 tog IBM ett stort steg ytterligare när man presenterade sin första äkta 16-bitars PC-dator. IBM PC/AT är baserad på processorn 80286 som har 16 bitars databuss och 24 bitars adressbuss, vilket ger möjlighet att adressera 16 M byte.

80286 är en mycket bra mikroprocessor med stor kapacitet. Den har dessutom en intressant finess: den kan emulera (efterlikna) en 8086. Kör man 80286 i "8086-mod" uppför den sig som en 8086 (16-bitarsvarianten av 8088) med den skillnaden att databearbetningshastigheten är väsentligt större (ca 3 gånger).

Med hjälp av denna egenskap hos mikroprocessorn var det möjligt för IBM att göra IBM PC/AT kompatibel mot IBM PC. Med det menar man att det skulle vara möjligt att köra program avsedda för IBM PC på IBM PC/AT.

När IBM presenterade PC/AT kom givetvis MicroSoft samtidigt med en ny version av MS-DOS som kunde hantera den nya datorn (ver 3.0). Bl a erbjöds stöd för ett nytt flexskiveformat som introducerades med IBM PC/AT. Detta format gav en kapacitet på 1.2 M byte per skiva, vilket uppnåddes med dubbel spårtäthet (96 tpi) och kvadruppel densitet. Version 3 av operativsystemet var emellertid helt inriktad på PC-kompatibilitet. 80286:an kördes enbart i 8086-mod.

Trots att man helt inriktade sig på PC-kompatibilitet - och helt bortsåg från möjligheterna att utnyttja 80286 fullt ut - lyckades IBM inte särskilt bra med ens detta. IBM PC/AT är inte IBM PC-kompatibel till mer än ca 30%! Åtskilliga av de idag förekommande PC/AT-kompatibla datorerna är mer IBM PC-kompatibla än IBM PC/AT.

IBM:s starka dominans på marknaden gjorde emellertid att dessa problem fick lösas av programtillverkarna istället för av IBM. Nya versioner av programmen skrevs om så att de kunde köras på både PC och PC/AT. På så sätt blev IBM PC/AT "kompatibel i efterhand".

IBM:s och MicroSoft:s något misslyckade koncentration på PC-kompatibilitet har fått till följd att IBM PC/AT - som är en i många stycken utmärkt dator - ännu inte kan utnyttjas till mer än en bråkdel av sin kapacitet.

De viktigaste anledningarna till att övergå PC (8088) är 640 K-gränsen (20 bitars adressbuss) och den otillräckliga databearbetningshastigheten. IBM PC/AT löser datorarkitekturmässigt dessa problem. Med dess 24 bitars adressbuss kan 16 M byte adresseras och 80286 erbjuder en jämförelsevis enorm datakraft. MS-DOS sätter dock fortfarande stopp vid 640 kB, och den högre databearbetningskapaciteten utnyttjas bara delvis. MicroSoft har naturligtvis insett det orimliga i situationen, och jobbar sedan länge med nästa generation av MS-DOS som skall spränga alla tidigare vallar. MS-DOS Ver 5.0 kommer - sägs det - att marknadsföras under namnet A-DOS (Advanced DOS) och utnyttja 80286 fullt ut, med möjlighet till direkt adressering av 16 M byte RAM. MicroSoft har haft oväntat stora problem med utvecklingsarbetet och det har därför dragit ut längre på tiden än vad som från början var avsikten. Enligt MicroSoft kommer den nya MS-DOS-versionen att släppas andra kvartalet 1987.

Då A-DOS gör entre är det dock definitivt adjöss för PC och 8088! A-DOS kräver 80286. Vi kan utgå från att det nya operativsystemet med dess ojämförligt större möjligheter kommer att tilldra sig programmakarnas fulla intresse och lämna gamla MS-DOS-versioner och PC i bakvattnet. Vi ser alltså här slutet för PC-epoken.

Datorköp idag - vad skall jag välja?

Du började med en ABC80. Så småningom gick du över till ABC800. Nu funderar du på att köpa en PC-kompatibel.

Känner du igen dig? Många gör det. Dagligen får jag telefonsamtal från folk i den situationen. Vilket råd får de?

Jag kan börja med att glädja dem med följande intressanta iaktagelse, som nog kan förvåna många.

PC/AT är idag lika kompatibel med ABC800 som ABC800 är kompatibel med ABC80!

Ett antal nya produkter har gjort detta påstående sant. Det är därför idag relativt smärtfritt att ta steget över i MS-DOS-världen.

Steg 2 är att bestämma sig för om ett byte över huvud taget är vettigt. Är PC och PC/AT så mycket bättre än ABC800 att det är mödan värt att ta steget? Svaret är ja! Datorn som sådan är visserligen inte så fruktansvärt mycket bättre att det räcker som motiv, men programvaran är det definitivt. Den programvara som idag finns tillgänglig för MS-DOS håller en kvalitet som är oöverträffad i historien. Inte för någon annan dator, inberäknat minidatorer, eller för något annat operativsystem någonsin, har det funnits så mycket och så bra programvara.

Steg 3 är att bestämma sig för PC eller PC/AT. PC kommer att ligga i dödsryckningar om ett halvår, och om ett drygt år är den död. Skall datorn skrivas av på ett år eller mindre är en PC-kompatibel dator i lägsta prisklassen ett bra alternativ. Skall datorn fungera som ett arbetsredskap i flera år framåt - och så är det nog för de flesta - är en PC/AT det enda rätta!

Steg 4 är att välja märke. Är det vettigt att köpa IBM? Nej, oftast inte! Det finns kompatibla märken som ger mer. Vill man ha en billigare maskin som har samma prestanda som IBM finns det sådana alternativ (t ex Victor V286) (*), vill man ha en dator i samma prisklass som IBM men med bättre prestanda finns det också sådana (t ex Nokia ABC1200) (*).

ABC1200 är förmodligen den bästa PC/AT-kompatibla datorn på marknaden idag. Dess enda nackdel är egentligen priset.

Hur klarar man övergången?

Hur flyttar vi oss då från ABC800 till PC/AT (t ex ABC1200)?

Ja, i stort sett på samma sätt som från ABC80 till ABC800. Program i BAC-format kan inte överföras, men textfiler går bra (t ex program i BAS-format). Flexskiveformaten är inte helt kompatibla (på samma sätt som 832- och 830-formaten för ABC800 respektive ABC80). Det finns dock bra metoder för överföring där också. De gamla standardprogrammen - som ju ligger i BAC-format - kan inte överföras. Det blir till att köpa nytt. (ABC80-program kan inte köras på ABC800 heller). Egna program skrivna i BASIC kan dock med fördel flyttas över.

Vad krävs?

För det första behöver vi en BASIC-tolk av samma slag som i ABC800. Den heter BASIC II/PC (*) och är mycket lik BASIC II/ABC800 och kräver därför bara små modifieringar av programmen (ABC-BASIC på ABC80 skiljer sig betydligt mer).

Överföringen av filer till det nya flexskiveformatet kan göras på flera sätt. Det enklaste är att lämna bort jobbet (*). Alternativ 2 är att använda en V24-sladd mellan datorerna och ta över filerna med hjälp av t ex Kermit. Den bästa metoden är att köpa programmet ABC-Disk (*). Detta program gör det möjligt att stoppa in ABC80 och ABC800 flexskivor direkt i en PC/AT och kopiera filer från dem. Version 1.0 av ABC-Disk klarar av 830- och 832-formaten. Övriga format (främst DataDisc 80/FD2) skall enligt konstruktören klaras av nästa version av programmet.

När det gäller standardprogram finns en del av de populära ABC800-programmen i versioner för PC. Det gäller bl a de populära registerhanteringsprogrammen Dbas.2 och REG.800 (*). Att flytta över ORD.800 eller OBS.800 till PC vore emellertid ganska korkat. Det finns ordbehandlingsprogram för PC och PC/AT som är helt överlägsna. Det kanske bästa heter Word Perfect (*). Det går utmärkt att läsa in textfiler från ORD.800 eller OBS.800 till Word Perfect!

En ny värld

En ny värld öppnar sig, en större värld som dessutom kommer att växa vidare med 80386 som jag hoppas få återkomma till i en annan artikel!

<365>
Göran Engelbo

(*) Säljs av cadett ab, tel 08 - 750 59 80

Möte MONITOR

(Text 4535) Bo Kullmar * <1789>
Ärende: Hängning
Systemet hängde sig den 28:e december troligen på grund av en nätstörning. Vi har fått en 2000 KV nätstabb av Taxi, Stockholm, som skall kopplas in på alla maskiner. Just nu är bara centralen skyddad. Detta skall förhoppningsvis göra att vi slipper denna typen av störningar i fortsättningen. Kent var tvungen att resetta närcentralen vilket i sin tur gjorde att MSG:s ISAM-databas fick en kvaddad nyckelfil. Denna har jag reparerat under kvällen/natten till lördagen med reperationsprogrammet i Meny0. Normalt brukar jag kunna få igång nätet utan att resetta nätcentralen vid de hängningar som inträffar med jämna mellanrum.

(Text 4536) Per Holmgren <5213>
Kommentar till text: 4535
Ärende: 2000 KV nätstab.
Menar du verkligen att vi har fått en grunka som stabiliserar 2 Miljoner volt? (. Vad ska vi med den till, undrar man då ängsligt.. .)
Du kanske menar att den klarar 2 KVA, och det är ju bra i så fall.

(Text 4571) Bo Kullmar * <1789>
Ärende: Testa av nytt modem
Klubben har sedan en tid ett Selic Mulit-modem som jag testar. Nu har det börjat att bli stil på programvaran så jag skulle vilja ha hjälp med testa från flera olika typer av modem.
Multimodemet är ett modem som kan koppa upp mot 1200/1200 V22, 1200/75 V23 och 300/300 V21 modem och sedan köra 1200/-1200 i alla lägen mot värddatorn.
Först testar modemet om det är 1200/1200 och sedan 1200/75 och sist 300/300. Det är en liten kort paus emellan dessa. Jag skulle alltså vilja att ni provringde till modemet med olika modem och rapporterade i brev till mötet sysop (brev sysop) hur det går. Berätta då vad ni kör för modem och hastighet.
Teleonnummret är 08-751 64 48 och det går till min DS90-10a. Slå några return och sedan skall texten "Login:" dyka upp. Skriv då "who", obs med små bokstäver. Därvid kommer det en kort text och sedan kopplas linjen ner.
Senare kommer jag själv att skaffa ett sådat modem och erbjuda alla medlemmar konto. Nu går det dock inte bra att köra 300/300, men det beror på DS90. Man ser detta genom att text tappas när man får kontakt. När jag erbjuder alla att köra skall dock detta vara fixat genom ett speciellt kort som jag får låna. Inget sådant problem finns om man kör på ABC802.
Om modemet fyller de fodringar som vi har så kommer vi att köpa ett till och sätta in dessa på 80 64 46 och 80 64 45. De modemen som friställs skickas till Linköping.
Jag ringer själv ut på samma linje och då går det inte att ringa. Iblad blinkar modemet till på DS90 (var 60:e sekund) om om man ringer just då så kanske inte modemet svarar. Ring igen då.
Tidigare har det gått bra att ringa upp med alla typer av modem, men det gick inte med ITT:s UNIMODEM. Detta är dock fixat nu, men vi vet inte om detta i sin tur kan ställa till problem för andra modem.

(Text 4596) Gunnar Tidner <1306>
Ärende: Uppdelning av filer i 8-bits och 7-bits i skilda bibliotek
Det finns textfiler som rör PC och MS-DOS både i normal svensk Ascii (7-bits) och textfiler i 8-bits ascii enl IBM standard. 8-bits filerna måste hämtas med KERMIT,SI och 7-bits filerna med KERMIT,S eller något ABCFIL-program. 7-bitsfilerna lagras enligt ABC-standard dvs med space-kompression så att en följd av blanka lagras med ascii-9 följt av en byte som anger antalet blanka i följd. Vidare lagras radslut som enbart CR, ej CRLF. Hämtar man en 7-bitsfil med KERMIT,SI i tron att filen låg i 8-bitsformat blir filen i stort sett oläslig i en PC.
Hämtar man med ABC80x en textfil som är lagrad i IBM 8-bits format blir det också galet.
Ett sätt att testa vilket format filen ligger i är naturligtvis att göra TYPE <filnamn> först och avbryta med CTRL-C när man ser vad det är. Jag tycker därför att det vore bra om man visste att alla filer i bibloteke hade antingen gemgående 8-bits format eller när det gäller ABC-program att alla textfiler ligger i 7-bitsformat, liksom programmen i BAS-format. Att programmen i BAC-format är 8-bits format råder väl ingen tvekan om.

(Text 4597) Bo Kullmar * <1789>
Kommentar till text: 4596
Ärende: Uppdelning av filer i 8-bits och 7-bits i skilda bibliotek
Jag föreslår följade:
Alla textfiler, t ex med extension TXT och INF, läggs upp i svensk 7 bits ASCII. Jag har möjlighet att konvertera de IBM 8 bitars textfiler som kommer in. Detta gäller också BAS-filer.
Övriga filer, dvs främst filer med extension ARC, COM och EXE läggs upp i som binärfiler, dvs 8 bitars IBM-filer.
Tyvärr går det inte att dela upp textfiler för ABC i ett bibliok och dito för IBM i ett annat eftersom det ofta finns en liten textfil skriven med WRITE till ett program som är inskickat med kermit. Detta innebär också att jag tar bort ABCDISK.INF till förmån för ABCDISK.TXT som alltså är samma fil i svenska 7 bitars ASCII.
Jag har möjlighet att per disk konvertera filer mellan ABC och MSDOS format. De MSDOS filer som jag därvid lägger i monitorn kommer då att få en storlek i jämna 4-tal ABC-sektorer (253 bytes per sektor).
Skall man hämta en textfil till PC går man alltså "KERMIT,S <fil>" och skall man hämta en binärfil så gör man "KERMIT,SI <fil>". Skall man skicka in filer går man alltid "KERMIT,RI" om det är från PC man kör. Filen hamnar då i det senare fallet i inlådan och är det en textfil så kommer den att konverteras till svensk 7 bits ASCII.
OBServera att man alltid måste använda paritet space när man kör kermit!

(Text 4603) Peter Thärning <3707>
Ärende: Chr$(127), backspace, fylld ruta
Rader i filer inskickade till monitorn som innehåller ovan nämnda tecken är omöjliga att ändra i (kommando ländra n). Man kan inte heller läsa från teknet eftersom det då försvinner och kan inte skrivas tillbaka.

(Text 4604) Bo Kullmar * <1789>
Kommentar till text: 4603
Ärende: Chr$(127), backspace, fylld ruta
Ja, det går inte! Det beror ju på att man vill att DEL skall ha betydelsen av BS i monitorn och att DEL används som fylld ruta i ABC. Man kan inte få allt!

(Text 4646) Bo Kullmar * <1789>
Ärende: Nya modem
Nu sitter det två Selic Multimodem på 80 64 45 och 80 64 46. Dessa modem börjar först och kolla om det är ett 1200/1200 modem som ringer upp och så ej är fallet så kollar det 1200/75 och till slut 300/300 om vi kommer så långt. Modemet kör alltid 1200/1200 mot datorn. Det finns en buffert i modemet och när den är full tas CTS ner.
Troligtvis kan man ej ringa upp mot modemet automatiskt med ett TGC modem om man ringer i 300/300. Tester tyder på detta. Fast man kör väl 1200/75 då i första hand så det går inget.
I mån av tillgång på pengar så är avsikten att byta resteraden modem på gruppnummret till denna typ av modem. Sendan skulle vi kunna om vi har råd byta 80 11 55 modemet till ett modem som klarar 1200/1200 och 2400/2400.

(Text 4650) Bo Kullmar * <1789>
Kommentar till text: 4623
Ärende: V22 bis har lurverket monopol på
Jag har nu beställt ett Alfa-NET modem till 801155 som klarar både 1200/1200 och 2400/2400. Eftersom vi bara skaffar ett sådant modem och byter de resterande på gruppnummret till Selic Multimodem så blir det bara ca 3000 kronor dyrare med Alfa-NET modemet.
Det är väl värt om monitorkörningen därmed kan gå snabbare för folk med höga telefonräkningar tycker jag. Att skaffa Alfa-NET modem till alla linjer blir för dyrt så länge dessa modem är dyra.
Telverket kommer att tillhandahålla dessa modem och då kommer man att kunna köra allt upp till 2400. Vi köper ett av Alfa-NET:s modem som de har tagit hem på prov och de har Televerkets tillstånd underhand att sälja ut dessa trots att man ej får köra 2400. Bytet av resterande modem kan ske senare i vår. I varje fall måste vi avvakta årsmötet.
Man måste dock räkna med att det krävs bra telefonlinjer för att köra 2400. Sen är det inte säkert att man vinner så mycket hastighet när man kör MSG eftersom systemet har en viss tröghet.
Jag har kört på 80 64 46 i 1200/1200 nu några gånger och inte fått en enda störning.
Jag undrar om det kan vara 1200/1200 Heath modemet som stör eftersom Selic-modemet står intill. Funderar sen på att byta modemen på 80 64 46 och 80 64 45 för att se om det är någon skillnad.
Alfa-NET modemet kommer troligen att installeras under helgen som kommer.

Möte Z80

(Text 639) Patric Ljung <5455>
Ärende: Rotate digit
Finns det nåt lämpligt sätt att flytta bitarna D0-D3 till D4-D7? Allra helst valfritt register. I nuvande fall görs det av 4 st sla c, vilket tar 8 minnes läsningar, vore intressant att få ner det till 4.
hoppas på häftiga svar.

(Text 640) Peter Thärning <3707>
Kommentar till text: 639
Ärende: Rotate digit
Här kommer ett RÅhäftigt svar:

```
Old way:                    New way:
==========================================
Instruction  M  T     Instruction   M  T
SLA r        2  8     LD A,r        1  4
SLA r        2  8     ADD A,A       1  4
SLA r        2  8     ADD A,A       1  4
SLA r        2  8     ADD A,A       1  4
                      ADD A,A       1  4
                      LD r,A        1  4
==========================================
             8  32                   6  24
```

Om du kan använda Acken från början så kommer du ned i dina efterlysta 4M. r ma bytas ut mot B, C, D, E eller H.

(Text 676) Patric Ljung <5455>
Kommentar till text: 672
Ärende: Liten lektion liten kommentar
För att kanske ge en bra överblick av maskinkods-programmering kanske man bör dra sig till minnes hur jag själv upptäckter -The secret of machine-code!
Det egentliga problemet var för mig att jag inte visste vad bit och byte var. Samt hur minnet fungerade, adressering osv. Hur en liten u-processor såg ut, generellt sett så är de väl ganska lika?
När jag vipps på en kvart förstod hur allt förhöll sig var det inga problem att veta hur man programmerar i maskinkod.
vad jag sedan lärt mig genom åren är olika tekniker att programmera. jag försöker skriva min maskinkod så funktionsbaserad som möjligt. Dvs att nästan gå från C-struktur till maskinkod. Dock används ju processorns register till stor del som direkta variabler. Istället för att ha dem utslängda i minnet. Man kanske borde börja så pass elemntärt så att folk som nästan inte hajat maskinkod kan komma med i 'kursen'.
Ett något klurigt problem kan vara att man behöver rita litet figurer.
Med allt detta menar jag att kanske en betydlig nyckel till att förstå maskinkod (och även hela datorns konstruktion, i princip) och kunna programmera den är att just att förstå det här med bit, bytes, adresser, CPU-register osv. För mig var det det jag behövde veta för att komma igång (Handassemblering med POKE).
När jag sedan fick köpte ASMZ fanns det ju en del att lära där, men det har ju faktiskt inte nå't direkt att göra med maskinkod.

(Text 677) Anders Franzén <5258>
Kommentar till text: 676
Ärende: Liten lektion liten kommentar
Jag håller inte riktigt med dig. Enligt min mening måste man ha en grundförståelse av vad programmering innebär. Man måste veta hur man strukturerar program, olika tekniker för att läsa filer och länga upp EOF osv. Kan man det kan man skriva program i vilket programmeringsspråk som helst.
Jag tycker t ex inte att det viktigaste är att lära sig Z80-s alla register så där direkt. Dem lär man sig allt eftersom man stöter på dem. Jag ser heller inget absolut krav på att man ska kunna räkna binärt på två fingrar. Med en assemblator kan man uttrycka tal i vårt vanliga decimala talsystem. Däremot anser jag det är bra att börja lära sig instruktioner som hanterar programflödet, t ex JP och CALL. Dessa kan direkt jämföras med andra språk och är lätta att förstå. När man har ett hum om hur dessa funkar kan man pula in instruktioner som hanterar data i flödet.
Det är nog bra att känna till begränsingar i talområdet, i BASIC kan man hantera riktigt stora tal men i assembler endast tal som motsvarar BASIC-ens heltal. Att de består av just 16 bitar behöver man inte känna till så där på en gång men eftersom man är nyfiken så lär man sig sådana saker efter hand.
Alltså, assembler är väldigt enkelt. Precis som allt annat när man väl har lärt sig det!

(Text 686) Hans Kaplan <6710>
Ärende: Assembler
För att bli intresserad av assembler måste man veta lite om fördelarna med språket.
Jag tycker faktiskt att man på en gång lika gärna kan sätta sig in lite i hur datorn jobbar (processorn med sina register och adressering)moder åtminstone och varför inte känna lite på MIKROPROGRAMMET som utför assemblerinstruktionerna)
Jag har aldrig programmerat Z80 så jag överlåter redogörelsen betr. olika adressering-issätt, men detta måste man ta med i början. Fick sån lust att berätta lite om det så sällan omtalade mikroprogrammet:
Om man är en lyckling person så har man en s.k. "assemblator" som översätter de ord-instruktioner (mnemonics) man fått fram till en sifferkod som avkodaren i cput kan förstå. Ett vanligt mellansteg före den ofrånkomliga binärkoden är "hexkod" som lätt kan översättas till binärkod. Basen i hexkod är 16, och tecknen 0 - 9 samt A - F.
Nåväl, man tänker sig att man är en ström av ettor och nollor som av "programräknaren" ett register som normalt räknar framåt marsch, hämtats ur en position i minnet och via ett dataregister plötsligt finner sig själv på databussen, en massa trådar - en tråd för varje etta eller nolla. Nyfiket följer man sen sin framfart över bussen och innan man vet "ordet av" så sitter man ohjälpligt fast i "instruktionsregistret" i kontrollenhet i cpu. Nu får man inte för lite känna sig själv, det visar sig att man är en OPERATIONSKOD som avkodaren börjar smaska i sig av. Jassså, är jag en operationskod, vae är det då och vad händer nu. Tanken är knappt färdigtänkt förrän ett litet program "som alltid finns där" hoppat till en rutin, bara för min skull.
Jag frågade programmet, det hette "mikro" och hade lika många rutiner som alla mina kompisar där borta i minnet om man undantar alla tvillingar. Dessutom fanns det där hoppet förstås.
Sen hände allting snabbt, jag förvandlades till en "signalflöjt" som öppnade vägen för en ansamling ettor och nollor i ett "register", via "internbussen" i cpu till ett annat register där en annan ansamling brutalt förintades.
Ja, det var faktiskt allt, själv förintas jag men min "själ" finns kvar i minnet.
Så lång historia om så lite, ja så kan det bli när man dristar sig att kika under assemblernivån, kanske kul att nån gång ha gjort det innan man kavlar upp ärmarna för programmering?
Vem redogör för fördelar med assembler och adresseringssätt?

(Text 687) Hans Kaplan <6710>
Kommentar till text: 686
Ärende: Assembler
Allting i minnet är väl inte "operationskoder", kan du inte redogöra för annat innehåll?
Och vad menar du med att u-programmet har lika många rutiner som "kompisarn i minnet utom deras tvillingar"??

(Text 688) Hans Kaplan <6710>
Kommentar till text: 687
Ärende: Assembler
Oh, det där med rutinerna först, det finns alltså en rutin för varje typ av operationskod. Vi tar ett annat exempel:
Instruktionen är JUMP (adr) vilket inte oväntat betyder ovillkorligt hopp till adressen i fråga.
ööö När operationskoden för jump (två hextal) når avkodaren så vet kontrollern genast att därefter ska följa en adress och skickar därför efter innehållet i de två följande minnescellerna, eftersom adresser består av fyra hextal. I varje minnescell ryms alltså två hextal (siffror) i en 8 bitsdator. När adressen har erhållits så matas den helt enkelt på programräknarregistret som fortsätter sin framräkning från det nya värdet och plockar ut innehållet i cell efter cell i minnet.
Minnesceller kan också innehålla förutom operationskoder och adresser rena "data" (!) som bland mycket annat kan vara ett jämförvärde som i pseudoassemblern:

```
Move 0,D1   (nollställer reg. D1)
LOOP    Inor D1     (D1 + 1)
Cpi  FF,D1  (Har D1 värdet FF?)
Bne  LOOP   (Nej)
xxx  xxxx   (ja)
```

Detta ger en liten tidsfördröjning (önskar jag kunde Z80-syntax, men principen är exakt samma)
När man skrivit ett sånt här program så får en assembler ta hand om det och då omvandlas allt till hexkod, labeln LOOP tilldelas en adress och sparas i en tabell, den kan ju komma att åsyftas återigen.
Den här programutvecklingen/editeringen görs ofta på en stordator och sedan "laddas" filen ner i en mikrodator eller får styra en prombrännare.

(Text 696) Hans Kaplan <6710>
Ärende: Lite jämföreiser 68K / 8 bitars processorer
68K har 23 pinnar för adressbussen vilket innebär att den direkt kan adressera 16M istället för det vanliga 64K.
Den 24:e pinnen har man sparat därför att den alltid skulle sända ut en nolla - ojämna bytes adresseras inte. De ord som kan genereras är 8/16 eller 32 bitar långa. Databussen och de använda minnena bygger på 16 bitar "word", 32 bitars ord "långa ord" sänds ut i två omgångar. 16 register + PC och SP innehåller 32 bitar, statusregistret har normala 16. De här längre orden medför att operationskoderna kunnat göras längre, mer och komplexare (kraftfullare) instruktioner samt möjligheten att medföra en del data i själva instruktionen är följden. Det är inte enbart positivt, den enklare instr.uppsättningen i Z80 och 8085 tex är mer överskådlig, och knappt någon programmerare och ännu mindre kompillator förmår använda uppsättningen fullt ut. Det försvårar också arbetet att ständigt hålla reda på bytes, words och långa ord - detta anges med prefixen .b resp .W (default) och .l. Gör man fel här så är risken stor att det hela spårar ur, om man tex ska jämföra ett värde som anges i instruktionen med ett värde lagrat någonstans i minnet så måste ordlängden förstås stämma exakt.
Det man annars mest uppskattar vid övergång från normal 8 -bitarsmygga är detta att man inte är stenhårt bunden till "ackumulatorn" som i 8085. Har man i den senare ett sånt där jämförvärde igen eller något som ska skickas ut på en utgång så måste man alltid först skyffla över värdet till ackumulatorn. Det behövs inte med 68K.
När man adresserar minnet med 8-bitars proc. så vill jag minnas att man måste ange alla fyra hexsiffrorna, men med 68K så gäller vid lägre adresser 4 hexsiffror ("absolut kort") och högre adresser 8 st ("abs. lång"). I övrigt betr. adresseringssätt ber jag att få återkomma i avvaktan på tillrättavisningar och frågor.

(Text 697) Per Fagerkvist <3862>
Ärende: HITACHI-processor
HEJ !
Jag sitter med manualen för HD64180 framför mig. Sälj av RIFA nu även av ZILOG. PROCESSORN är Z80 kompatibel,men har dessutom en rad extra finesser.

Kan adressera 512 kb (8 st 64 kb block)
2 st DART
2 ST CTC
2 ST DMA som direkt kan adressera 512 kb.
1 st SIO

Allt i samma kapsel.Intern wait-state gen.
64 kb I/O omfång.
8 bitar refresh.
Allt som behövs är minne,stab,och (TTL/RS232) transiever.
För att ha en komplett dator.
MvH. PER.

(Text 698) Gunnar Forssell <1631>
Ärende: u-program.
Azzä, det där med u-program tycker jag ni har kråkfat till onödigt mycket! Det är ju enkelt som bara den! Operandkoden pekar ut en adress i u-minnet, och dess innehåll går ut och styr allt annat, t.ex. ALU'n, registerminnet, etc. Ungefär som en massa tappkranar. Den utrustning som KTH har för detta ändamål (MILDA = Mikroprogrammerbar LaborationsDAtor) är jättekul, man fattar verkligen vad det rör sig om. Att få skriva u-pgm'et för t.ex. "JP NZ,LABEL" är MYCKET lärorikt!
Jag har hört att japsen ska komma ut med en processor med EXTERNT u-pgm i EPROM, någon som vet mer? 32-bitars ALU skulle den visst ha.

(Text 699) Hans Kaplan <6710>
Kommentar till text: 698
Ärende: u-program.
Om något är redogjort i detalj, steg för steg, så är det inte "tillkrånglat utan avpassat för dem som aldrig en kanske hört talas om saken ifråga. Detta gäller u-prog. och många "datafantaster", t m lärare.

(Text 700) Gunnar Forssell <1631>
Kommentar till text: 699
Ärende: u-program
Av det jag läste om u-program, som du hade skrivit, fattade jag inte vad du menade. Om jag sedan inte hade vetat vad u-program var så hade jag nog undrat vad sjutton u-program var för något.
Alla som vill lära sig data, borde ta reda på hur processorn funkar i detalj!

(Text 701) Kristoffer Eriksson * <5357>
Kommentar till text: 698
Ärende: u-program.
Riktigt så enkelt som du beskriver det är det nog oftast inte. Den enkla konstruktionen med instruktionskoder in i u-programminnet, och processorstyrningssignaler ut, duger bra för enkla instruktionssset, men det finns en hel del processorer med mer komplcerade instruktionsset (före RISC-eran), där man får ta till något mer komplicerat, t ex två-nivåers-avkodning.
Ta t ex Z80:ns extra instruktioner som inleds med ED(hex)-koden. Eller 16-bittars-processorer med 16-bittars instruktionskoder, ska de ha 64kord (ord troligen betydligt större än bytes) u-programminne?
Och så finns det vissa processorer med så rätfram instruktonsuppsättning att de kan avkodas med ett vanligt grind-nät.

(Text 702) Peter Goldmann <5080>
Ärende: CPU:er
Inläggen i mötena kan ju publiceras i ABC-bladet. Jag tycker det vore intressant med en genomgång här i mötet av de olika typer av CPU:er som finns idag - om någon ville göra en lite mer genomarbetad sammanfattning.
Den kan ju gärna jämföra de olika processorerna med utgångspunkt från Z80 och ange deras fördelar och nackdelar -användbarhet i olika special sammanhang.
Själv snubblade jag över en 8039 som en vänlig person upplyste om var en maskprommad CPU. Senare fick jag lära att man kunde via ett stift på kapseln välja bort den interna masken och koppla in yttre PROMMAR istället.
Jag har provat på 1802 lite. Det var en CMOS-processor - strömsnål och den hade ingen refresh för dynamiska minnen. Däremot hade den 16 st 16 bitras interna register som med enkel instruktion kunde ta över som programräknare eller minnespekare. Vidare hade den en utgång som kunde styras programässigt att vara hög eller låg en sk Q-utgång som på en vippa. Vidare hade den fyra ingångar som kände av nivåer och som kunde styras programmässigt. Hela CPU:n kunde kontroleras med en clockingång antingen ansluten till kristall eller stegas med valfri hastighet upp till max -frekvens. Intressanta egenskaperi Strömsnål, enkel att använda utan många yttre kretsar för I/O. DMA - direkt memory access fanns.

(Text 703) Gunnar Forssell <1631>
Kommentar till text: 701
Ärende: u-program.
Jo, det är klart att dagens avancerade processorer har mer avancerade u-program! Men principen är (väl?) densamma för alla processorer, eller hur?
Allt kan göras svårt och komplicerat, men hur det i grund och botten funkar bör vara ganska lätt att beskriva, eller är jag en komplett idiot på datamarknaden?

(Text 706) Hans Kaplan <6710>
Kommentar till text: 703
Ärende: u-program.
Nej, Risc-konceptet innebär bl.a. reducerad instr. uppsättning och rel. enkla instruktioner -men snabba, alla fullföljes på en klockcykel mot normalt 2 - 6 cykler tror jag.
Detta för att kompillatorer och människor har svårt att överblicka och utnyttja en stor uppsättning fullt ut. Ingen av oss är väl på datamarknaden, så du är alltså ingen "idiot på datamarknaden".

(Text 707) Per Holmgren <5213>
Kommentar till text: 697
Ärende: HD64180
HD64180 görs av Hitachi och numera även Zilog, och du kan köpa den via de svenska distributörerna Ferner och Rifa.
Det har gått en serie artiklar i Byte om en ytterst kompakt dator med just den processorn, 512k byte Ram, fd och hd controllers, och nu senast ett grafikkort gjort på HD63484 Advanced Crt Controller. Mycke godis!

(Text 708) Per Holmgren <5213>
Kommentar till text: 698
Ärende: 32 bit processor med u-kod i Eprom.
Tja, det låter som Hitachi's AI32. Det kan dröja ett tag innan vi kan köpa den i affärerna, men den är på väg.

(Text 709) Gunnar Forssell <1631>
Kommentar till text: 706
Ärende: u-program.
Att genomföra en operand på bara en klockcykel borde vara svårt, men inte omöjligt. I den kurs jag läser om datorers arbetssätt används en enkel modelldator, med gemensam data/adressbuss. Mycket jobbigt! Detta gör att man får hålla reda på vad som finns på bussen, vart det ska, etc. Om man bara kan dela på adresserna och datan bör man kunna hotta upp den datorn (klockan går t.ex. på 1 kHz...) rätt ordentligt.
Känske en ny grej: Bygg din egen Z80 utan Z80!

(Text 710) Gunnar Forssell <1631>
Kommentar till text: 708
Ärende: 32 bit processor med u-kod i Eprom.
Mums! Kan du något mer om den? Tänk att få ha u-koden i [illegible] det kan ge en hel del kul effekter.

(Text 711) Per Holmgren <5213>
Kommentar till text: 710
Ärende: 32 bit processor med u-kod i Eprom.
Det där med u-kod i Ram har diskuterats här rätt nyligen. Det framgick att Datasaab gjorde en datta kallad D23 som kunde modifiera u-koden från applikationsprogrammet. Bäddat för *grymma* loppor...
Det jag har om AI32 är av typen förhands-info, som snabbt kan ändras.
- o 32 bit arkitektur, 32 bit buss och 32 bit ALU och Shifter.
- o Redigt med minne ombord, 2 register uppsättningar med vardera 256 register a 32 bitar. Kan konfigureras som instruktions/data cache, eller som vanliga register.
- o 4k ord x 64 bit Eprom för mikrokoden.
- o Tänkt för AI tillämpningar, bildanalys, taligenkänning, expertsystem, etc. Support kommer att finnas för Smalltalk, Prolog och Lisp.

Återkommer med mera detaljer när jag får reda på dem.

(Text 712) David Andersson <5201>
Kommentar till text: 696
Ärende: Lite jämförelser 68K / 8 bitars processorer
Nämnas kan att 68000 har 16 stycken 32-bitars register. Ett av dem (A7 tror jag) används som stackpekare för subrutinanrop och interupt. Annars finns det inget som hindrar att programmeraren använder vilket som helst av registrerna A0 till A7 som stackpekare. (Speciellt om man vill ha flera stackar).
Oftast adresserar man minnet med "långa" eller "korta" relativa adresser. Det kan vara PC eller ett register plus/minus ett 16-bitars tal (korta) eller ett 32 bitars tal (långa). Absoluta adresser är inte bra när man har så mycket minne som det brukar vara frågan om när man har 16/32-bitars processorer. Programmen blir för lite flexibla.

(Text 715) Gunnar Forssell <1631>
Kommentar till text: 711
Ärende: 32 bit processor med u-kod i Eprom.
Jovisst kan det bli loppor så att man baxnar, men tänk på vad man kan göra! "Ska vi köra lite Intel, eller kanske lite Motorola idag?"
Ska bli ql att få titta närmare på den när den kommer...

(Text 717) Benny Löfgren * <2615>
Kommentar till text: 712
Ärende: Lite jämförelser 68K / 8 bitars processorer
Har man vettig hårdvara med MMU så gör det inget om man har absoluta adresser, då konverterar hårdvaran automatiskt den logiska adressen till en fysisk.

Möte CP/M

(Text 675) Thomas Lindqvist <6774>
Ärende: Bygga ut ABC80 för CPM
Jag har en abc80 32K +FD2 drive. Kan någon vänlig själ med ett brev i ådan tipsa om hur jag enklast (och billigast) bygger ut detta system för CPM. Kanske någon också har nödvändig utrustn. till ett humant pris.

(Text 676) Bo Michaelsson <913>
Kommentar till text: 675
Ärende: Bygga ut ABC80 för CPM
MyABs CP/M-tillsats innehåller ett kort och programvara. Problemet är din disk som antagligen gör det hela trångt. Det blir ett väldigt skyfflande med skivor om det alls går. Man kan ju bygga ut disken med ett annat kontrollkort och få dubbel densitet. Det billigaste sättet att göra detta är kanske att köpa MyABs UNIDISK-kort men det vet jag inte om det fungerar. I klubben har dock vissa sagt att ett sådant kort gör om disken till en FD2-D, tror jag.
Jag tycker CP/M är så pass vettigt och din fråga så intressant att jag heilre svarar så här.

(Text 677) Kent Berggren * <6019>
Kommentar till text: 676
Ärende: Bygga ut ABC80 för CPM
Fanns någon beskrivning i mikrodatorn för ett antal år sedan

(Text 678) Tomas Wikström <1398>
Kommentar till text: 677
Ärende: Bygga ut ABC80 för CPM
Jag är också mycket intresserad av det. Kan Du komma ihåg,Kent,på ett ungefär när det var?
Finns det någon som vet om man med en "vanlig" (eller med en Gejo's) 64 (128) K-ombyggnad kan få snurr på CPM i en ABC80?
Jag skickade för ca en vecka sedan en intresseanmälan till CPM-klubben i Värmdö (tror jag om jag kommer ihåg rätt) men inget hört ännu. Någon som vet nåt om denna klubb?
De har monitor (!),om de nu kallar den så,med KOM-liknande system. Ca 3-4 tusen prg-titlar,m.m....
Så jag väntar med spänd förväntan på svar därifrån.

(Text 679) Kent Berggren * <6019>
Kommentar till text: 678
Ärende: Bygga ut ABC80 för CPM
Ge-Jo kan inte byggas om till CPM. det går inte helt enkelt. Tror attt det har att göra med att man inte kan flytta bildminnet eller om det var doset. Tyvär blir alla dessa om bygnader av ABC80 ganska lång-samma. Personligen tycker jag inte det är någon ide att sats på v 3,0 utan endast 2.2. Jag har en del programvara till den och kan i mån av tid hjälpa till att få fram program.

(Text 686) Bo Michaelsson <913>
Ärende: CP/M multisystem
CP/M multisystem heter MyABs system för CP/M med olika diskar samtidigt anslutna. Man kan tydligen ha 830, 832 och 838, ev även hårddisk. Nu har jag fått höra att CP/M 3,0 multisystem inte fungerar med Luxors nya kontroller-kort utan bara med det gamla eller med MyABs eget Unidisk-kort. Något nytt passande CP/M lär väl inte vara att vänta. Har någon löst problemet utan att köpa ett gammalt kort med kabel? Finns någon annan lösning?

(Text 687) Jaan Tombach <4283>
Kommentar till text: 686
Ärende: CP/M multisystem
Kör CP/M 2,2G, fungerar med olika drivar och med nya kontrollerkortet. Hur den fungerar med winchester vet jag inte.
Mvh Janne

(Text 688) Bengt Holgersson <559>
Kommentar till text: 686
Ärende: CP/M multisystem
Vad jag har märkt så fungerar det utmärkt med Luxors nya kontrollerkort
Vari skulle problemen ligga ?
M.v.h Bengt H

(Text 689) Bo Michaelsson <913>
Kommentar till text: 688
Ärende: CP/M multisystem
Systemet börjar boota men det stannar innan prompten. Finns en rutin för automatisk inläsning av parametrar så exelveras den inte. Systemet låser sig så effektivt att man bara kommer loss med reset. Det går alltså ej att köra det.
Jag har också provat ett gammalt kontolikort men det är byglat för 832 och går ej i någon bygling alls om man kopplar om det. Jag har två skivor framställda med multisystem av MyAB men ingen går.

(Text 690) Bengt Holgersson <559>
Kommentar till text: 689
Ärende: CP/M multisystem
Problemet med MYABs multisystem i ori-gialutförande är att det vill boota från A: vilket är hårddisk. Jag har modifierat mitt system så att A:-B:=ABC830 C:-D:=ABC832 E:-F:=ABC838 .
Detta system bootas från ABC830 i drive A: och finns i monitorn under CPM. Vill man boota från ABC832 måste man ha denna som A: dessutom måste man fixa en bugg i boot-Prommet.
M.v.h Bengt H

(Text 691) Bo Michaelsson <913>
Kommentar till text: 690
Ärende: CP/M multisystem
Aha, mycket intressant. Kan Du beskriva hur man klarar av att boota från 838 som är det jag mest använder?

(Text 692) Per Fagerkvist <3862>
Ärende: CPM-DISC-FORMAT
HEJ !
Jag undrar lite över sektor indelning (antal), på CP/M Jag har en 5 "1/4. disc basf 6106 40 spår. Om jag nu kör ABC800, SÅ har jag 16 sektorer.
Jag har inte kört CP/M men jag har i minnet att sektor antalet är annat . Normalt körs ju CP/M på 8" drivar.
******** tacksam för svar ********

(Text 693) Bengt Holgersson <559>
Kommentar till text: 692
Ärende: CPM-DISC-FORMAT
Svaret på din fråga är inte entydigt. Antalet fysiska sektorer på ett spår är samma medan antalet logiska sektorer kan variera. I CP/M 2.2 är sektorstorleken 128 bytes medan i CP/M 3.0 är sektorstorleken 256 bytes. Då menar jag alltså BIOSens sektorstorlek vid direktläsning av skivan. Läser man via BDOSen så är recordstorleken alltid 128 bytes.
M.v.h Bengt H

(Text 694) Daniel Deimert <5439>
Ärende: CPM (vad annars..)
HEj! Jag kör CPM på min 806 med 832 drive.
Är det någon som kan skicka några CPM program till mig på skiva? (CPM plus 3.0) Jag har även en kompis som har CBM 128 med CPM3.0 Är det samma diskformat? Skriv!!!
ps Alla skivor returneras naturligtvis! ds
Mvh DD

(Text 695) Johan Hedberg <2755>
Kommentar till text: 694
Ärende: CPM (vad annars..)
CBM 128 har ej samma format som CP/M på ABC832 men däremot samma som ABC830 (egna erfarenheter).

(Text 700) Jan Johansson <1774>
Ärende: Bokföringsprogram
Jag söker ett bokföringsprogram (under CPM). Det ska användas i ett mindre företag, men bör innehålla möjligheter till projekt-redovisning (helst). Känner någon till något lämpligt/bra? Förslag på leverantörer?
/Hälsningar JJ

(Text 701) Bengt Alm'n <6415>
Kommentar till text: 700
Ärende: Bokföringsprogram
Ja HOGIA DATA har gjort ett proffsigt program som har fått goda resentioner. De bör nog ha kvar något ex.
Mvh Bengt

(Text 702) Hans Johansson <1678>
Ärende: CPM User Group
Hejsan!
Jag söker information om CPMUG. Är den fortfarande aktiv? Hur blir man medlem? Finns fortfarande möjligheterna att få diskar med Public Domain program?
Många frågor, hoppas att någon kan reda ut detta åt mig..
Hälsn Hasse...

(Text 705) Per Fagerkvist <3862>
Ärende: ASSMEBLER Z80
HEJ !
Finns det någon assembler till Z80 under cp/m
Eller är allt 8080 i cp/m.
Bara en stillsam fråga ?
MV. Per

(Text 706) Peter Hollsten <2431>
Kommentar till text: 705
Ärende: ASSMEBLER Z80
Det finns flera Z80 assembler under CP/M men dom flesta kostar en massa kosing. Dock finns det några som är omgjorda så att dom klarar Z80 tex Zasm. Om du (eller någon annan) vill kan jag skicka in den samt lite andra debughjälpmedel bla en fullskärmseditor (Z8E) som är PD.

(Text 719) Bo Michaelsson <913>
Kommentar till text: 718
Ärende: CPMUG
MyAB i Göteborg sålde skivor med CP/MUGs program. Jag tror man kan få filerna från QZ också.

(Text 743) Kent Berggren * <6019>
Ärende: Forth
Jag söker ett forth program till en CPM-maskin men hälls Forth83. Är det någon som har en hör av dej till mej.

(Text 747) Gunnar Faith-Ell <2733>
Kommentar till text: 745
Ärende: Forth
Det finns FORTH under FORTH-biblioteket på monitorn för CP/M även om det kanske inte är FORTH 83.

Visste Du att...

- man i princip kan bygga ut ABC80 till hur mycket RAM som helst, dock får man då arbeta med 'sidor' om 32 kB
- ett tecken på skärmen blir sex grafiska punkter
- en fylld ruta (CHR$(127)) fås med Ctrl-<
- markörens adress på skärmen ligger i cellerna 65011 och 65012
- POKE 65031,128 ger samma resultat som STOP
- inmatningsbufferten ligger mellan adress 65088 och 65207
- klockfrekvensen på ABC80 är bara 3 MHz, trots att Z80A har kapacitet för 4 MHz
- ett dubbelord avläses med PEEK(adress)+ PEEK(adress+1)*256
- BOFA på 16kB-ABC80 är 49152, på 32 kB-ABC80 32768
- man kan få reda på checksumman med FOR I=1 to 16383:A=A+PEEK(I):NEXT I: PRINT A
- checksumma 9913 är den senaste versionen av ABC80
- BOFA-pekaren ligger på adress 65052
- man bara har tillgång till 14,5 k-RAM på en 16 kB ABC80-kassett. Resten går åt till buffertar, systemvariabler o dyl
- SmartAid, SchoolAid, SuperSmartAid och resten av gänget kopplas till ABC-bussen
- en 60 minuters kassett rymmer ca 120 kB, på båda sidor ger alltså 240 kB?
- Cray 1/S, en superdator, kostar som mest 17 miljoner dollar(!)
- man i Sovjet använder miniatyriserade radiorör istället för transitorer och det är snabbare
- ABC80 BASIC är lika snabb som t ex PASCAL på många andra datorer
- ABC80 BASIC är ca 4 gånger snabbare än MicroSoft BASIC för CP/M-maskiner?
- ABC80 har en grafisk upplösning av 72*80 punkter
- standardskärmarna för de gamla ABC80 är bärbara TV-apparater
- ljudet på ABC80 blir högre om man stänger av bandspelaren, t ex med OUT 6,7
- man kan känna av den senast nedtryckta tangenten på tangentbordet med INP(56). Om INP(56)>128 så är tangenten nertryckt
- man kan få blinkande test, överst till vänster med POKE 31744, PEEK(31744)+128 Prova med t ex

  POKE 31745, PEEK(31745)+128
- SYSOP betyder SYStemOPeratör
- BARK är förkortning för binär aritmetisk reläkalkylator
- BESK är förkortning för binär elektronisk sekvenskalkylator
- BASIC är förkortning för Beginners Allpurpose Symbolic Instruktion Code
- IBM PC/AT är förkortning för International Business Machines Corporation Personal Computer Advanded Technology
- ASCII är förkortning för American Standard [illegible]e for Information Interchange

Standardbandspelare till ABC80

Eftersom många har funderat över hur man kopplar in en vanlig bandspelare till sin ABC80 så tänkte jag och Anders Johnson beskriva hur vi gjorde.

Meningen med detta bygge är alltså att koppla in en vanlig bandspelare, kassettdäck eller en rullbandspelare till sin ABC80 för att t ex ha som backupminne. Det vore för övrigt ganska trevligt att ha en rull-bandspelare och FASTCAS som backup.

Vi utgick ifrån orginalbandspelaren till ABC80 (skokartongen). Det hade säkert även gått att modifiera bandspelaren ABC821, men vi valde den gamla varianten. För den intresserade så bör också nämnas att det finns en servicemanual till ABC821 (Luxor 66 77821-40) med schema, blockschema, komponentförteckning, layout mm.

För att kunna koppla in en vanlig bandspelare till kortet så var det nödvändigt att göra ett par ändringar. För det första så måste signalen ut till bandspelaren minskas till en lagom nivå (D202, D203), dessutom så ligger det en likspänning på utgången som måste filtreras bort (C201). På ingången på avspelningsdelen så ligger det två dioder parallellt med R104 som begränsar in nivån, eftersom det annars kan uppstå problem med vissa bandspelare.

Kretskortet.

Det enkelsidiga kretskortet framställs på sedvanligt vis med fotoresist och etsning. Här bör nämnas, att isolationsavstånden på vissa ställen är ganska små, och att kortet därför skall kontrolleras noga efter etsningen (se fig 2). Kortet lämpar sig knappast för den, som aldrig tidigare hållit i en lödpenna.

Anvisningar för hålborrning.

De två rektangulära fälten med lödplättar är avsedda som hjälpmedel för eventuella kompletteringar av kortet och behöver ej borras annat än vid behov. (Den som är road borrar naturligtvis ändå. det är svårare att borra i efterhand.) De fyra fästhålen (ett i varje hörn av kortet) borras med 3.1-3.3 mm borr. Hålen för S7 och K4, samt de tre hålen för spänningsmatning respektive jord borras med 1.3 mm. De två omärkta hålen vid K4, vilka inte ligger närmast kortets kant, behöver borras endast om man avser att skruva fast K4 (vilket ej är nödvändigt). I så fall borras de till ca 3 mm. Skruva först, löd sedan.

Alla övriga hål borras med 0.8 mm. Beroende på komponentval kan det kanske bli nödvändigt att borra upp vissa kondensatorhål till mera än 0.8 mm. För C101 finns två alternativa hålpositioner i 0-voltsledningen för att underlätta montering av kondensatorer av olika storlekar.

Glöm inte att löda dit R201 innan S7, eftersom den ligger placerad under S7.

Komponenter.

Allmänt, ekonomi.

Det är meningen, att man skall kunna bygga upp interfacet med billiga komponenter. Detta har haft viss betydelse vid kretskortets tillkomst. Om man har tillgång till ett lågprisställe för komponenter, kan åtskilliga tior sparas.

Resistorer. 1/4-Watts användes. De flesta typer (utom trådlindade) duger.

Elektrolytkondensatorer (C102, C103, C201). I prototypen har använts droppantal-elektrolyter, men även vanliga elektrolyter kan användas.

Övriga kondensatorer: Man tager vad man haver. Såväl keramiska som diverse plasttyper duger.

Halvledare: Man använder de angivna typerna, eller ekvivalen.

Mekaniska detaljer.

Till S7 har i prototypen använts en 6-polig 2-vägsomkopplare med dubbeltryck, fabrikat Schadow. 6-pol 2-vägs omk finns att köpa som surplus för bara 2:75 + moms hos Labb Elektronik. Man torde få betala minst ca 30:- för motsvarande om man betalar fullpris. En passande knapp till omkopplaren finns hos Elfa, med artikelnummer 35-599-29.

Till K4 har använts en vinklad 5-polig DIN-chassiekontakt för kretskortsmontage, Elfa 42-215-29. Priset är ca 3:-. En något dyrare variant med skärmat hölje är Elfa 42-191-27. Man kan givetvis använda någon trådansluten typ av DIN-kontakt i stället, eller rent av löda DIN-sladden direkt i kortet. K4:s stiftnummer på komponentplaceringsv yn stämmer överens med DIN-kontaktens. De onumrerade hålen är fästhål.

Hållare. På prototypkortet har använts hållare till de tre IC. Det är visserligen inte nödvändigt, men rekommendabelt, i synnerhet för den som inte är mycket lödvan. En 8-polig hållare har använts även till den 6-poliga IC201.

Prototypkortet är monterat på M3-gängade distansbussningar, en i varje hörn av kortet, som bör monteras i en skärmad låda.

Spänningsmatning.

Drivspänningar (+12V och -12V) till prototypen togs från V24-kontakten i ABC80. För att V24-kontakten skulle kunna användas till annat också, gjordes en "förlängare" av två 9-poliga D-subdon (se fig 5 och fig 6). Följande två typer (ett av var) användes: 9-poligt hylsdon med lödkoppar (Elfa 44-060-05)
9-poligt stiftdon för kretskortsmontage(Elfa 44-060-54)

D-subdonen skruvas först löst ihop med distanserna, samtidigt som man ser till att det ena donets lödstift hamnar i det andra donets lödkoppar och inte utanför eller i mellan (se fig 5). När detta är klart, drar man åt skruvarna och löder sedan alla nio förbindelserna. Slutligen löder man fast de tre matningstrådarna enligt fig 6. Lämpligen använder man trådar med olika färg, för att undvika förväxling. Ett praktiskt sätt är att använda tre sammanhängande ledare från en flerfärgad flatkabel.

Bandspelarkabel.

Beroende på bandspelare så väljer man en DIN 5- eller phonokabel. DIN 5-kabeln ska vara utav samma typ som används mellan en bandspelare och en förstärkare (rak kabel). Om kortet placeras i en låda så kan det vara lämpligt att använda en DIN 5 (hona) för chassimontage (typ Elfa 42-2060-4). För anslutning av kontakten se fig 4. Observera att stift 5 på S7 ska kopplas till stift 2 på kontakten. För phono-kontakter kan det vara lämpligt att använda två dubbla chassimonterade honkontakter (typ Elfa 42-2520-7). Se till att ansluta kabeln för bandspelarjord (stift 5 på S7) till höljet på bägge kontakterna. För den som har motorstyrning på sin bandspelare och vill kunna styra denna, så behövs det två telefonproppar (Elfa 42-7200-1) och lite tvåtrådig kabel.

Inspelning.

Vid inspelning så ska S7 vara intryckt. Skriv ut någonting till bandspelaren (t ex ett program) och sätt bandspelaren i inspelningsläge och tryck ner paus. Koppla bort eventuell brusreducering (typ Dolby) och justera om möjligt inspelningsnivån till ca +1dB medelvärde (visarinstrument) eller +3dB toppvärde (lysdiods-display). Tyvärr gäller samma sak som till t ex Spectrum, att man måste prova olika bandspelare till sin dator. Nu är allting klart att användas.

Övrigt.

Glöm inte att rengöra bandspelaren åtminstone 2 gånger per år. För rengöring används Tops och isopropylalkohol eller isopropanol (finns på apoteket). Rengör tonhuvud, raderhuvud, kapstanaxeln och tryckkudden. Tryckkudden rengörs med bandspelaren satt i läge play. Därefter ska bandspelaren avmagnetiseras med t ex en Emag Defluxer. Dra ur bandspelarens nätkontakt och sätt spetsen på "emag" så nära den detalj som ska avmagnetiseras som möjligt, utan att vidröra den. Tryck in knappen och för långsamt bort Defluxen till ett avstånd av ca 1.5 m, och släpp knappen. Detta förfarande upprepas på alla delar som ska avmagnetiseras.

De delar som ska avmagnetiseras är tonhuvud, raderhuvud och kapstanaxeln.

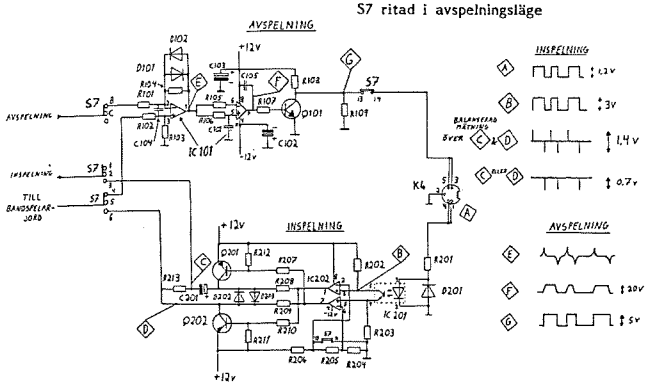
Vid osäkerhet lämna in bandspelaren till en serviceverkstad för rengöring och avmagnetisering.

Den som är intresserad av hur bandspelarminnet fungerar kan läsa mer i 'Mikrodatorns ABC' av Gunnar Markesjö.

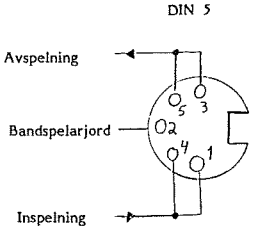
<5575> **Kjell Enblom**, 013-17 28 87,
<4001> **Anders Johnson**

Adresser:
LABB Elektronik, Tjärhovsgåtan 16, Stockholm, 08-41 86 30.
Elfa, Industrivägen 23, Solna, ordertelefon 08-735 35 35.

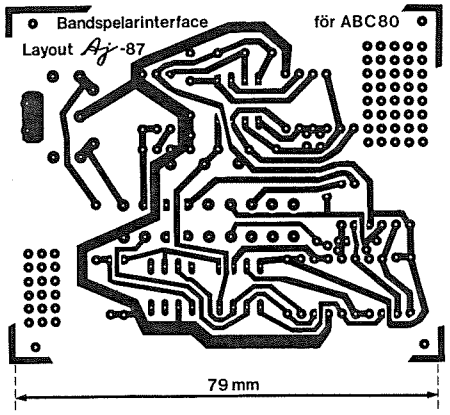
S7 ritad i avspelningsläge



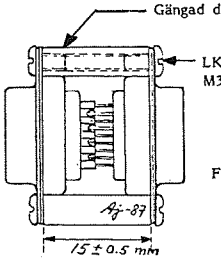
Figur 1



Figur 4

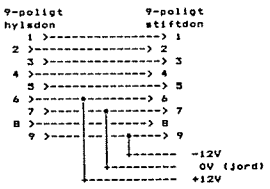


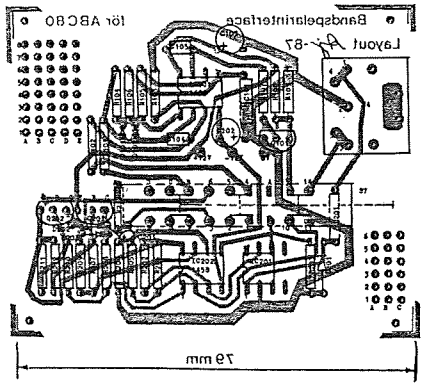
Figur 2



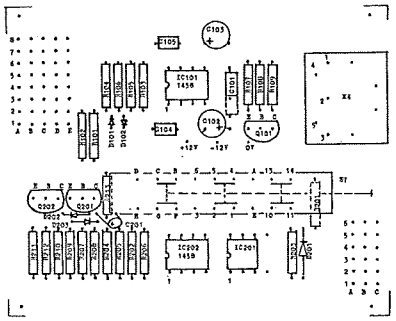
Figur 5

Förlängare med spänningsuttag för V-24-kontakten i ABC80





Figur 3



Komponenternas placering

Komponentlista

Resistorer 1/4 W

resistor	värde
R101	3.3k
R102	3.3k
R103	330k
R104	330k
R105	3.3k
R106	3.3k
R107	10k
R108	1.5k
R109	1.2k
R201	39 ohm
R202	56k
R203	5.6k
R204	10k
R205	22k
R206	10k
R207	100k
R208	39k
R209	39k
R210	100k
R211	10k
R212	10k
R213	220k

Kondensatorer övriga

kond	värde
C101	47 nF
C104	2.2 nF
C105	2.2 nF

Dioder

diod	typ
D101	1N4148
D102	1N4148
D201	1N4148
D202	1N4148
D203	1N4148

Transistorer

transistor	typ
Q101	BC547
Q201	BC557
Q202	BC557

Övrigt

8-poliga IC-socklar	3 st
kontakter	se text
omkopplare	se text
låda	se text
gängade distanser	se text

Kondensatorer dropptantal

kond	värde
C102	10 uF/25v
C103	10 uF/25v
C201	2.2 uF/35v

IC

ic	typ
IC101	1458
IC201	TL1 111
IC202	1458

FOXYCOPY

Äntligen har vi fått ett ABC-program som motsvarar COPY II PC! Foxy betyder slug, listig eller rävaktig. Programmakarna bakom FOXYCOPY vill dock inte använda översättningen rävaktig utan föredrar att tänka på en kopiering som är slug eller listig!

Rävaktig?

FOXYCOPY är ett seriöst program som är avsett för att du skall kunna ta en backupkopia på dina kopieringsskyddade disketter. LLBC Datakonsult som säljer FOXYCOPY var lite tveksam till att marknadsföra programmet, men tveksamheten försvann när Nokia lade ner produktionen av ABC806 och ABC802.

Hur sker kopieringen?

FOXYCOPY kopierar en hel diskett i drive 0 till en ny diskett i drive 1. Olika versioner av FOXYCOPY finns för olika typer av flexskivor och det går inte att kopiera från en typ av diskett till en annan typ. Programmet frågar om man vill kopiera 1 eller 2 sidor eftersom kopieringen blir så mycket snabbare om enbart en sida kopieras. Du startar kopieringen genom att ange "!", vilket jag dock tycker verkar vara en lite konstig form att starta ett program på.

FOXYCOPY markerar varje spår som kopieras på bildskärmen med en fylld ruta. Verifiering sker och fel vid läsning eller skrivning markeras med "X". "*" markerar att skivan är skyddad med laserhål, detta simulerar FOXYCOPY enligt uppgift på något sätt!

Det kan bli fel vid kopieringen på grund av ett kopieringsskydd eller av att orginaldisketten är skadad. Användaren rekommenderas enligt bruksanvisningen att testa den första skärhetskopian och att formatera backupdisketten före kopieringen.

Kontrollerkort?

Programmet klarar såväl DIAB/Luxors gamla som nya snabba kontroller. MYAB:s UNIDISK kan ej användas. En version för MYAB:s kontrollerkort uppges vara planerad, men det tycks vara osäkert om man kommer att ta fram en UNIDISK variant eftersom efterfrågan för detta verkar vara liten. De flesta som har köpt UNIDISK har i flera fall sitt gamla kontrollerkort kvar och kan använda det för att ta fram säkerhetsskopior.

DIAB/Luxors snabba kontrollerkort måste vara bestyckat med 8 KB minne för att FOXYCOPY skall fungera. Detta innebär att FOXYCOPY inte fungerar på diskettstationer av typ Tranfor med DIAB:s nya snabba kontrollerkort eftersom Tranfor bestyckar korten med 2 KB minne. Luxor sålde i början också de nya kontrollerkorten med 2 KB minne såvitt jag har hört, men gick sedan över till 8 KB minne.

Har man en snabb DIAB/Luxor kontroller med 2 KB minne så kan man enkelt byta minneskapseln till 8 KB minne och ställa om en bygel på kortet så att kortet upptäcker att det finns mera minne.

FOXYCOPY klarar version 1.05, 1.06 och 1.08 av kontrollerprommet till DIAB/Luxors snabba kontrollerkort och i de flesta fallen också version 1.07. Version 1.07 uppges dock vara så behäftad med buggar att alla rekommenderas att byta ut detta snarst!!!

Program som kan kopieras enligt LLBC Datakonsult:

ORD III, Basregister, Bokföring 800, Redovisning 800, Kalkyl 800, IDA, Tidsredovisning 800, Toolbox, CUTE, DIAGRAM II, CompSeOm, Administration 800, Skydd 800, Assembler 800, Multibas, ORD 800, Bokföring III, Redovisning II, REG 800, Protax, CompRegina, Teledata 800, ABCUTE, Bild 806, OFL 800, Administration II, Squeez och alla motsvarande Facitprogram.

FOXYCOPY själv är inte kopieringsskyddat, men varje program är märkt med innehavarens namn och serienummer.

Övriga hjälpprogram

På disketten finns några hjälpprogram också, förutom själva FOXYCOPY.

VERIFY kan jämföra två disketter. LLBC rekommenderar att man gör detta på den första säkerhetskopian. Skulle inte backupkopian fungera kan man begära att VERIFY korrigerar felen.

SKIVTEST kontrollerar en diskett på samma sätt som brukar ske vid efter formatering.

SYSOP är till för att köras för att ge programmakarna information ifall FOXYCOPY inte fungerar på viss utrustning.

Slutsats

Jag kan rekommendera programmet för säkerhetskopiering av kopierinsskyddade disketter. Jag har testat programmet och funnit att det fungerar utan anmärkning på min utrustning som består av en ABC806:a och en ABC834. Jag har dock bara testat det på något program eftersom jag inte har tillgång till så många kopieringsskyddade program.

En säkerhetskopia gör att man inte blir beroende av leverantören för att få en ny diskett om orginalet förstörs. Detta kan vara speciellt viktigt om leverantören inte finns kvar!

Noteras kan att kopieringsskyddade ABC830 disketter kan kopieras med programmet COPY II PC på en IBM PC eller en med IBM PC kompatibel maskin.

<1789>
Bo Kullmar

GRAF806

För ett tag sedan fick jag tag i denna underbara tillsats till ABC806, som jag varmt kan rekommendera alla som äger en 806:a och intresserar sig för grafik.

Normalt har ju 806:ans grafik få kommandon och den är inte så snabb heller, det vet alla som någonsin har letat efter ett sätt att göra sprites utan att ha fonderade kunskaper i bildminneshantering och maskinkod.

GRAF806 ger Dig nu 9 nya BASIC-kommandon, 40 assembler-entries i hopptabell och om Du kör ett litet program (EXTBASIC.BAC) som medföljer laddas ytterligare 4 BASIC-instruktioner i RAM. På köpet får Du timeout på PR: och V24:.

Programmet har körts buggfritt i över ett år och är fullt kompatibelt med all annan programvara (som BILD806). Det snabbar dessutom även upp exekveringen av de befintliga BASIC-kommandon.

Även om grafik inte är så lätt att beskriva i ord, vill jag göra ett försök och presentera de nya kommandona. Vad man kan göra med dem, ser man snyggt på den medföljande demo-disketten. Redan den är värd nästan hela priset.

FGCTL MIX

Du kan blanda färgerna (två och två), så att Du får en större skala färger att välja emellan.

FLASH/NO Flash

Du får en blinkeffekt där Du kan byta mellan 2,3 eller 6 färger.

FGCLR

tömmer grafikminnet.

PROCEDURE - DO

kan Du anropa en funktion i stället för det knöliga DUMMY=FN...

MIXLINE/MIXSTATUS

Du kan rita streckade linjer i blandade färger. Om Du lägger FLASH till detta, kan Du få en effekt som gör att linjerna ser ut som rör med rinnande vätska i. Det är ytterst effektivt om Du skall göra ritningar som skall visualisera flöden: VVS-installationer, fabrikationsschemata eller funktionen av Din destillationsapparat.

FGSPEC

ger Dig den fascinerande möjligheten att rita något på en bild utan att förstöra den underliggande bilden - på detta vis kan Du flytta en sprite över en bakgrund. Du kan t o m låta spriten gå delvis framför delvis bakom delar av den existerande bilden.

FGDRAW/FBRUB

med dessa två kommando kan Du rita en figur resp sudda bort den. Det tar visserligen ett tag innan Du har gjort en figur - ett program medföljer visserligen som gör det enklare, men Du måste ändå definiera hela figuren. Men sen går det fort, det går så fort att ens gamla 806:a knappt hinner med.

Paketet

Detta lilla paket, som förr salufördes av T-D-X, består av ett prom, en demodiskett och en manual på 26 sidor som förklarar allting uttömmande och lättfattligt. Och som sagt demodisken är redan värd pengarna.

Numera kan Du som ABC-medlem få hela paketet för det facila priset av 175.00 SEK, som Du sätter in på postgiro 86 05 25 - 5. Det är Anders Berg, Väktargatan 10 B, 754 22 Uppsala, 018 - 13 79 89 som har ställt det till klubbens förfogande. Kom ihåg att ange på postgiroblanketten vilket diskett-format du vill ha.

J C Liebe-Harkort

"Tål du den där så tål du den här"

Hur långt räcker strömförsörjningen i en ABC80?

Den första ABC80 hde 16 kB internminne och 40 teckens skärmbild. Sedan dess har många ABC80-ägare byggt ut till 32 kB och lagt in 80-teckenskort. Det första var nog förutsett vid konstruktionen av strömförsörjningen, men troligen inte det andra. Hur det var gick det ändå bra.

Man kan också hänga på numeriskt tangentbord, Superbasic eller Smartaid, anpasningskort för CATNET osv. Alla dessa extra tillsatser skall också få sin strömförsörjning från samma aggregat. Till att börja med går det bra, men en vacker dag knäcks kamelens rygg av den sista stickan.

För mig inträffade detta första gången när vi hängde på ett specialtangentbord för en handikappad och datorn ibland började uppföra sig mycket konstigt.

För något år sedan lät jag sätta in ett Gejo extra RAM-disk på 64 kB. AVsikten var att kunna ladda in BASICII och kör apparaten som RAM-maskin, vilket andra hade lyckats med. Men för mig gick det dåligt. Det blev skräptecken på skärmen, då och då föll man ur och måste starta om. Ibland låste sig hela maskinen.

Så småningom kom det fram att detta med all sannolikhet berodde på att jag hade den "nya" ABC80, men Philips monitor och "mellanlåda" med kraftförsörjningen. Det tycks vara så att denna låda ger mindre "kräm" än transformatorn i originallådan. När jag klagade min nöd i ABC80-mötet i msg-systemet, kom följande inlägg från Tomas Wikström <1398>:

"Den 'lilla mellanlådan' har som alla numera vet lite svårt att orka leverera ström då man har 'hängt på' alla fina tillbehör som finns på marknaden. Lite bättre går det med den 'gamla' originalmonitorn som har ett lite kraftigare supply.

Men det finns ett par enkla sätt att fixa till sin 'mellanlåda' så den blir t o m mer kraftfull än monitorsupplyet. Inte blir det så väldigt dyrt heller, ca 300:- inkl moms för att en gång för alla se till att man slipper pling, plong och allsköns skräp på skärmen med förlust av eventuell minnesinnehåll som följd.

Den här ändringen som jag skall föreslå innebär att man jobbar med, och ändrar delar där nätspänningen, 220 V, huserar. Vad det innebär med risk för liv och lemmar, hoppas jag alla tänker på om ni följer mitt exempel här nedan:

Först, skaffa från Deltron, Valhallavägen, Stockholm följande :

1 st trafo, 1*8.5V (8 A !!), 2*16 V(1.7 A)/120 VA.
1 st 25 A likriktarbrygga, best nr 12-6340.

Dessa går, utan problem, att få plats med i lådan. Trafon är en ringkärna och passar inte direkt på kretskortet, men med en platta av kretskortslaminat i hålet efter den gamla trafon så får man den att sitta fast säkert med ordentliga avstånd till alla strömförande delar. Likriktarbryggan passar precis 'på golvet' framför glättningskondingen. Den byts för att få en brygga med lägre Ri, dvs inre resistans, vilket innebär att den släpper igenom strömmen lättare. Dessutom blir den brygga som sitter som original ganska 'stressad' och kommer sannolikt att lämna in med tiden, speciellt eftersom den inte har någon kylhjälp.

Nästa steg är att angripa själva tangentbordet, där den faktiska orsaken till dykningen finns!!!

Det som händer är att datorn drar ström lite ojämnt, och mäter man upp strömmen i den två skilda +5volt-grena som finns, ser man att med 80TKN, Supersmartaid, Gejo ramdisk(128K el. 256K), 32K extra osv. så ligger ström uttaget på ca 1 A på den ena grenen och 1.2 A(!) på den andra, Gejo's egen regulatorgren bortglömd då den klarar belastningen vid både 128K-alternativet såväl som 256K.

De regulatorer som sitter som original klarar enligt spec 1 A, sedan börjar de dra ned strömmen med spänningsfall och dykning som följd.

Det fixas genom att man på Elfa för några tior köper likadana regulatorer (7805) fast den varianten som klarar 2 A och bara helt enkelt byter ut de gamla. Det ger förutom att man nu klarar de belastningskrav som finns (och kan komma att finnas ?), också en annan liten bieffekt: Kylflänsens normala(?) glödheta temperatur blir avsevärt lägre, utan risk för att man bränner fingrarna.

Man kan hos en del maskiner råka ut för att de inte vill gå stabilt i det spänningsfönster (4.75 till 5.25 volt) som TTL vill (skall) ha. Då kan man sätta en liten kiseldiod i serie med referensbenet på regulatorerna (den som går till jord). Detta höjer 5-volten till ca 5.6 volt, men detta är inget problem då det i TTL's spec står att det garanteras en breakdown-voltage på 7.5(?) volt.

Med denna totala modifiering så har man en gång för alla, och för en billig penning (ca 300:-) undanröjt de problem som uppstår med nätaggregatet (det lilla). Efter detta (man grejar det lätt på en kväll) så kan vi som gjort det börja kalla den gamla bildskärmen för 'det lilla bildskärmsaggregatet' då 'lådan' efter ändring blir ungefär dubbelt så kraftfull som bildskärmssupplyet !!!!!!

En sista liten sak bara. En ringkärnetrafo drar betydligt mer ström i själva startögonblicket varför en uppsäkring till det dubbla på primärsidan kanske blir ett måste. Dessutom bör man byta den 3.15 A säkring som sitter för 5-volten till en på kanske 6.3 AT.

Adresser o telnr till de leverantörer som jag föreslog hittar ni säker genom att fråga runt i bekantskapskretsen då de är bland de största och mest kända i hela landet. Naturligtvis finns chansen att ni lokalt kan få tag i motsvarande delar."

Sedan detta referat färdigställts kom Tomas med ett tillägg:

(Text 3221) Tomas Wikström <1398>
Ett litet tillägg, bara, till mitt tidigare inlägg :
En sak, som jag glömde ta upp, som är mycket viktig i sammanhanget är att få grepp om hur mycket rippel man har från strömförsörjningen. Rippel är en benämning på den 50 Hz (och de övertoner därav, främst 100 Hz) som 'rinner igenom' likriktarbryggan och 'förorenar' den matningsspänning som datorn kräver skall vara ren och stabil.

Detta rippel kan, i en utbyggd ABC80 med den lilla nätdelen, bli mycket hög, jag har mätt upp till 2 volt topp-till-topp. Ett acceptabelt värde är nedåt 50 mV (eller 0.05 V).Det som då inträffar när ripplet blir högt är att det för det första stör regulatorerna och orsakar ökad värmeutveckling i dessa och de kretsar som ligger efter, dvs i datorn. Om ripplet blir tillräckligt högt kan det till och med hända att det 'sätter bittar' i logiken och till slut kan inte mikroprocessorn klara av att hålla ordning i det sammelsurium av felaktigheter som uppstår.

Vad är det då som orsakar ripplet och hur får man bort det ? Rent generellt blir det högre rippel när man belastar ett nätaggregat mer, eller om trafon är underdimensionerad för uppgiften.

I nätdelen så sitter, direkt efter likriktarbryggan, en glättningskondensator som bl a har till uppgift att filtrera bort ripplet. Man bör för det första se till att den är frisk och försöka att i nätdelen hitta plats att parallellkoppla fler kondensatorer för bättre filtrering. Som standard sitter där en kondensator på 10000 mikroFarad men det skadar inte att försöka flerdubbla den siffran. Det går att få plats med en del, jag har med nya, lite mindre (i fysisk storlek) kondensatorer fått in hela 60000 mikroFarad i min nätdel, låt vara med 'glidmedel och skohorn', men det gick! Man kan också, om man har en komposant av högre frekvens, ansluta en kondensator med mindre kapacitans (ca 0.1 mikrofarad) som är bättre på att filtera höga frekvenser.

Sedan bör man också försöka ansluta ett par rejält stora kondensatorer inuti själva ABC80'n. Delvis får man ytterligare en filterverkan, men man får också ett stort skydd mot nätstörningar och man kan kanske till och med klara korta strömavbrott! Försök gärna hitta plats för två stycken 10000 mikroFarad att ansluta före regleringen för att inte mörda regulatorerna.

En fundering jag gjort är att sätta samman ett helt nytt nätaggregat med den video-'blandare' som behövs för att man skall kunna ansluta en monitor till sin ABC80. Den skulle i så fall dimensioneras så att alla problem en gång för alla skulle vara lösta. Kostnaden skulle, tror jag, stanna under 1000-lappen i stycketal, skulle vi gå ihop några blir det billigare, förstås.

Det finns numera nya, ändrade regler för S-märkning. Är det någon som vet vad som gäller idag, t ex för hembyggen ?
/tomas
(Text 3221)

Det här sista kanske är en extra komplikation för ett hembyggarblåbär som jag, men nu vet vi i alla fall vad som går att göra.

Lycka till, alla hembyggare!

<1384>
Sven Wickberg

Profiler på årsmötet

Kortfattat sammandrag från ABC-klubbens årsmöte 1987-02-28.

Som underlag har använts årsmötesprotokollet, vänligen ställt till förfogande i maskinläsbar form av Bengt Sandgren. Vi hoppas kunna komma tillbaka i nästa nummer med en fylligare rapportering från ABC-dagen.

På förmiddagen hade den utsällning som brukar gå av stapeln på ABC-dagen öppnat. Några leverantörer visar då nyheter och tillbehör mm till datorer.

Årsmötesförhandlingarna hölls som vanligt i Brommasalen, Gustavslundsvägen 168, som ligger i samma hus som klubblokalen.

Årsmötet öppnades av ordförande Stig Löfgren som hälsade de 62 närvarande medlemmarna välkomna.

Gunnar Tidner valdes till mötesordförande och Bengt Sandgren till mötessekreterare. Till justeringsmän och rösträknare valdes Sten Cederholm och Sten Staxler. Man konstaterade att kallelse till årsmötet hade gått ut i ABC-bladet 4, 1986 och att mötet därför var behörigt utlyst.

Man beslöt att fastställa styrelsens förslag till dagordning för årsmötet samt att diskutera ABC-klubbens framtid under en ny punkt innan budgetdiskussionen.

Man beslöts att lägga styrelsens verksamhetsberättelse till handlingarna med några få ändringar och tillägg

Revisor Lars Gattberg, Bohlins Revisionsbyrå, meddelade att revisionen ej kunnat genomföras då samtliga erforderliga handlingar ej förelegat i tid.

Årsmötet beslöt att under förutsättning att revisorerna kommer med en "ren" revisionsberättelse med tillstyrkan om ansvarsfrihet för styrelsen denna ansvarsfrihet skall anses ha beviljats av årsmötet. Beträffande fastställande av balansräkning beslutade årsmötet att följa de rekommendationer som revisorerna ger i sin revisionsberättelse.

Efter en lång och livlig debatt beslutade årsmötet att ge styrelsen i uppdrag att se över forumleringen i stadgarnas par 1 och framlägga förslag innebärande att klubben öppnas för andra datorer.

Beslöts i enlighet med styrelsens förslag att faställa medlemsavgiften för 1988 till 190 SEK för seniorer respektive 130 SEK för juniorer.

Motion nr 2 från ABC Öst behandlades och förklarades besvarad med det fattade beslutet.

Årsmötet beslutade att fastställa den av styrelsen föreslagna budgeten att lända styrelsen till huvudsaklig efterrättelse.

Beslöts att välja Stig Löfgren till ordförande och Torsten Ljungström till vice ordförande.

Beslöts att välja Bo Kullmar, Jan Holmberg, Jan Liebe-Harkort, Tom Sjöberg, Kent Berggren och Kjell Brealt till styrelseledamöter.

Beslöts att välja Ulf Hedlund, Jonas Klackenborn och Jaan Tombach till suppleanter.

Beslöts att välja Kjell Järbin och auktoriserad revisor Lars Gattberg, Bolins revisionsbyrå, till revisorer samt Karl Lindström till revisorssuppleant.

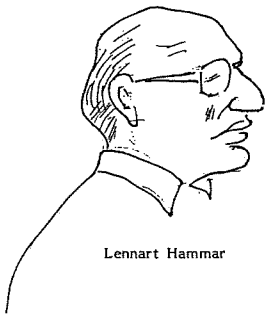
Beslöts att välja Gunnar Tidner, Joe Johnsson och Göran Sundqvist till valberedning med Gunnar Tidner som sammankallande.

ABC Öst hade i motion nr 1 föreslagit ändrade regler för bidrag från ABC-klubben till lokalföreningarna. Årsmötet beslöt i enlighet med motionen. (Det nuvarande startbidraget om 5 SEK per ABC-medlem i regionen kvarstår).

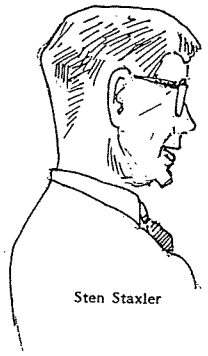
Stig Löfgren utdelade gratifikation till Anders Olsson <1019> för programmet "ABCDISK".

Gunnar Tidner avslutade årsmötet kl 16.50.

ABC-dagen avslutades med subskriberad traditionell middag varvid dagens händelser och klubbens framtid diskuterades.

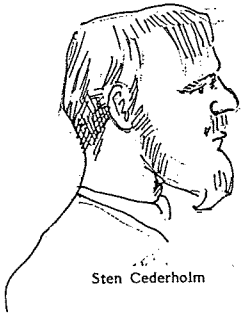


Lennart Hammar



Sten Staxler

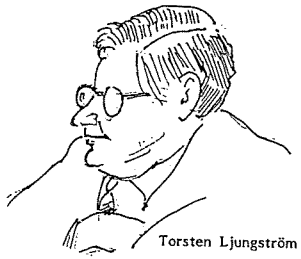
Gunnar Siedler



Sten Cederholm

Jaan Toombach

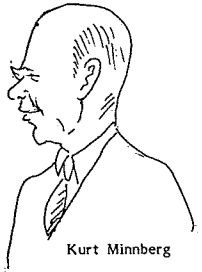
Teckningar: Per Arne Säwén



Torsten Ljungström

Bengt Sandgren

Joe Johnson



Kurt Minnberg

Johnny Isberg

Karl Lindström

Benny Löfgren

Christer Weinigel

Anders Johnson

Ulf Sjöstrand

Funktioner för oss blåbär

LITEN INLEDNING

Den senaste tiden har man kunnat läsa en hel del om flerradiga funktioner för BASIC II i denna tidning. Inspirerad av dessa artiklar har jag försökt utröna vad som egentligen händer i BASIC-tolken när man använder funktioner. Tyvärr äger jag "bara" en ABC80, men man kan komma ganska långt även med en sådan. Om man vet hur det funkar i ABC80 är steget inte långt till BASIC II.

I denna artikel gör vi en utflykt till BASIC-tolkens sköna värld. Låt oss börja med tolken i ABC80. Det är den enda jag har tillgång till så det enda som är garanterat sant i denna text är allt som handlar om ABC80. Jag tror att även ABC800-ägare kan ha nytta av innehållet i denna text.

För övrigt hoppas jag att eventuella proffs inte sätter melonkärnor i halsen när jag påstår att man kan likna flerradiga funktioner med subrutiner. En subrutin är ju en rutin som utför någonting och som placeras utanför själva huvudprogrammet. Och så är det ju också med en funktion. Sedan är det en annan sak att man anropar subrutiner på olika sätt, t ex med GOSUB eller Z=FNA. Det är praktiskt om subrutinen kan använda variabler som inte påverkar anropande programsnutt. Man kan nog säga att en flerradig funktion är ett snyggare sätt att skriva subrutiner på än med GOSUB-RETURN.

ENRADIGA FUNKTIONER

Låt oss börja med den enradiga funktionen, föregångaren till de bekanta flerradiga. En enkel sådan kan se ut så här:

```
10 DEFFNA(X)=X+1
20 ; FNA(1)
```

På rad 10 definieras själva funktionen och nästa rad innehåller ett anrop av den. Om man exekverar detta program finner man att talet två skrivs ut på bildskärmen. Man kan också ha en funktion som ser ut så här:

```
10 DEFFNA(X$)=ASC(X$)
20 ; FNA('APA')
```

Nu skriker många till och säger att det kan man visst inte. Kanske säger ni att strängar är förbjudna i funktioner på ABC80. Lugn gott folk, det är faktiskt tillåtet att använda strängar som argument till en funktion. Nu har vi infört ett nytt begrepp, nämligen ordet ARGUMENT. Med argument menar vi hädanefter de parametrar vi skickar in till en funktion. Ordet argument är i denna text likvärdigt med ordet INPARAMETER.

Jojo, strängar kan man ha som argument. Men vad är det då som inte fungerar vad gäller strängar och funktioner i ABC80? Om vi kör det sista exemplet med RUN skrivs talet 65 ut på skärmen. Detta tal kallar vi för SVAR. Ett svar är det resultat en funktion lämnar. I ABC80 är det så att ett svar måste vara numeriskt, svaret får inte vara en sträng. Vi säger att svaret måste vara av TYPEN tal. En TYP är den form data presenteras på. I ABC80 finns det tre typer: heltal, flyttal och strängar.

Nedan ser vi ett exempel på en otillåten funktion i ABC80. Det går utmärkt att skriva in programraderna men när vi skriver RUN protesterar maskinen med ERR 8, finns ej i detta system.

```
10 DEFFNA$='APA'
20 ; 'R'+FNA$
```

I ett funktionsanrop kan vi skicka med upp till fyra argument. Så här kan man göra:

```
10 DEFFNA(X1,X2,X3,X4)=X1+X2+X3+X4
```

SAMMANFATTNING

Med denna genomgång av kanske elementära saker har vi kommit fram till att argument får vara av valfri typ, vi kan ange max fyra argument i ett funktionsanrop och att svaret inte får vara en sträng.

När man börjar dissekera saker i dess minsta beståndsdelar är det viktigt att använda en terminologi som är konsekvent och inte kan missuppfattas. Vi har här raskt infört tre termer som är viktiga för vidare resonemang: ARGUMENT, SVAR och TYP.

LOKALA VARIABLER

Låt oss sjunka djupare ner i ABC80-tolken. Betrakta följande programsnutt:

```
10 DEFFNA(X)=X+1
20 X=100 : Z=FNA(3) : ; X
```

Hör och häpna, om man kör detta program kommer talet 100 att skrivas ut på skärmen. Men varför skulle nu detta vara så märkvärdigt? Låt oss undersöka dessa rader. Programrad 10 är endast en definition av en funktion och kommer alltså inte att utföra någonting vid RUN. Rad 20 är däremot mer intressant. Först så tilldelar vi variabeln X värdet 100. Därefter tilldelas Z svaret av ett funktionsanrop med ett argument. Slutligen skriver vi ut värdet av X på skärmen. Kör vi programmet skrivs talet 100 ut. Men X används ju i funktionen som anropas innan utskriften av X. Förstörs inte värdet i X när funktionen använder den? Nej!

Denna lustighet beror på att variabler som används som argument i funktionsdéfinitioner är LOKALA. Att en variabel är lokal medför att när BASIC-tolken vill hämta dess värde, kanske för att kunna genomföra en addition, så tas inte detta värde från den vanliga variabellistan. Värdet på den vanliga variabeln X kommer alltså inte att ändras vid funktionsanrop.

FIXRUTINEN

Uj, här måste det förklaras vad en variabel egentligen är för någonting. Antag att vi knappat in ett litet program, t ex de två raderna som vi diskuterade ovan. Om vi nu skriver RUN tror kanske många att programmet direkt börjar exekveras. Fel! Det första som händer är att programmet FIXAS. Som bekant omvandlas en BASIC-rad, så fort vi skrivit in den, till en massa konstiga koder som bara tolken förstår. En heltalsvariabel upptar fem bytes i en programrad och ser ut så här:

```
193,n,n,x,x
```

Vi behöver inte nämna vad koderna betyder, bara säga att två bytes av dessa fem är en pekare som direkt pekar ut en adress i minnet där variabelns värde ligger. Det är fix-rutinens jobb att se till att det reserveras minne där variabelns värde kan placeras samt att lägga adressen till denna area i programraden, alltså fixa pekaren. När fix-rutinen har gjort sitt finns alltså alla variabler, som programmet använder, i datorns minne. Inte nog med det, alla variabler är dessutom nollställda. En trevlig prick, den där fixaren. Nu är programmet klart att börja exekveras!

Det här med fixning är inget som vi behöver tänka på. Det sker helt automatiskt av BASIC-tolken när vi skriver RUN.

MER OM LOKALA VARIABLER

När tolken vill hämta ett värde på en vanlig variabel i ett program för att utföra någon beräkning finns det alltså en direktpekare i programmet som pekar ut variabelns värde. Det är en bidragande orsak till att BASIC-en i ABC80 är så snabb.

För att göra det mer lättförstått kan vi jämföra med en sur chef som när som helst vill kunna se vad sina anställda gör på fritiden. Han lämnar helt enkelt adressen hem till den anställde till en springpojke som genast ilar dit, fotograferar av den anställde samt återvänder med bilden till chefen. Den sura chefen motsvaras i datorn av programmet, springpojken är BASIC-tolken, husen de anställda bor i är variabler (som ju ligger på ABSOLUTA ADRESSER) och de anställda är värdet på variablerna. Om en anställd tröttnar på chefen och flyttar till en annan stad får en variabel ett nytt värde. Med absolut adress menar vi ett ställe som vi direkt kan nå med en unik adress.

Men vad gör då fixrutinen, den som jobbar efter det att vi skrivit RUN och innan själva programmet börjar rulla, med en variabel som används som argument i en funktion? Eftersom den skall vara lokal i funktionen (får alltså ej påverka den riktiga variabeln X utanför funktionen) får inte fixrutinen lägga en absolut adress till variabeln X där den används i funktionen. Istället används indexerad adressering. När en funktion anropas placerar tolken inkommande argument i en tabell och numrerar den från noll och uppåt. När vi refererar till ett argument i funktionen placerar fixrutinen argumentets ordningsnummer i de bytes där normalt annars den absoluta adressen (som pekar ut variabelns värde) ligger.

Jämför med den sura chefen. Istället för att han lämnar en adress till springpojken säger han "Spring och fotografera anställd nr 2!".

När BASIC-tolken alltså vill ha värdet av en lokal variabel (dvs ett argument) får han ett nummer av programmet. Tolken rusar till sin tabell med argument och räknar sig fram till den önskade. Observera att det tar längre tid att hitta nr tre än vad det gör att hitta nr noll (glöm inte att vi började numrera från noll). Det är viktigt att veta att det generellt tar längre tid att hämta en lokal variabels värde än en vanlig variabels värde.

Men när skapas då den här tabellen där de lokala variablerna hamnar? Som vi sagt tidigare skapas alla vanliga variabler av fixrutinen direkt när vi skriver RUN. Men de lokala måste skapas dynamiskt varje gång vi anropar en funktion. Vid ett funktionsanrop staplas alla argument upp i en tabell. Sedan kan funktionen räkna och ha sig. När detta är klart sker uthopp ut funktionen med ett svar. Samtidigt tas tabellen med argumenten bort från minnet så att de inte ligger där och dräller.

REKURSIVA FUNKTIONER

Kan en funktion anropa sig själv? Denna fråga låter kanske lite underlig men titta då på detta exempel:

```
10 DEFFNA(X)=FNA(X+1)
20 ; FNA(3)
```

När vi kör detta program får vi efter en liten stund ERR 3, minnet fullt! Låt oss se vad som händer när vi skriver RUN. Vår vän fixrutin tar tag i händelserna och lägger numret på argumentet X (som ju är noll) överallt i DEFFN-satsen där den används. Fixaren har massor av hyss för sig men vi kan inte gå in på alla här. Därefter startar exekveringen av programmet. Rad 10 hoppas över och rad 20 attackeras. Funktionen FNA(X) anropas med ett argument som placeras i en tabell (det blir en liten tabell med bara ett värde!).

Så långt är allt grönt. Nu ska uttrycket i DEFFN-satsen beräknas. Aha, tänker tolken, ett funktionsanrop! Bäst att beräkna värdet av argumentet först. Variabeln X är tydligen lokal för här finns ingen absolut adress till variabelns värde, här ligger bara en siffra (en nolla). Efter lite letande i (den lilla) tabellen hittas värdet tre. Nu beräknas 3+1 vilket blir fyra. Eftersom det är ett funktionsanrop måste en ny tabell med argument skapas. Reservera minne, skapa tabell osv...

Eftersom det skapas en ny tabell varje gång funktionen anropar sig själv kommer så småningom allt tillgängligt minne att ta slut, vi får ERR 3. Aha, tänker den klurige, låt oss strunta i alla argument. Utan argument så blir det ju en tom tabell! Och en tom tabell är ju ingen tabell alls?!

```
10 DEFFNA=FNA
20 ; FNA
RUN
```

Tyvärr, det blir också ERR 3. BASIC-tolken måste ju veta vad han ska göra när fuktionen är utförd och sparar därför undan lite information som också tar upp minnesutrymme. För varje nytt funktionsanrop förbrukas ytterligare lite minne. Det är inte mycket, bara några bytes, men många bäckar små... Jämför:

```
10 GOSUB 10
RUN
```


UPP TILL BEVIS

Fantastiskt, ABC80 är alltså förberedd för rekursiva funktioner, dvs sådana som anropar sig själva. Dessutom finns logik för lokala variabler inbyggt. Det är precis sådana fin esser som finns i ABC800. Kanske har man medvetet minskat på finesserna i ABC80 för att ha något att "komma med" när nästa generation av maskin (ABC800) skulle presenteras? Eller var det så att allt inte var helt utvecklat? Dumt att spekulera, låt oss hellre undersöka verkligheten! Hur vet vi att en funktion verkligen anropar sig själv? Kanske beror det där ERR-3-et på något annat. Låt oss därför titta på detta program:

```
10 X=9007 : Y=47
20 IF PEEK(X)<>213 LET X=9005 :
Y=45
30 POKE 65408,33,160,255,114,43,115,
43,235,195,Y,35
40 DEFFNP(X)=CALL(65408,X)
```

Funktionen på rad 40 skriver ut argumentet på bildskärmen. Låt oss se hur det fungerar. Rad 10-20 är till för att låta X få adressen till den rutin i BASIC-tolken som skriver ut radnr om vi använder TRACE. Eftersom denna rutin ligger på olika ställen beroende på datorns checksumma ser det hela lite knöligt ut. Rad 30 placerar en liten maskinkodssnutt i RAM. Denna snutt anropar TRACE-rutinen på ett snyggt sätt. På rad 40 ser vi en funktion som anropar maskinkodssnutten. Vi kan anropa rutinen med Z=FNP(3) och vips så skrivs talet tre ut på bildskärmen. Låt oss nu lägga till två rader:

```
50 DEFFNA(X)=FNP(X)+FNA(X+1)
60 Z=FNA(1)
```

Rad 50 definierar en funktion som dels skriver ut argumentet på skärmen (med hjälp av FNP) samt sedan anropar sig själv. För att testa det hela anropar vi funktionen FNA på rad 60 med argumentet ett. Notera att variabeln X används på flera ställen men att den är lokal i varje funktion där den används. Om programmet körs kommer först talet ett att skrivas ut. Det sker då funktionen FNP(1) anropas. Sedan anropar funktionen sig självt med argumentet plus ett, dvs två. Talet två skrivs ut osv... Till slut får vi ERR 3. Det går inte att bryta det hela med CTRL-C därför att vi hela tiden utför en BASIC-sats (RAD 60).

FLERRADIGA FUNKTIONER

Nu när vi vet hur enkelradiga funktioner fungerar är steget inte långt till flerradiga. Sådana finns inte i ABC80 men i ABC800. Ett exempel på flerradiga funktioner ser vi i detta program:

```
110 DEF FNA LOCAL X
120    X=20
130    Z=FNB
140    RETURN Z
150 FNEND
160 DEF FNB
170    PRINT X
180    RETURN Z
190 FNEND
200 X=10
210 Z=FNA
```

Exemplet är hämtat från en artikel i ABC-bladet nr 1 1987. Några ändringar är dock gjorda. Rad 120-130 har placerats sist i programmet för att det hela ska se likadant ut som ABC80-exemplen. Dessutom har END-satsen tagits bort då denna inte behövs.

Två funktioner är definierade, FNA och FNB. Som man ser i listningen är det ett mellanslag mellan DEF och FNA. Dessutom är raderna i en funktion förskjutna åt höger. Programmen i en ABC800 är lite lättare att läsa när man skriver LIST än de är på ABC80. En funktion avslutas med FNEND. Uthopp ur en funktion sker med RETURN <svar>. Man ser att det är möjligt att definiera lokala variabler i en funktion. Fortfarande så gäller att argument till en funktion är lokala i funktionen men här kan man alltså definiera ytterligare slask-variabler att använda i funktionen.

Om man kör detta program med RUN skrivs talet tio ut på skärmen. Att det är så kan man läsa i bladet nr 1 1987. I den artikel där detta exempel återfinns häpnar man över detta och menar att BASIC-en är dålig. Så enkelt är det emellertid inte. Låt oss se vad som händer!

På rad 200 tilldelas variabeln X värdet 10. Sedan anropas funktionen FNA på rad 210 utan argument. Funktionen FNA definierar i sin tur en lokal variabel X. Låt oss nu anta att det förhåller sig så att även lokala variabler placeras i en tabell likt argument som anländer till en funktion. Att det är så vet vi inte säkert men det är troligt. Det är i så fall lika bra att placera de lokala variablerna i samma tabell som argumenten, det finns ingen anledning att skilja dem åt! I exemplet har vi inga argument så det blir en liten tabell (nu igen!) med endast en lokal variabel.

Den lokala variabeln X tilldelas värdet 20 på rad 120. Sedan anropas på nästa rad funktionen FNB. I FNB finns inga lokala variabler. Men variabeln X då? Den är INTE lokal trots att den är lokal i anropande funktion. Det som skrivs ut på rad 170 är alltså värdet av orginal-X, dvs talet 20.

Även i en ABC800 sker fixning när vi skriver RUN, precis som i ABC80. Det första fixaren gör är att hitta funktionen FNA. Aha, tänker tolken, en funktion! Nu ska alla efterföljande variabelreferenser till lokala variabler inte få en absolutadress till orginalvariabelns värde utan få ett nummer. Eftersom X på rad 120 är den första (inga argument finns ju) får X nummer noll. När hela FNA är fixad (fixningsrutinen läser en rad i taget sekventiellt från början) kommer funktionen FNB. Aha, tänker fixaren, en variabel som heter X (rad 170). Den är inte definierad som lokal i funktionen, alltså skall den absoluta adressen till variabelns värde läggas i programmet.

Se där, helt plötsligt har vi fått förklaringen till varför en variabel som är lokal i en funktion helt plötsligt blir global (global = icke lokal) i en annan funktion som anropas av den första. Det hela är en kompromiss för att få högsta möjliga fart på BASIC-programmet. Genom att låta fix-rutinen göra så mycket som möjligt minskar man programmets arbetsbörda. Kanske kan man lösa det hela på ett annat sätt men det skulle antagligen medföra att BASIC II blev långsammare.

Observera att det inte alls är säkert att det fungerar så här på en ABC800. Det hela är en gissning men jag tror att den är ganska kvalificerad. Förstår man att man kan adressera data på olika sätt, t ex med en absolut adress eller med ett index i en tabell, så har man lärt sig mycket.

EPILOG

Mmm... Det är riktigt kul att vara ett litet blåbär. Det är ju väl känt att även små växter, som kanske betraktas som ogräs, kan spränga genom asfalt och få vägarna att blomma. Och kanske kommer någon och plockar mig. Blåbär är ju så gott!

Anders Franzén <5258>

De stora talens mystik

Om asciiräkning på en ABC-dator
Många av Bladets läsare har nog hört historien om schackbrädet:

Den som uppfann schackspelet uppfinnare skänkte sin uppfinning till Härskaren (om det nu var kalifen av Bagdad eller Djingis Khan eller vem det kan ha varit), som blev så förtjust att uppfinnaren skulle få önska sig en gengåva. Uppfinnaren bad då om ett vetekorn för första rutan, 2 för nästa, 4 för nästa osv, hela tiden dubbelt upp för nästa ruta ända tills brädets 64:e ruta.

Härskaren blev en smula förnärmad över att uppfinnaren inte bad om en värdigare och större gåva, och han överlät åt sin storvesir att räkna ut hur många vetekorn det kunde bli.

Storvesiren kom snart tillbaka för att tala om att så många vetekorn inte fanns i hela världen, och man kan anta att kalifen nödgades avrätta den oförsynte uppfinnaren, eftersom det var omöjligt att uppfylla hans begäran...

Nog om legenden; hur många vetekorn var det fråga om? Med den matematik jag fick lära mig i det gamla realgymnasiets tredje ring kunde man räkna ut att det måste bli

```
2**64 - 1
```

dvs två upphöjt till 64 minus ett. Man konstaterar genast att det är ett MYCKET stort tal. I binärtalsammanhang stöter man ofta på serien: 1 2 4 26 32 64 128 256 osv ända upp till 65536 som är 2**16, men sedan orkar man inte med längre.

De första miniräknarna var stora som en halv telefon och kostade över tusen kronor. Under några hade jag tillgång till en sådan och låg och försökte räkna ut hur många vetekorn det var. Eftersom apparaten i fråga bara visade 8 siffror, måste uppgiften delas upp i bitar. I slutändan fick jag göra en massa skarvräkningar för hand, men till slut kom det fram ett värde med 19 heltalssiffror som jag sparade för framtida bruk (?!).

Flera år senare hade räknedosorna blivit billigare och bättre. Jag prövade en med inbyggd exponentialfunktion, men fick nöja mig med svaret:

```
1.8447 + E19
```

vilket i vanlig tappning betyder något mindre än 2 gånger 10 upphöjt till 19. Men vilka är de övriga 14 siffrorna? Med några trix som jag nu har glömt kunde man få fram ett par siffror till men man kom inte längre än 1.8446744 E19.

Bland det allra första jag gjorde när jag fick tag i en ABC80-dator var att skriva

```
;2 ** 64%
```

vilket omgående gav det nedstämmande beskedet:

```
1.84402 E+19
```

Färre antal siffror än i miniräknaren, och därtill avvikande resultat redan i femte siffran! Vad skulle man nu tro på?

Vad är sanning?
ABC80 är inte byggd för stora matematiska bedrifter, men den är faktiskt försedd med en möjlighet att klara de fyra vanliga räknesätten med omkring 28 siffror. (Gränsen är flytande på grund av att det måste finnas utrymme för delberäkningar.)

Denna möjlighet - ascii-räkningen - gav lösningen på gåtan. Den innebär kort och gott att talen presenteras som strängar och att datorn räknar med siffrorna per styck, ungefär som man gör med papper och penna. Exakt hur behöver vi som vanligt inte bekymra oss om, det räcker att lära sig kommandona ADD$, SUB$, MUL$ och DIV$. Vad de står för säger sig själv, och syntaxen ser ut så här:

```
T$ = ADD$(A$,B$,0)
```

där A$ är första talet, skrivet som sträng (t ex '2') och B$ är andra talet, skrivet på samma sätt. Nollan säger att svaret T$ skall ges med noll decimaler, varvid man får angivet antal decimaler med korrekt avrundning.

För kalifens schackproblem får man följande lilla program:

```
10 REM SCHACK
20 A$='2'
100 FOR I=2 TO 64
110 A$=MUL$(A$,'2',0)
120 ; I' 'A$
200 NEXT I
```

A$ måste få ingångsvärdet '2'. Observera att den tvåan noteras som en teckensträng. (Man kan naturligtvis ha "dubbelblipp" lika väl som 'enkelblipp': "2" går lika bra, men jag är svag för att utnyttja ABCns möjlighet att skriva strängavgränsare utan shift-tangent)

Räknaren I anger i detta fall hur många faktorer 2 vi hunnit multiplicera, varför vi börjar med 2.

I rad 110 multipliceras A$ (dvs första gången '2', sedermera det senast framräknade värdet) med konstanten '2'. Resultatet begärs ut med noll decimaler och hamnar i det nya värdet på A$.

På rad 120 skriver vi för kontrollens skull ut resultatet av varje varv. (Vill man imponera på sina icke datorkunniga vänner hoppar man över den raden och begär utskrift först när alla varven räknats färdigt.)

Det var med stor spänning jag tog fram papperet med resultatet av mina vedermödor med räknedosan under de där sjukdagarna och jämförde siffrorna. De stämde! Och vi kan genast också avgöra att ETT av de tidigare angivna värdena INTE är korrekt. (Någon annan får skriva en artikel som förklarar varför vissa räknemaskiner är så dåliga på exponentiering...)

(Pedantens anmärkning: För att få rätt antal vetekorn, glöm inte bort att i slutsumman dra bort en! Formeln var ju 2 ** 64 - 1, och det vore ju pinsamt om uppfinnaren fick fel antal!)

Om man misstror gymnasiematematiken kan man i stället räkna med ADD$ och skriva en rutin som fördubblar för varje ruta och lägger ihop med föregående. Det får bli hemläxan för denna gång.

Periodiska decimalbråk
Vi vet från skolmatematiken att bara 2 och 5 går jämnt upp i 10, vilket innebär att flertalet allmänna bråk av typen 1/3 aldrig går jämnt upp utan ger ett oändligt antal decimaler om man förvandlar dem till decimalform.

De flesta (alla?) rationella tal (dit de allmänna bråken hör) ger dock periodiska decimalbråk. Teorien säger att periodens maximala längd för bråket 1/n är n.

Det tål att undersöka, och det hjälper oss asciiräkningen med:

```
10 REM Period
20 ; 'Nämnare:'; : INPUT N$
100 ; '1/'+N$'='+DIV$('1',N$,25)
```

Här väntar programmet först på att man skall mata in den nämare man vill undersöka. Nämnaren tas emot som strängen N$.

Det sökta decimalvärdet beräknas med division mellan talet ett och talet N$ i DIV$('1',N$,25). Vi begär att få se 25 decimaler (mer om antalet decimaler strax).

Men för att utskriften skall bli litet mera upplysande vill vi ha den så här:

```
1/8 = .125000000000000000000000
```

Detta föreskrivs i rad 100.

Nu är det klart för experimentell prövning av teorin. 1/3 och 1/6 ger mycket korta perioder innan decimalerna upprepar sig igen. Prova med nämnaren 7, 13, 19. Nå? Hittar du något mönster för långa decimalperioder?

Fakultet
Ett annat räkneprov gäller de s k fakulteterna. De brukar skrivas:

```
N! = 1*2*3*.....*N
```

Det betyder att 3 fakultet är 6

```
3! = 1*2*3 = 6
```

och 4! är 24, men sedan blir talen raskt mycket långa.

När man skriver ett asciiräkningsprogram för detta måste man använda två parallella beräkningar: en för det tal man skall multiplicera med nästa gång: 1, 2, 3, osv och ett för 1!, 2!, 3! osv

```
10 REM FAKULT
20 ; 'Ge talet:'; : INPUT T
30 A$='1' : B$='1'
40 FOR I=2 TO T
50 B$=ADD$(B$,'1',0)
60 A$=MUL$(A$,B$,0)
70 ; B$+'! = 'A$
80 NEXT I
```

Nästa faktor beräknas i rad 50 till B$ och faktultetsvärdet på rad 60 i A$. Om man funderar litet över programexemplet ger det sig nog ganska lätt.

Antalet siffror
Som redan nämnts har ascii-räkningen har sina gränser. Om man i programexempelt SCHACK sätter I=1 to 100 upptäcker man att det inte går att få ut mer än 28 siffror. Detsamma gäller om man i exemplet FAKULT tar ett värde på T som gör att A$ får mer än 28 siffror.

Nu är 28 siffror VÄLDIGT mycket. Hur stora tal behöver vi? Man anger sin längd till 1,88 m, sin ålder till 36 år, sin bilhastighet till 75 km/tim. För praktiskt bruk tycks ABC80s räknenoggrannhet vara fullt tillfredsställande.

Men den mänskliga nyfikenheten har inga gränser. Hur skall man räkna ut värden med ännu fler siffror än asciiräkningen medger?

Storebror ABC80x kan räkna med dubbel precision och ge 12 siffror i stället för 6 och jag vill minnas att det också finns asciiräkning som ger uppåt dubbla antalet siffror i förhållande till lillebror 80.

Men hur fixar vid ÄNNU fler? Lösningen på det problemet är att skriva en egen rutin som arbetar med ett par siffror i taget och mer eller mindre gör detsamma som vi när vi räknar med papper och penna. Några sådana rutiner publicerades i Mikrodatorn för numera ganska länge sedan, så man kunde ju leta där.

Dock vore det roligare om någon ville försöka skriva dem själv. Hedersplats i Bladet garanteras!

<1384>
Sven Wickberg

Blåbär vara eller inte vara

Jag kan inte låta bli att hoppa in i debatten. Jag älskar att vara med och diskutera (klaga) när alla hoppar på alla, speciellt då Sven Wickberg.

En programmerare som hindrar en annan programerare att utforma saker och ting på sitt sätt är naturligtvis inte bra. Men vad är det för fel att göra en subrutin till en funktion? Jag skulle nog själv göra så, eventuellt att jag skulle fixa så att det blev lokala variabler men inte mer. Åh andra sidan så har jag nog aldrig haft med en subrutin i något enda program jag gjort.

Och blåbär är som sagt inga proffs och behöver oftast inte något 'proffsprogram'. Man sitter vid sin dator och försöker tota ihop något. Det roliga ligger i att upptäcka att gör man si så blir det inte riktigt som man tänkt sig, men hur gör jag för att det ska bli så? Många gånger blir väl inte programmen så välstrukturerade men dom funkar oftast som det var tänkt med ev modifikationer.

Vad gör det då om det är ett bra blåbär som försökt tota ihop några blåbärsfunktioner? Det var bra gjort för det var uppskattat! För vem vet inte bäst vad ett blåbär vill ha om inte ett blåbär själv?

Men å andra sidan så vore det bästa om ett 'Blåbär' och ett 'Proffs' tillsammans satte sig ner och gjorde några BRA blåbärsfunktioner, sådana man har nytta av. Så att man klarar av att göra de vettiga program man vill göra. Och glöm inte bort då att samtidigt förklara vad de olika raderna gör! Det finns inget värre än när man mitt i en rutin inte kan luska ut vad en rad gör, för det mesta klarar Ni av att tala om vad en rutin i helhet gör men det är tydligen inte nödvändigt att förstå vad varje enskild rad gör? Gud vad jag hatade det i början av min 'programmerarkarriär'. För det är inte alla förunnat att ha någon i sin närhet som kan förklara vad :

```
IF ASCII(MID$(Ord$,Y,1))>95 AND ASCII(M
ID$,Y,1)) < < 127 THEN MID$(Ord$,Y,1)=C
HR$(ASCII(MID$(Ord$,Y,1)) AND 223)"
```

utför, eller hur?

Jag hoppas nu att alla reagerar på vad jag har skrivit, mina åsikter är kanske inte helt vettiga men jag tror på dom tills någon har övertalat mig att tvÖru= Qannat.

<6530>
Kerstin Jansson

Marknaden

MULREG
Integrerat programsystem för statistik, databas, diagram för IBM-PC/AT

IDATRON HB presenterar MULREG - ett integrerat programsystem för statistisk analys, databas och diagram. Utvecklingen av programmet sker i Sverige.

MULREG innehåller både enklare och mer avancerad statistik som variansanalys, multipel regression, korrelation, tester, multipla jämförelser etc. det används t ex för analys av mätdata från laboratorier i industrin och i medicinsk forskning.

För grafiken används ANSI-CGI-standard, vilket gör att grafik kan f fås i högsta upplösning och i färg på i stort sett samtliga printrar, plottrar och skärmar, nu och i framtiden. Databasen kan innehålla 500 variabler och 32 000 poster. Formeluttryck och urvalsvillkor kan definieras av användaren.

Programmet är menystyrt men har även möjlighet till batchkörning av en serie beräkningar eller diagram. Inbyggda hjälptexter och felkontroller medverkar till att göra programmet lätt att använda. Manualen om 250 sidor är på svenska och innehåller många exempel. Användarren kan utveckla egna programdelar till MULREG i Turbo-Pascal.

För vidare information kan man hänvänd sig till
IDATRON HB, Krokslätts Parkgata 69 A, 431 38 Mölndal, 031-20 38 33 eller 20 91 29 och 013-28 28 14 84 eller 29 88 91

Nya modem från TGC
TGC har kommit ut med några nya, billiga modem. Det är ett modem för 300 och 1200/75 med hastighetskonvertering i det senare fallet. Autosvar finns. Modemet heter 1100 M och kostar normalt 2550 exkl moms.

TGC 1230 M är avsett för 300 och 1200 och har automatisk hastighetsinställning och uppringare som kan styras från tangentbordet till datorn. Modemet kostar normalt 2990 exkl moms. Båda modemen kan köras som autosvarande modem.

Dessutom säljer TGC Taiwanmodemet Lighspeed. Modemet finns dels i lös låda och dels som PC-kortmodem. Modemet klarar 300 och 1200 och har uppringare som styrs med hayeskommandon. Priserna är 3800 respk 3500 för kortmodemet.

TGC säljer även ett Lightspeedmodem för 2400. Detta är dock ej godkänt av Televerket eftersom de ännu har monopol på 2400. 2400:a modemet kostar 4900 exkl moms.

/ Bo Kullmar

Kamelens andra puckel eller komplikationer för hembyggaren

En ABC80 tillhör i dag veteranerna, i stil med T-forden eller kanske bättre den första VW (med osynkroniserad växellåda som fordrade att man dubbeltrampade...)

Hittar man i dag en urgammal 80, så kan man fortfarande hitta verkstäder som kan sätta in extra minne, 80 tecken, RAM-minne och allt annat vi genom åren uppdaterad veteranen med. Men den tiden närmar sig när allt står och faller med att hembyggaren kan rycka ut och göra modifieringen själv.

Redan får vi också allt fler förslag till ännu djupare ingrepp, t ex förra numrets "kamel"-artikel ("Tål du den så tål du den".) Behovet av hembyggen kommer säkerligen att öka.

I msg-systemet har av och till diskuterats vilka formella krav som riktas mot hembyggen. Det handlar ju om apparater som skall anslutas till det elektriska nätet. För sådana gäller krav på S-märkning, bl a.

I förra numret av Bladet beskrevs hur man kunde bygga om strömförsörjningsburken till ABC80 för att få mera kräm för diverse tillsatser. Slutet på artikeln uttryckte en undring om vad S-märkningsreglerna kunde innebära för komplikationer.

I ett brev till redaktionen ger Carl-Gunnar Hillefors <552> ett axplock av gällande bestämmelser. Carl-Gunnar är radioamatör, och i den intressegruppen har man länge haft notoriska hembyggare, varför amatörradioföreningarna i många år försökt sprida kunskap om bestämmelserna för elektrisk apparatur.

Han påminner om att man ALDRIG får sälja eller överlåta någon hembyggd nätansluten apparatur.

Dessutom, när man går in i en tidigare S-märkt apparatur, så upphäves omedelbart S-märkningen AUTOMATISKT! Därmed blir det o öjligt att sälja datorn i detta ändrade skick - och det lär inte vara möjligt att få den återställd till originalskick hos Scandia Metric eller Luxor...

Till yttermera visso - man är personligen ansvarig för den modifiering man gjort! Skulle det inträffa något, brand till följd av någon kortslutning eller att någon får sig en stöt och "kilar runt hörnet" på grund av den stöten - så kan man faktiskt bli åtalad! Det är ju inte så trevligt, kommenterar Carl-Gunnar.

Om man vill sälja

Den som vill sälja en apparat som skall kopplas in på elnätet måste alltså ha den S-märkt, vilket innebär att SEMKO testar och ev godkänner apparaten. Men det är en omständlig procedur.

För att få den testad måste man lämna in inte mindre än fem likadana apparater för test. Minst tre av dessa kommer att vara helt förstörda efter testen. De blir nämligen isolationstestade med 4 000 volt!

Dessa förstörda apparater får inte slängas på sophögen hur som helst. De måste slås sönder och kasseras fullständigt - man får inte ens ta några delar ur dem.

Ett test kostar i dag 6 000-8 000 kr per testad enhet. vilket betyder uppemot 40 000 kr. Kort och gott: det är knappast troligt att någon kan få ett hembygge S-märkt.

För vidare information hänvisar Carl-Gunnar Hillefors till boken **Grundläggande teknik för radioamatörer** som kan fås hos Sveriges Sändaramatörer, Farsta, Stockholm, och som har en lång förteckning över de säkerhetsföreskrifter som utgivits av Kommerskollegium. Radioamatörerna har som sagt fått brotttas med dessa problem under lång tid.

Carl-Gunnar har dock inte bara dystra kommentarer.

"Däremot kan man själv "snickra ihop" en ganska god nätdel till sin dator och strunta i den medlevererade diton genom at bestämma sig för hur stort strömuttag man maximalt önskar."

"Rekommendabelt är att lägga rätt "kraftig ström" till:

+5 V	8-10 amp
-5 V	ca 4 amp
+12 V	3-5 amp
-12 V	ca 2 amp

och ev en positiv PROM-programmeringsspänning

+12 till +28 volt ca 1 amp."

"Ett eget konstruktionsarbete är alltid en rolig fritidssysselsättning och ger till slut en egen tillsatsnätdel vilken senare också kan användas i andra hobbysammanhang."

Jaha, då vet vi det! Lycka till alla hembyggare! Nu har ni blivit varnade.

<1384>
Sven Wickberg

Ett "proffsblåbärs" små (stora !) problem

Nu har jag lessnat på alla hårda kalla 'fack-artiklar' om programmering och åter programmering, någon enstaka gång har mönstret brutits för en artikel om hur man använder programmet 'GRODAN'. Så jag gör ett försök att bryta alla facktermer. Hoppas att inte alla hoppar på mig, det finns en viss tendens till det när man är lite avvikande.

Att sätta upp en dator är att riskera att man upptäcker att det fattas ett kort eller kanske en kabel i värsta fall en dator. Att programmera är lyckan att se ett eget program växa fram för att falla pladask på magen när programmet blir klart och inte klarar något vettigt. Tänk vad tråkigt det är när strömmen försvinner och allt 'bara försvann'.

Har Ni tänkt på vad alla kassetbandspelare-användare blev glada när disketten kom? Fantastiskt vad mycket som rymms på en diskett och fort går det att söka reda på data. Men en dag är allt borta, fy §§%&* u§$&% vad arg man blir när man upptäcker att inget finns på disketten, man tänker 'jag skulle aldrig lagt gemet där'. För det finns väl ingen som har sådan otur som jag?

En gång för länge sedan så delade ABC80, symaskin, oscilloskop och komponentskåp hörna. Det fanns även en tjej som var (är) lite slarvig. Hon hade gjort det perfekta programmet det som bara fanns på DEN disketten. Nu hör det till saken att det var hennes första 'riktiga' stora funktionsdugliga program hon gjort, det hade tom lite onödiga finesser. Det fanns även en liten 4-årig bror som tyckte om att klippa i tyg, men tyvär så hade han inte lärt sig att man inte ska klippa i det som finns under tyget. Den en gång så stolta tjejen blev en mycket arg och ledsen tjej. Som varade i en hel dag. Sedan började hon på ny kula. Ett par år senare hade fortfarande inget mer 'förolyckats' plötsligt så vill inte den nu gamla ABC80 något mer, den vägrade totalt. Hon tom hotade med ridspöt det hjälpte inte. Det kom en ny ABC802, tyvär blev även konkurensen hårdare, pappa ville också vara med på ett hörn vilket hon skrek högjutt åt. Numera är det mest PC som gäller.

De flesta av oss blir väl medlemmar av egen fri vilja eller hur? Men jag har väl alltid haft maximal otur.. Jag hade en lärare som tyckte att jag satt lite för mycket ute med datorerna så han ordnade fram ett 'stipendium' i form av ett medlemskap i ABC-klubben. Sedan har det gått av bara farten. Det kom ett inbetalningskort och man betalade bara snällt in.

Första gången man skulle logga in i MSG var nervöst, man bara satt och läste. Kom till slut underfund med att det fanns något som hette 'endast xx'. Jag slapp plöja igenom flera hundra texter. Men ack ack den glädjen varade inte länge. När man väl vant sig vid detta forum så kom nästa problem att inte våga ställa frågor det var ju så mycket 'proffs' som körde, tänk om de skrattar åt mina små frågor? Och i vilket möte hör mina små inlägg hemma egentligen? Passar det i MEDFORUM kanske FRITT eller i något annat möte?

<6530>
Kerstin Jansson

Programmeringstips

Efter några års arbete med ABC80 och ABC806 vill jag ge några programtips. En del är körbart med båda maskinerna, men tipsen med olika funktioner i EXTENDED BASIC II hänför sig bara till ABC806. Troligen fungerar de även i andra maskiner i 800serien.

1. Sammankoppling med : i stället för radnummer.

Detta sätt att komprimera programmen är platsbesparande. Varje kolon sparar 2 bytes i minnet. Varning när det ingår IF-satser. Sätt inte kolon efter IF-sats om du inte vet exakt vad som händer! Däremot kan man med fördel avsluta en rad med : IF ---- THEN ----.

Av översiktsskäl kan det ibland vara lämpligt att splittra upp satserna. För att lättare se vart en ON-sats pekar kan man skriva exv:

```
470 ON Ö9 GOTO 500,501,502...,509
500 GOTO 300
501 GOTO 300
502 GOTO 300
.....
509 GOTO 300
```

Man kan nu lätt se vart ON-satsen pekar genom att värdet på Ö9 alltid är radnumret minus 500.

2. THEN och GOTO i IF-satser

ABC-maskinernas BASIC accepterar i allmänhet att THEN och/eller GOTO utesluts ur IF-satser. Man spar 1 byte på uteslutningen, och programmet blir minst lika läsligt. Skriv alltså inte exv:

```
IF F<>0 THEN GOTO 1300
```

Det går precis lika bra med:

```
IF F% 1300
```

Variabeln F är logisk och hoppet kommer att ske för alla värden på F som icke är 0. I detta fall måste man använda THEN i NO EXTEND-mode, eftersom tolken annars läser förbi mellanslaget mellan F och 1300. Man kan komma förbi detta genom att i stället skriva:

```
IF F THEN 1300
```

Procentteckne t gör, att tolken läser F som en varibel. Problem uppstår emellertid om man lagrar programmet i LIST-mode, och sedan försöker ta in det igen. Tolken gör då samma fel på nytt. Så länge man håller sig till SAVE-ade filer går det bra.

I EXTEND-mode kan man skriva:

```
IF F 300
```

3. DIM

Sätt ut DIM-satserna tydligt i början av programmet. Utnyttja möjligheten att dimensionera många variabler i samma DIM-sats.

Skriv hellre:

```
150 DIM A$=20, A(15)=20, Unit$=4
```

än:

```
150 DIM A$=20:DIM A(15)=20: DIM Unit$=4
```

4. Förhandsinställda enhetsnamn, mm.

I långa program är det svårt att hålla reda på alla ställen där man exv. använder printer, om man använder skärmen som fil mm. Definiera därför i början av programmet exv:

```
160 Prunit$='PR:VSA26C70.7':Unit$='CON:'
```

Om man senare t. ex. vill anpassa programmet till någon annan skrivare behöver man bara ändra på ett ställe i programmet.

Ofta använda uttryck, exv: 'Tryck på någon tangent!' läggs med fördel även som variabler:

```
170 Qsl$=' Tryck på någon tangent!'
```

5. Skriva på 25-e raden.

Den 25-e raden i ABC-806 är inte direkt skrivbar i en PRINT-sats. Den scrollas inte heller. Man kan utnyttja denna rad för att skriva felmeddelanden mm, utan att störa innehåller i övrigt på skärmen med hjälp av följande rutin:

```
180 DIM DISP$=0 : POKE VAROOT(DISP$),
    80,0,128,127
```

Felmeddelande skriver man då ut exv:

```
7100 Disp$='Filen '+Fil$+' finns inte
     på skivan. '+Qsl$
```

Glöm inte nollställa variabeln med:

```
Disp$=SPACE$(80)
```

efter det att meddelandet inte är aktuellt längre.

6. Tömma skärmen och skriva rubriktext.

Detta gör man lämpligen med en funktion. Kalla på funktionen med exv: Z=FNCON('MENY')

Funktionen har följande utseende:

```
9100 DEF FNCon(X$) LOCAL X$=160
9102   ; CHR$(12,10,10) STRING$(74,42)
         CHR$(13,10),'    ' X$
         CHR$(13,10)STRING$(74,61)
         CHR$(13)
9104   RETURN
9106 FNEND
```

7. Stora bokstäver

Funktionen FNUpcase$ gör om alla små bokstäver till stora i en sträng. Kallas exv: A$=FNUpcase$(A$).

```
9108 DEF FNUpcase$(A$) LOCAL A$=160,
       B$=160,I,B
9110   I=1 : B$='' : WHILE I<=LEN(A$)
9114     B=ASCII(RIGHT$(A$,I)) : IF (B
         AND 224)=96 B$=B$+CHR$(B AND 95
         ELSE B$=B$+CHR$(B)
9118   I=I+1 : WEND : RETURN B$
9120 FNEND
```

8. Ja/nej

Funktionen FNJa har diskuterats i ABC-bladet några nummer nu. Här är en bra modell, som ger lättfattliga program. Skriv exv:

```
7200 IF FNJa('Skall filen sparas') 5000
     ELSE 300
```

Om svaret är ja hoppar programmet till 5000, om nej till 300. Funktionen FNNej behövs inte. Vill man vända logiken skriver man exv:

```
7200 IF NOT FNJa('Skall filen sparas')
     300 ELSE 5000
```

Funktionen ser ut så här:

```
9910 DEF FNJa(Q$) LOCAL Z$=1,Q$=40,Z
9912   ; Q$ '?  JA/NEJ  '; : GET Z$ :
         Z=ASCII(Z$) AND 95 : IF Z=74 ;
         'ja' CHR$(7) : RETURN -1
9914   IF Z=78 ; 'nej' CHR$(7) :
       RETURN 0 ELSE ; 'Svara J eller
       N !' CHR$(7) : GOTO 9912
9916 FNEND
```

9. Flagga

En funktion kan ha hur många in-variabler som helst, men kan bara returnera en variabel. Om man inte vill sätta en global variabel inuti funktionen utan bara vill sätta ett flag-kritierum kan man använda sig av någon minnescell som inte används av systemet. Om kasett inte används i systemet kan cellerna 65433, 65454 och 65455 användas som flagg-register.

Läsning av flaggorna göres enklast i en funktion:

```
9900 DEF Fnflag = PEEK(65455)
```

I programmet skriver man exv:

```
1000 Z = FNA + FNB + FNC:IF FNFlag 300
```

Hoppet görs då till rad 300 om någon av funktionerna FNA, FNB eller FNC har satt flaggan.

10. PROCEDURE-Funktioner som logiska funktioner

En funktion används ofta i stället för en subrutin för att utföra en viss procedur. Programmet kan då bli lättare överskådligt, om man låter funktionen returnera TRUE (-1) om det blev fel i funktionen, annars FALSE (0).

Skriv då exv. i huvudprogrammet:

```
1200 IF FNA + FNB + FNC + FND ; FNFel
     (FNFlag) : GOTO 300
```

Om någon av funktionerna FNA, FNB, FNC eller FND genererar ett fel, så skriver funktionen FNFel ut ett felmeddelande som bestäms av värdet på flaggan.

11. Lokala variablers räckvidd.

Som påpekats i ABC-bladet nr 1/87 finns bara globala och lokala variabler i BASIC II. Lösningen på detta problem består i att deklarera allting som inte behövs globalt. i en funktion som lokalt, men att de lokala variabler som behövs i en underfunktion 'skickas med' i variabellistan:

```
1000 X=10
1010 Z=FNA
1020 END
2000 DEF FNA LOCAL X,Z
2020 X=20
2030 Z=FNB(X)
2040 RETURN Z
2050 FNEND
3000 DEF FNB(X) LOCAL X 3010 PRINT X
3020 RETURN 0
3030 FNEND
```

Hälsningar <6127>
Sten Öhman

Lagring av data på flexskiva med ABC80

Lagring av data på band och flexskiva

ENKEL TEKNIK
Det kan vara kul att veta hur lagring av data egentligen fungerar på kassettband om man använder en ABC80. Tekniken är fysiskt sett mycket enkel. Får man läsfel avbryts läsningen genast, datorn klarar inte av att göra ett nytt läsförsök. Mjukvarumässigt sett är tekniken också enkel. Det finns ingen logik som matematiskt kan beräkna värdet på en eller flera bitar som är felaktiga. Upptäcks ett fel vid läsning blir man tvungen att starta om från början.

FILER BESTÅR AV BLOCK
Lagringen av data sker i filer. Dessa kan skapas eller läsas i t ex BASIC. Varje fil består av minst två block. Ett block är en del av en fil. Datat delas upp i block för att göra I/O-hanteringen effektivare. Med I/O menas INPUT/OUTPUT, dvs läsning/skrivning från yttre media såsom kassettband. Ett block består av 256 bytes varav man kan använda 253 bytes till egentligt data.

För att ett block skall kunna läsas från ett kassettband fordras en del extra bitar både före och efter blocket. Efter varje block finns t ex en checksumma på två bytes. En fullständig lista över vad som finns före och efter ett block ser ut så här:

- åtta bytes med nollor (ASCII 0)
- tre bytes med synktecken (ASCII 22)
- en byte "start of text" (ASCII 2)
- ETT BLOCK (256 BYTES)
- en byte "end of text" (ASCII 3)
- två bytes checksumma
- en byte med en nolla (ASCII 0)

Synktecknen är mycket viktiga. När man skriver ut en byte på bandet hamnar alla åtta bitarna i en följd med den minst värda biten först. Därefter kommer omedelbart nästa byte. Det betyder att om det ligger flera bytes efter varandra på bandet vet man inte riktigt var en byte slutar och nästa börjar. För att kunna synka in, hitta startpunkten för en byte, används synktecknen. Titta på detta exempel:

```
          v     v     v
00000000000001101000011010000110100O...
```

Så här ligger bitarna på bandet, från vänster till höger. Där pilarna pekar börjar en byte. Hur får datorn reda på det? Jo, genom att leta efter bitkombinationen som motsvarar ASCII 22. Eftersom bitarna lagras med den minst värda först letar alltså datorn efter kombinationen 01101000. Det sker genom att först läsa in åtta bitar. Därefter jämförs det inlästa värdet med det sökta. Om det inte stämmer läses en ny bit in och den först inlästa biten skrotas. Så fortgår sökningen efter den rätta kombinationen. Om inte insynkningen sker korrekt kan man inte läsa datat på bandet.

(Vid datakommunikation mha modem över synkron linje sker insynkning på motsvarande sätt. Vi amatörer kör med asynkron överföring och där används en annan teknik för att komma i fas. Där använder man en startbit som signalerar att en byte kommer.)

Efter synktecknen kommer ett tecken STX (start of text) så att datorn vet när det är slut på synktecknen. Det kan ju hända att insynkning inte sker på första synktecknet av någon anledning. Därefter följer så ett block med 256 bytes. Efter denna kommer en byte ETX (end of text) för att markera slut på blocket. Två bytes checksumma för kontroll att datat i blocket är inläst ordentligt.

FILNAMNSBLOCK
Det finns två typer av block. Det första blocket i en fil är ett filnamnsblock och innehåller endast filnamn. Eftersom ett filnamn kan bestå av max elva tecken plus punkt blir det en hel del utrymme som inte används till något vettigt. Ett filnamnsblock ser ut så här:

- tre bytes med ASCII 255
- filnamn åtta tecken (utfyllt med mellanslag om kortare än åtta tecken) samt tre tecken extension (utfyllt med mellanslag om kortare än tre tecken)
- 242 bytes med nollor (ASCII 0)

När man öppnar en fil för läsning (OPEN) jämför datorn sökt filnamn med det som står i blocket. Är det felaktigt letar datorn upp nästa filnamnsblock för ny jämförelse osv. Anger man inget filnamn till openrutinen skrivs filnamnet i första funna filnamnsblock ut på bildskärmen (FOUND filnamn).

DATABLOCK
Efter ett filnamnsblock följer minst ett datablock. Det är i denna typ av block som filens egentliga data lagras. Ett datablock ser ut så här:

- en byte filnr (alltid noll vid lagring på band)
- två bytes blocknr (alla datablock numreras från noll och uppåt)
- 253 bytes med data

Det finns två typer av data som man kan lagra i en fil, antingen kan man spara textfiler eller så kan man spara binärfiler. Exempel på textfiler är vanliga texter skrivna med en editor eller BASIC-program sparade med LIST. Binärfiler är t ex BASIC-program sparade med SAVE. Om filen är en textfil läggs alltid ett extra block, med sex inledande nollor, sist i filen.

FEL SOM KAN INTRÄFFA
Det är endast när man läser från kassett som man kan råka ut för fel. Det finns endast tre fel som kan uppstå till följd av problem med hårdvaran (bandspelare, band, glappkontakt osv) och de är:

- enhet ej klar (t ex bandspelaren ej startad eller för lång tystnad på bandet före ett block, max fem sekunder, dator med checksumma<>11273 tål längre tystnad)
- checksummafel (block inläst men innehållet i det är felaktigt beroende på läsfel)
- felaktigt blockformat (block inläst men det är i ett icke ABC80-kompatibelt inspelningsformat, kan också bero på läsfel)

Råkar man ej ut för det första felet kommer alltid ett hel block att läsas in, därefter sker en del kontroller i denna ordning:

1 ETX saknas efter block
2 checksummafel vid läsning
3 de tre första byten i filnamnsblock är ej 255.
4 första byten i datablock (filnr) är ej noll
5 blocknr i inläst datablock stämmer ej med intern räknare (för varje inläst datablock ökas en intern räknare med ett)

Sker sökning efter en speciell fil (filnamnsblock med önskat filnamn) ignoreras kontrollerna 1, 2, 4 och 5. Läsfel vid läsning av block som ej tillhör eftersökt fil ignoreras sålunda.

SNABBARE MED SKIVA
Hur lagras data på en flexskiva? Vet man hur det fungerar på kassettband kan det vara roligt att jämföra kassettlagringen med flexskivelagringen. Läsning och skrivning på skiva går relativt snabbt. Dessutom har man direkt access till önskad fil, man behöver inte spola något kassettband fram och tillbaka. Även felhanteringen är bättre, datorn klarar av att göra nya läsningar om det går snett.

SKIVANS STRUKTUR
Innan man använder en skiva första gången måste den formateras. Skivan delas då upp i ett antal spår, t ex 40 st. Om man tänker sig skivan sedd uppifrån är ett spår lika med en cirkel på skivan. Andra cirklar innanför eller utanför denna bildar ytterligare spår. Varje spår delas in i ett antal sektorer, t ex åtta stycken. En sektor kan man tänka sig som en tårtbit av skivan.

Skivan delas in i tre olika delar. De yttersta spåren är reserverade för DOS, diskoperativsystemet, för att t ex markera vilka sektorer på resten av skivan som är lediga. Därefter följer biblioteket som är skivans innehållsförteckning. Datadelen, som är den största delen, ligger innanför biblioteket.

BIBLIOTEKET
Biblioteket kan innehålla maximalt 128 filbeskrivningar om vardera 16 bytes. Det behövs en filbeskrivning för varje fil på skivan. En filbeskrivning innehåller:

- en byte med det spårnr där filen börjar
- en byte som används till två saker: - sektornr på det spår där filen börjar - flaggor för skrivskydd och raderskydd
- två bytes med fillängd (pga fel i DOS uppdateras ej dessa)
- elva bytes med filnamn (utan punkt)
- en byte med ASCII 255

DATADELEN
Datadelens spår används för att lagra en fils egentliga data. Det är inte säkert att en fils sektorer (en sektor rymmer ett block med 256 tecken) ligger efter varandra på skivan. När en fil skapas väljer DOS ut de sektorer som är lediga. Flera sektorer som ligger bredvid varandra sammanförs till segment. En fil kan bestå av flera segment spridda över skivans datadel. För att hålla reda på var en fils segment befinner sig, inleds en fil alltid med en segmentsektor. En segmentsektor ser ut så här:

- en byte med filnr som är en referens till var i biblioteket som filbeskrivningen ligger
- två bytes med nollor
- en byte med det spårnr där segment börjar
- en byte som innehåller två saker: - sektornr där segment börjar - antal sektorer i detta segment
- en byte med 255

Observera att om filen innehåller flera segment än ett upprepas de två bytes som innehåller spårnr och sektornr som pekar ut ett segment. En byte med 255 tolkas som slut på segment, inga fler segment finns för filen.

Segmentsektorn pekar alltså ut filens olika segment, dvs klumpar av datablock, som ligger utspridda på skivan. Det ideala tillståndet för en fil är att ha endast ett segment. Om flera segment ligger utspridda på skivan tar det längre tid att hitta alla.

Ett datablock, som är en del av ett segment och ryms i en sektor, ser ut så här:

- en byte med filnr som är en referens till var i biblioteket som filbeskrivningen ligger
- två bytes med blocknr (alla block i en fil numreras från ett och uppåt)
- 253 bytes med data

EFFEKTIVITET
Om många sekventiella filer på en flexskiva sparats på nytt i nya versioner, har kanske längden på de nya versionerna varit längre än de gamla. Filerna får då nya segment och skivan får en ökad fragmentering. Segmenten som hör till en viss fil hamnar längre ifrån varandra och det går långsammare att läsa filen. Ett sätt att komma över detta är att kopiera över alla filer en och en till en ny skiva.

FEL SOM KAN INTRÄFFA
Med flexskivehantering kan flera olika fel inträffa. Exempel på några vanliga är:

- filen skrivskyddad (med en liten tejpbit kan man skrivskydda en skiva mekaniskt)
- filen raderskyddad (man kan sätta en flagga per fil som säger att just denna fil inte får förändras)
- skivan full
- skivan ej klar (kanske har man glömt att stoppa in en skiva i driven)

<nnnn>
Mr X

Angående storleken på texten i MSG-utdragen

MSG-utdrag har funnits i ABC-bladet sedan 2,1985. Den gången omfattade utdragen inte fullt tre sidor. De var vidare då ordnade efter innehåll och sovringen hade varit mycket hård. Texten var satt i tre-spalt. Vi fortsatte sedan två nummer till med tre-spalt och materialet blev då mycket dominerande. Då var en av anledningarna att vi hade en eftersläpning i en allmän redovisning av vad MSG står för.

MSG-material i ABC-bladet

Nr	rader	spalter
2,85	700	3
3,85	2200	3
4,85	6300	3
1,86	5120	4
2,86	8920	4
3,86	4760	4
4,86	1790	4
1,87	6300	4
2,87	11000	4
3,87	4500	4

Redaktionen funderade på vad som kunde göras. Redigeringsarbetet tog mycket ideelt arbete när vi hade ambitionen att korrigera och rätta till något som skrivits on-line. Vi tyckte också att MSG-materialet inte skulle ges samma utformning som genomarbetade artiklar men att den känsla av närvaro som materialet ger i oredigerad form skulle tas tillvara. Frågan var bara hur.

Vi beslöt oss för att inte "rätta" så mycket som vi tidigare gjort. En viss redigering av huvudena blev nödvändig för att man skulle kunna hitta i texten och för att det inte skulle bli helt oläsbart. När det gäller kommentar och texthänvisningar beslöt vi att bara ta med sådana som syftade framåt, förutsättande att man läser texten från början, åtminstone en gång. Att det sedan kan ha blivit olika beror på att det kan ha varit olika krafter som (frivilligt!) hjälpt till med detta volymmässigt mycket stora jobb.

Storleken på texten.
Sedan man väl fattat beslutet att texten inte skall ha samma dignitet som artiklar är storleken på texten en konsekvens av utformningen av tidningen i övrigt.

Normalt görs tre spalter om 33 tecken på varje sida. Detta betyder att vi kör ut all text med en spaltbredd på 33 tecken. Dessa vaxas sedan på förtryckta monteringspapper och lämnas till tryckeriet. En artikel som beskriver detta närmare finns i ABC-bladet 3, 1986 sidorna 54 och 55.

Med denna förklaring är det kanske lätt att förstå att också MSG-utdragen körs ut med spaltbredd 33 tecken. Vi får alltså samma rutiner, endast med den skillnaden att vi monterar dem på förtryckta monteringspapper med fyra spalter. I siffror 3*88=264 rader jämfört med 4*119=476 rader. Detta representerar något mer är 80 % mer text. Må vara att vi inte utnyttjar hela spalten men i och med att varje rad tar större plats i tre-spaltalternativet blir ju detta än högre siffror. Med andra ord skall vi ha trespalt på utdragen och inte öka tidningens omfång (ytterligare) måste vi inskränka MSG-utdragen till drygt 55 % av vad vi annars kan få med.

För tydlighet skull måste vi tala om att vi har tagit "lässvårigheten" av den mindre texten på allvar och att detta "bara" är en teknisk konsekvensbeskrivning grundad på våra arbetsrutiner inom redaktionen. När det gäller värderingar av vad som skall stå i tidningen har vår ambition varit att ABC-bladet inte bara skall bli ett MSG-blad, hur viktigt vi än må anse det vara som skrivs i MSG och kommer alla medlemmar till del.

Av tekniska skäl kan någon ändring inte ske i detta nummer.

ABC-bladets redaktion

Ulf Sjöstrand Claes Schibler

KRONSTAT

KRONSTAT är ett avancerat statistiskt programpaket, som ABC-klubben distribuerar.

KRONSTAT finns både till ABC800 och MS-DOS. Programspråket är BASIC-II.

KRONSTAT är helt öppet och kopieringsbart.

Det finns en tryckt manual. Behöver du en manual, beställ den via ABC-klubben, se klubbannonsen. På programdisketten finns en textfil, README. Den innehåller kompletteringar till manualen.

I våras kom version 3.0 av KRONSTAT. Den versionen blir troligen slutversion beträffande ABC-800.

Det var den första versionen till MS-DOS. MS-DOS versionen kommer att utvecklas ytterligare.

MS-DOS

Följande avser enbart MS-DOS versionen.

Till MS-DOS-versionen behöver man DIAB:s BASIC-II. Man behöver den fullständiga versionen av BASIC-II, inklusive ISAM och Grafik.

Run-time version duger inte, man behöver kunna skriva Basic-avsnitt.

Textfilen README innehåller en hel del anvisningar till hur man installerar KRONSTAT. Läs den, t ex med hjälp av en editor.

BUGGAR I 3.0, MS-DOS.

De allra första exemplaren som distribuerades innehöll en alvarlig bugg, som kom in när jag skulle pressa in en distributionsversion på 360K diskett. I programmet FILTER.BAC strök jag ett avsnitt som inte används, men strykningen blev några rader för kort och ett, inaktiv, avsnitt kom att innehålla strukna funktioner. Programmet hoppar då genast ur, när man beställer en tabell. Felet rättades efter några dagar och alla som fått den felaktiga versionen, ska ha fått en rättad version. Rättelsen består i att raderna 15060-15070 stryks.

I rutinen som skapar en sorteringsfil finns en bug som gör att programmet avslutas, går till Basic. Felet inträffar dock när sorteringsfilen är skapad och man kan återstarta KRONSTAT och sorteringsfilen finns där. Felet rättas genom att ändra rad 1830 i BERSORT.BAC, till '1830 CLOSE', dvs 'Z=FNClog(2)' stryks.

Sorteringsrutinen kan trassla på ytterligare ett sätt. Antal samtidigt öppna filer blir c:a 10. En del datorer, t ex Bondwell 8, har ett defaultvärde, lägre än detta. Detta åtgärdas genom kommandot FILES i CONFIG.SYS, se din DOS-manual. FILES=15 räcker för KRONSTATS behov.

Chi-2 i korstabeller blir fel. I TVANALYS.BAC ska 'Z.' i rad 6953 bytas mot 'Chi2.'.

ENANALYS.BAC, rad 2700, bör kompletteras till '2700 FNAnova Local mkv.,ikv.'

PÅGÅENDE UTVECKLING.

Nästa version av KRONSTAT kommer att kunna utnyttja plotter. Anpassningen görs till HPGL. Det innebär HP7475 eller kompatibel plotter, t ex Facit 4550.

För övrigt tillkommer främst nya statistiska rutiner, som

- Logistisk regression, både 'unconditional' och 'conditional'
- Wilcoxon rangsummetest, med exakt signifikans vid 'ties'
- Exakta konfidensintervall vid binomialfördelning
- Variansanalys vid 'Repeated measurements'
- Loglinjära modeller

<4090>
Anders Lindeberg

Mera om WHILE och WEND

Sven Wickberg ville har mera utförligare förklaringar till "IF 0 WEND" enligt ett brev i MSG. Han fick så klart snabbt ett svar i ett personligt brev i MSG. Kanske det finns även andra som vill ta del av förklaringen därför denna artikel.

"WHILE -1" är alltid sann och "IF 0 WEND" är aldrig sann, dvs en loop som ser ut så här:

```
WHILE -1
...
WEND
```

kommer att snurra i evighet medan loopen

```
WHILE -1
....
IF 0 WEND
```

inte är en loop utan gås igenom en gång bara. På D-BASIC V dvs BASIC:en till DS90 kan man skriva:

```
IF A=1
...
IFEND
```

Detta går ej på ABC och BASIC II/PC och ersättes då av

```
WHILE A=1
...
IF 0 WEND
```

Slutet med "IF 0 WEND" är alltså ett trick för att kunna skriva struktuerade IF satser.

I D-BASIC V kan man också skriva flerradiga IF-satser med ELSE satser kombinerade med IF. Man kan t ex skriva:

```
IF A=1
...
ELIF A=2
...
ELIF A=3
...
ELSE
...
IFEND
```

Om A är 2 utförs enbart det som står mellan satsen "ELIF A=2" och "ELIF A=3". Är A skilt från 1, 2 och 3 utförs det som står efter ELSE. ELSE behöver inte nödvändigtvis finnas med. På BASIC II måste detta skriva något mera klumpigt:

```
WHILE A=1
...
IF 0  WEND
WHILE A=2
...
IF 0  WEND
WHILE A=3
...
IF 0  WEND
WHILE A<>1 AND A<>2 AND A<>3
...
IF 0  WEND
```

Ett alternativ till "IF 0 WEND" vore att sätta A till 0 genom "A = 0" och på så sätt skulle man kunna skriva:

```
WHILE A=1
...
A = 0
WEND
```

Man får dock problem med sista satsen. Den skall genomlöpas om A var noll från början men inte om någon del av funktionen har genomlöpts och A har blivit noll enbart för att man skall komma ut ur den första while.

För att lösa detta måste man sätta upp en flagga som talar om för sista WHILE satsen att uthopp har skett. Här kallar jag denna för Break. Kommer man in i den sista funktionen så används variablen Break för att se till så att man kommer ur den.

```
Break = 0
WHILE A=1
...
A = 0
Break = -1
WEND
....
....
WHILE A<>1 AND A<>2 AND A<>3 AND
Break=0
...
WEND
```

<1789>
Bo Kullmar

Om iterationer

"Liksom överlämnad åt sig själv
går upprepningen genom världen..."
(Majken Johansson)

Redan före datorns egentliga genombrott lyckades någon teoretiker bevisa, att ett datorprogram inte behövde mer än tre grundläggande typer av operationer:

följder	(sekvenser)
upprepningar	(iterationer)
val	(selektioner)

Denna artikel handlar om upprepningar i basic1 (ABC80) och basic2.

GOTO

Det är mycket vanligt att man behöver gå tillbaka i ett program och passera en genomgången rutin ytterligare en eller flera gånger.

I basic finns det förkättrade kommandot GOTO, med vilket man kan hoppa vart som helst, när som helst och hur som helst. Att ovana programmerare också gör det har givit basic dåligt rykte - så dåligt att somliga programmerarskolor menar att den som en gång kommit i kontakt med basic är för all framtid förstörd som programmerare! Mindre kategoriska profeter nöjer sig med att säga, att man absolut inte skall använda GOTO (och vissa programspråk saknar denna möjlighet, eller har fått den insatt motvilligt och ganska sent - COMAL, Pascal, Ada).

Sant är att GOTO bör undvikas; eftersom man i ABC80-basic inte har tillräckligt många alternativa möjligheter, utan måste använda GOTO, får man i stället vinnlägga sig om mycket strikta regler i användningen: man skall bara hoppa till början eller slutet på en rutin, eller för att successivt hoppa från en valmöjlighet till nästa.

Just i ABC80-basic har man gjort det psykologiskt litet lättare genom att göra själva ordet GOTO överflödigt i villkorssatset:

```
IF <villkor> THEN radnr ELSE <annat radnr>
```

(Att man t o m kan hoppa över THEN förstör förstås argumentet litet...)

Under alla förhållanden bör man undersöka vad man kan göra för att skriva på annat sätt, inte minst därför att GOTO binder programskrivaren vid ett radnummer, som i bug-letandets hetta lätt kan förändras okontrollerat och ge upphov till nya buggar. Man måste alltså veta exakt var man skall leta efter den rad som hoppsatsen pekar på.

Det enklaste sättet att förklara detta är att titta på vilka alternativa möjligheter till upprepningar som finns i basic2, och sedan hur man emulerar (=efterliknar) dem i annan basic.

FOR - NEXT

Om en rutin skall upprepas ett förutsägbart antal gånger är FOR-NEXT den naturliga metoden, och den finns även på ABC80:

```
FOR I=1 to 100
...
NEXT I
```

Kombineras rutinen med en valmöjlighet kan den användas i stället för de finare rutiner som saknas i ABC80. Mera därom nedan.

FOR-NEXT-slingan är skriven för maximal hastighet, vilket man kan tänka på om den inte innehåller alltför många krångligheter och man räknar med att behöva löpa slingan många gånger och vill att det skall gå undan.

WHILE-WEND

Basic2 innehåller modernare grejor:

```
WHILE <villkor>
...
WEND
```

Så länge <villkor> är SANT skall rutinen genomlöpas från början och till WEND; därifrån skall man gå tillbaka till WHILE och pröva villkoret på nytt.

Villkoret kan vara sammansatt:

```
WHILE Kod<>13 AND Radlängd<Radmax
```

En bra sak med denna rutin är att den inte är bunden till radnummer. När <villkor> blir FALSKT fortsätter exekveringen på raden efter WEND.

WHILE-raden fungerar som dörrvakt: om villkoret INTE är uppfyllt slipper man inte in. Det kan betyda att man måste har en initieringsinstruktion före WHILE eller en viss typ av instruktion, trots att samma instruktion upprepas inuti slingan:

```
I=I-1
WHILE I>5
....
I=I-1
WEND
```

Detta sätta att skriva kallas "read ahead" (läs i förväg) och är ganska vanligt.

Om man har behov av en slinga som alltid genomlöps minst en gång, börjar man se sig om efter något i stil med:

```
REPEAT
...
UNTIL NOT I>5
```

som finns i bl a Pascal. I Basic2 fixar man det med:

```
WHILE -1
---
IF NOT I>5 THEN WEND
```

-1 är beteckningen för SANT, därför kommer WHILE -1 alltid att släppa fram programexekveringen. Villkoret har nu hamnat i sista raden; och observera att det måste formuleras omvänt mot tidigare. Så länge villkoret är SANT kommer slingan att fortsatta från WHILE-raden. När villkoret blir FALSKT, vilket betecknas noll i en dator, då utförs inte resten av IF-raden, alltså inte WEND heller och exekveringen fortsätter med raden efter.

FÖR ABC80

I den gamla trotjänaren ABC80 måste man använda GOTO för att klara den här sortens upprepningar. Med samma slags villkor som i ovanstående exempel kunde det bli:

```
xx  IF NOT I>5 THEN yy
    ...
    GOTO xx

yy ...
```

Hoppet sker alltså antingen till början av slingan eller till raden efter slingan. Man kan undvika det ena hoppet med

```
xx  FOR I=1 to 1000
    IF NOT I>5 THEN yy
    ...
    NEXT I
yy ...
```

Det gäller att ta till så många varv att brytvillkoret säkert uppnås; eller också måste man före rad yy i alla fall sätta in ett återhopp till rad xx. Utan GOTO går det tydligen inte!

Om så önskas kan man sätta villkoret sist i FOR-NEXT-slingan och får på så sätt en rutin som alltid genomlöps minst en gång.

FLERRADIG IF-SATS

Ibland har vi villkorssatser med ganska långrandiga följder. Om man trycker vänsterpil under editering skall radlängden kortas av och sista tecknet i raden suddas på skärmen m m:

```
GET W$
IF ASC(W$)=8 THEN A$=LEFT$(A$,LEN(A$)-1)
:;CHR$(8,32,8);:X=X-1 osv
```

I vår basic måste man skriva alltsammans på samma rad efter IF, såvida man inte upprepar samma IF på nästa rad. Dessutom kan det visa sig nödvändigt att avsluta raden med ett GOTO förbi alla andra rutiner som genomförs ifall villkoret på denna rad inte är uppfyllt.

Eftersom basic inte tillåter hur långa programrader som helst (och de dessutom blir svåra att överblicka och bevärlia att ändra i) längtar man ibland efter flerradiga IF-satser (något som lär finnas i senare basicversioner).

Basic2 ger en möjlighet att emulera flerradiga IF-satser:

```
GET W$
WHILE ASC(W$)=8 ! vänsterpil
   gör alla
   saker man
   behöver göra
   på flera
   rader
IF 0 WEND
```

IF NOLL är ALLTID falskt och WEND utförs inte; slingan genomlöps en enda gång och exekveringen fortsätter med nästa rad.

FÖR ABC80
I basic1 går inte detta; har man stort behov av en flerradig funktion finns dock vissa knep:

```
xx  IF ASC(W$)<>8 THEN yy
    nu kommer
    alla
    konsekvensraderna
    ! slut här

yy ...
```

Konsekvensrutinerna genomlöps en enda gång, men endast om det ursprungliga villkoret (ASC($)=8) är uppfyllt. Lägg märke till att man här får "vända" villkoret.
Är konsekvensraderna tillräckligt många kan det löna sig med en subrutin:

```
IF ASC(W$)=8 GOSUB zz
```

Detta tar mer plats, både i programkoden och i arbetsminnet, och slöar ned programmet en aning. I våra äldsta maskiner med bara 16K RAM-minne kan det någon gång ha betydelse vilket man väljer; annars är det mera en smaksak. Och med GOSUB slipper man GOTO, om nu det är en tröst, men man måste ändå peka på ett radnummer, även om det - med rätt placerade REM-satser - bör vara relativt lätt att återfinna.

ON ERROR GOTO
Felfinnarfunktionen innehåller, dess värre ett obligatoriskt GOTO. Det kanske inte går att skapa den på annat sätt?
Ofta använder man ON ERROR för att åtgärda vad som inte i och för sig är ett programfel: data slut, filen slut, fel data inmatade osv. Man vill bara återföra programmet till inmatningssatsen eller gå vidare med nästa rutin.

```
xx  ON ERROR GOTO xx
..  ;"Välj (1-9)"; : INPUT A
.. IF A<1 OR A>9 THEN xx
```

Om A inte är en siffra hamnar man automatiskt på rad xx igen; raden därpå testar gränsvärdena för A och skickar likaledes tillbaka till xx. (Observera att man måste genomlöpa ON ERROR-raden igen, annars kan felövervakningen sluta fungera!) Exemplet är inte heltäckande, och man kan finna elegantare metoder att övervaka inmatningar, men här visas principen som handlar om att man visserligen blir bunden av ett radrummer, men att rutinen är lätt att överblicka och rätta till.
När man avläser en fil blir det krångligare. Anta att vi vill flytta hela filinnehållet från §1 till §2:

```
yy  ON ERROR GOTO xx ! fortsätt till
    filslut
..  INPUT §1, A$
..  ;§2, A$
..  GOTO yy
xx  (fortsätt)
```

GOTO pekar snällt på antingen början eller slutet av resp rutin. En basic2-kännare föredrar nog:

```
..  ON ERROR GOTO xx
..  WHILE -1
..    INPUT §1, A$
..    ;§2,A$
..  WEND
xx  (fortsätt)
```

Då slipper man det mest irriterande GOTO, det som hoppar bakåt och som är svårast att kontrollera.

I FUNKTIONER
Medan ABC80 måste använda GOSUB för att hoppa till underrutiner har basic2 funktioner att ta till. Även om funktionerna även kan ha andra funktioner (vits!), duger de mycket ofta som subrutiner. En stor fördel är att man slipper bindningen till radnummer.
Funktionerna hjälper också programmeraren genom att förhindra vissa grepp. Man kan lämna en funktion BARA med RETURN (någonting). Inga GOTO här inte! De enda GOTO som tillåts är de som pekar på en rad inom funktionen själv.
ON ERROR GOTO inuti en funktion är helt lokalt begränsad och upphör när man lämnar funktionen med RETURN. Felhanteringen måste således tilldra sig inom funktionens ram (eller en inom funktionen anropad ny funktion!).
I BIT FÖR BIT anvisas följande metod:

```
DEF FNFunktion ...
   ON ERROR GOTO xx
   ...
   RETURN (någonting)
xx Felhantering
   RETURN (någonting annat)
FNEND
```

Man har dubbla utgångar; men under alla förhållanden sker återhopp till instruktionen efter funktionsanropet. Inga radnummer behövs utanför funktionen själv.

LÅSTA SLINGOR
Om en slinga börjar med WHILE 0 (noll) är den "stängd", dvs programmet kommer inte in i den eftersom 0 alltid är FALSKT och hopp sker till raden efter WEND.
Den fingerfärdige Kristoffer Eriksson visade mig på ett synnerligen sofistikerat trick med en stängd WHILE-slinga.

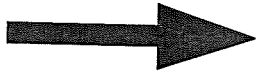
```
ON ERROR GOTO xx
WHILE A>B
läs i fil
hämta A och gör något
WEND  ! här avslutas om A=<B
!
WHILE 0
xx Felhantering, t ex stäng fil
IF 0 WEND
!
rutinen fortsätter
```

Här har man dubbla avslutningsvillkor, antingen att inte längre A<B, eller att det blir programfel, t ex filen slut.
Om villkoret A>B bryts först, sker normal utgång ur den första WHILE-slingan. Eftersom nästa börjar med WHILE 0 hoppar programmet raskt över den och går vidare till "rutinen fortsätter". Men om t ex filen tagit slut, sker ett hopp till rad xx, dvs in i den "stängda" slingan, och felhanteringen utförs!
Detta gör man knappast om i ABC80, men även om vi gamlingar får lov att dras med begränsningarna i basic1, kan vi göra mycket för att rensa bort onödiga och felaktiga GOTO-hopp. Och då finns det hopp (vits 2!).

<1384>
Sven Wickberg

PS

Anledningen till att jag kom att intressera mig så mycket för WHILE-slingorna var att jag skulle lära mig programmera i Pascal, där i princip GOTO är ett fult ord. Man KAN hoppa till en s k LABEL (etikett), men det anses inte fint och alldeles onödigt.
Min övningsuppgift var att skriva en del av en editor. Jag behövde ett program som tog emot text tecken för tecken och bröt raden vid föregående mellanslag ifall raden blev längre än ett förut bestämt värde, Radmax.
Raden kunde också bytas med RETURN, och så kunde man avsluta editeringen genom att skriva > först på en rad. Det var massor av villkor att hålla reda på, och jag gick rejält bet och införde två LABELs. Programmet fungerade, men min programmerarära var inte värd mycket...
Min son, som minsann studerat programmeringskonsten på KTH och är yrkesverksam i branschen, suckade, muttrade något om det dåliga inflytandet från basic, och påpekade att jag kunde "sätta några flaggor" i stället...
Så småningom kom jag väl fram till den rätta knycken och kunde undvara mina två LABELs. Programmet översattes sedermera till basic2 och fungerar jättefint.
För den intresserade visas de viktigaste dragen här:
Flaggorna heter Slut och Bytrad och Klar.



Från början är alla flaggor nollställda.
"Slut" inträffar när man skriver > först på ny rad. Då sätts Slut=-1 ("flaggan sätts " eller "hissas"), vilket sker på den rad som utmärkts med >.
Som synes överhoppas då alla följande WHILE-slingor och funktionen avslutas, och man återvänder till huvudprogrammet som har för sig en del filslut, printerutskrifter och annat - editeringen är för denna gång avslutad.
Men om man skriver andra tecken lagras de i Rad$ tecken för tecken ända tills Rad$ överskrider en viss längd Radmax. Om så är fallet (vid >> i progrmmet) sker hopp till funktionen Klipp som undersöker alla möjligheter att göra ett snyggt radklipp. Rad$ delas upp i två rader (borttaget här) och man hisssar flaggan Bytrad för att tala om att klipp har skett. I så fall skall nämligen hopp ske till xx för vissa åtgärder (förenklade här), varefter Bytrad kan tas ner och en ny rad börja byggas upp.
FNKlar innehåller dels en WHILE-WEND-sats som avslutas när flaggan Klar har hissats, och dels en rad flerradiga IF-satser. En mängd detaljer har givetvis utelämnats, och jag är övertygad om att det finns gott om läsare som är beredda att omgående visa hur man kan lösa denna uppgift med alldeles extra finurligheter. Vi avvaktar med intresse en ström av nya, intressanta artiklar i Bladet!

<1384>
Sven Wickberg


```
================================================
DEF FNSkriv
  WHILE NOT Slut
    Kod=0 : Slut=0  ! Flaggan nollställs
    ...
    While Kod<>13 AND NOT Bytrad AND NOT Slut
      GET W$ : Kod=Ascii(W$) ! hämta tangenttryck
>     IF Rad$="" AND W$=">" THEN  Slut=-1
        ! > först på tom rad hissar flaggan Slut
      ...
      WHILE Kod>31 AND NOT Slut
        ; W$; : Rad$=Rad$+W$
        ! Skriv tecknet och lägg den till Rad$
>>      IF LEN(Rad$))Radmax THEN Z=FNKlipp ! korta raden
      IF 0 WEND
    WEND
    !
xx  WHILE NOT Slut
      (skriv gamla raden på fil)
      (börja ny rad med sista ordet i gamla)
      (m m)
      !
      Bytrad=0  ! Nollställ flaggan igen
      !
    IF 0 WEND
  WEND
  RETURN 0
FNEND
!---------------------------------------
DEFFNKlipp ! klipper lång rad vid senaste blank
  ...
  Klar=0   ! Flagga
  WHILE NOT Klar
    (leta blank - ganska invecklat!)
    (när blank funnen) Klar=-1
    ...
  WEND
  !
  WHILE (ingen blank)
    (trunkera vid Radmax)
    ...
  IF 0 WEND
  !
  ...
  WHILE (blank funnen inne i raden)
    (normalt radklipp)
  IF 0 WEND
  !
  Bytrad)-1 ! Flaggan hissas
  !
  RETURN 0
FNEND
================================================
```

Datahistoria

FÖREGÅNGARNA

Abakus uppfanns 5000 år f.Kr. Den används fortfarande i banker och på kontor i Kina, Sovjet och Japan. Det är en träram med rännor eller järntrådar längs vilka man flyttar små kulor. Den kallas även kulram, och den fick bli vår föregångare till datorn.

Sedan kom Pascals räknemaskin, den första maskinen för numerisk räkning som förtjänar namnet. Blaise Pascal döpte den till Pascalinen, den fungerar på samma sätt som vägmätaren i bilen. Räknemaskinen av tysken Wilhelm Schickard var den första aritmetiska maskinen som kunde klara de fyra räknesätten, byggdes 1624. Han kallade den för räkneur.

Gottfried Wilhelm Leibniz uppfann år 1671 en mekanisk kalkylator liknande pascalinen men mer utvecklad. Pascals maskin kunde bara dra ifrån och lägga till. Leibniz maskin kunde även multiplicera, dividera och dra ut kvadratrötter. Bägge maskinerna byggde på den stegvisa beräkningen. De upprepade en och samma operation t ex en serie additioner. I våra dagar arbetar ett stort antal datorer på samma sätt. Tänk att man aldrig blir av med gamla spöken :-).

1835 presenterade Charles Babbage sin 'analytiska motor'. Denna apparat var världens första numeriska dator. Den analytiska maskinen kombinerade aritmetiska operationer med beslut grundade på dess egna beräkningar. Den använde ett system av 50 kugghjul och inmatningen av data gjorde med hjälp av hålkort. Vid denna tidpunkt brevväxlade han med Lady Ada Byron, Grevinnan Lovelace, dotter till den berömde engelska poeten. Hon stod honom mycket mycket nära. Hon fick Charles Babbage att utföra mekaniska ritningar som visade hur man kunde utföra ett stort antal komplexa analyser med hjälp av hans maskin. Tyvärr var 1800-talets teknik inte tillräcklig för att förverkliga merparten av deras lysande upptäckter. När Charles Babbage konstruerade maskinen gjorde grevinnan Lovelace programmen till maskinen. Hon var historiens första programmerare. Det är ganska konstigt att programmering inte blev ett typiskt kvinnogöra, eller hur ? Förmodligen skulle Charles Babbage aldrig kunnat konstruera maskinen utan grevinnan Lovelace hjälp.

Charles Babbage fick inte sin maskin att fungera tillfredsställande. Men två svenskar, George och Edvard Scheutz (far och son), byggde år 1853 en fungerande differensmaskin som tillverkades i två exemplar. Den ena användes länge som tabellberäkningsmaskin vid ett amerikanskt observatorium.

Den första räknemaskinen som serietillverkades och kom till allmänt bruk uppfanns 1878 av svensken Willgodt Odhner (1845-1905). Odhners mål var att skapa en liten räknemaskin och han lyckades konstruera en, som byggde på det av honom patenterade pinnhjulssystemet. För tillverkning av maskinen grundade han en fabrik i Sankt Petersburg, vilken var verksam till 1917 och fick fortsättning i det svenska företaget AB Orginal - Odhner. På Odhners räknemaskiner var det möjligt att enkelt utföra de fyra räknesätten. Ni har väl alla sett den hos morfar-/farfar?

Utan att vara medveten om sin betydelse av sin uppfinning fann en fransk företagare, Joseph-Marie Jacquard (1752-1834) en praktisk tillämpning av den numeriska styrning i form av mekaniska vävstolars funktion. Jacquards vävstolar blev år 1804 den första tillämpningen av hålkort där varje hål representerade en siffra som styrde vävens mönster.

George Boole 1815-1864 engelsk logiker och matematiker, utvecklade det binära systemets logiska operatorer, såsom OCH, ELLER etc. I sin Treatise on differential equations publicerad år 1859, framlade han den symboliska metod på vilken den booelska algebran och den binära kommunikationen idag vilar.

Uppfinningen av kodning kan tillskrivas amerikanen Herman Hollerith (1860-1929). Kodning av hålkort kallas ibland Hollerithkod till minne av denne föregångsman. Vidare kan man notera fransmannen Emile Baudots bidrag, han uppfann den telegrafiska koden, patentsökt år 1874. Kodningen går annars tillbaka till de första anordningarna för hålkort såsom Jaquards vävstolar eller Charles Babbages analytiska maskin.

Valdemar Paulsen (1869-1942) en dansk ingenjör, presenterade den första apparaten för magnetisk registrering vid världsutställningen år 1900. Poulsens uppfinning lagrade data på en bit rund stålträd som snabbt rullades mellan två spolar. Tråden var föregångare till magnetbandet som vi känner till idag, med järnoxid på tunt plastband.

Det dröjde till 1906 innan en amerikansk uppfinnare, Lee De Forest (1873-1961) skapade en anordning som gjorde det möjligt att använda Booles binära system i praktiken. Utgående från elektronröret kunde man tillverka det binära systemets logikkretsar. Det gjordes i början på de elektroniska datamaskinernas tidsålder.

Automatiken, en vetenskap som behandlar studiet och förverkligandet av mekanismer och system som kan fungera utan mänskligt ingripande, beskrevs först av den spanske ingenjören Leonardo Torres Quevedo 1913. Han kallade denna vetenskap för "autom tico", efter användningen av automater.

BESK (binär elektronisk sekvenskalkylator) konstruerades under ledning av teknologie doktor Conny Palm på uppdrag av matematikmaskinsnämnden och togs i bruk år 1953. Den hade från början elektrostatiskt minne, som sedan byttes ut mot ett ferritminne. Den förfogade även över ett trumminne och innehöll ca 3000 elektronrör. Professor Erik Stemme hade ansvaret för den andra, BARK, som var föregångare till de svenska maskinerna Facit EDB, SARA, SMIL, DASK och TRASK.

Första generationen

Den första binära räknaren gjordes av George Stibitz 1939 vid Bell-laboratoriena, under namnet Mode 1 Relay Computer eller Complex Number Calculator. Det rörde sig om en logisk maskin där de data som matades ut representerade summorna av dem som matades in. I sin maskin använde Stibitz telefonireläer som fungerade enligt principen allt eller inget (dvs användning av ett eller noll) med sikte på att utarbeta en universiell räknare. Till detta använde han några kasserade reläer, två glaskolvar och bitar av en tobaksburk, alltsammans ihopplockat under ett veckoslut.

När andra världskriget började sökte de brittiska dechifferingsexperterna ett sätt avkoda de tyska meddelandena. De anförtrodde åt ungraren Alan Turring bosatt i England sedan 1936, ledningen för en grupp med uppgift att lösa detta problem. Före kriget hade Turring preciserat begreppet beräkningsbarhet och anpassat algoritmbegreppet till beräkningen av vissa funktioner. Han hade definierat Turringsmaskin, som i teorin hade förmågan att beräkna varje beräkningsbar funktion. År 1943 i Bletchley Park, kördes den första Colossus igång. Denna dator innehöll över 2000 elektronrör och kunde behandla 5000 tecken/sekund. Det var här ifråga om en specialiserad maskin som dechifferiserade tyskarnas meddelanden, och ända till krigets slut var den engelska regeringen underrättad om de tyska förehavandena.

Men amerikanarna eftersträvade att skapa en modern version av Babbage maskin. År 1944 presenterade Howard H. Aiken, Harvards I även kallad IBM Automatic Sequence Controlled Calculator. Denna räknare förbättrade på ett förnuftigt sätt Charles Babbage dröm genom att lägga till två innovationer:

1. En klocka i syfte att synkronisera de olika operationssekvenserna.

2. Införandet av register, en ide som skulle komma tas upp av alla datorkonstruktörer.

År 1946 byggdes den första elektroniska universaldatorn, ENIAC. Den vägde 30 ton och upptog en yta på 160 kvadratmeter och innehöll 18000 elektronrör. Tack vare elektroniken förde den in snabbheten i datorernas värld. Det var under arbetet med ENIAC som termen bit skapades. ENIAC skulle sedan användas till beräkningar.

Det var i gruppen Moore School vid Princetons Institute of Advanced Study i New Jersey, som iden till den programmerbara räknemaskinen föddes. von Neumanns revolutionerande ide var att minnet skulle innehålla både program och data. Arbetet utmynnade den 24 Jan 1948 med att man kunde presentera historiens första dator som kunde ta emot ett inläst program, IBMs SSEC. Den använde 13500 vakuumrör, 21000 reläer och utförde additionen av 3500 14-siffriga tal på en sekund.

GRUNDBEGREPP

BINAC (Binary Automatic Computer) var en dator med stor användbarhet, som använde fördröjningsledningar till minne (512 ord). Den fullbordades år 1946 av amerikanen J. P. Eckert och J. Manchly. I själva verket bestod BINAC av två datorer som samtidigt utförde samma beräkningar, varefter resultaten jämfördes. BINAC var den första datorn som arbetade i reell tid. Dess driftsäkerhet var fantastisk för den tiden: en av de två datorerna fungerade 1949 i 44 timmar utan fel.

Gene Amdahl utarbetade det första operativsystemet år 1954 det användes till IBM 704.

En assemblerare, eller snarare symbolspråk för assemblering, användes först gången ca 1950 i Cambridge i England på EPSAC-datorn av en grupp ledd av H. V. Wilkes. EPSAC var den första helt elektroniska dator som var utrustad med bildskärmar för att visa innehållet i dess fördröjningsminne. Den första assembleraren som levererades av en datortillverkare var SAP, som konstruerades av United Aircraft Corporation (USAP) och installerades i IBM 704.

Det var H. V. Wilkes som 1951 införde begreppet mikroprogrammerare för att förenkla funktionen hos datorernas räkneenhet. Multiprogrammering förekom första gången 1961 i Stretchdatorn. Multiprogrammering är en användningsprocess där man har flera olika program lagrade i datorn, vilka används "inflätade" i varandra.

1961 användes för första gången tidsdelning hos MIT (Massachusetts Institute of Technology) under ledning av F. Corbato, som utarbetade det kompatibla (dvs går att använda i flera olika maskiner utan stora ändringar) tidsdelningssystemet (Compatible Time Sharing System, CTS) för användning i IBM 709 och 7090. Det första kommersiellt erbjudna systemet för tidsdelning var PDP1 år 1962.

Man anför ofta det amerikanska företaget Computer Science Corporation, som det första som sålde tillämpningsprogram (1959).

En databas såg sitt första ljus år 1952 skapad av det amerikanska företaget RCA på en BIZMAC, som byggdes för att ta hand om RCAs informationslagring.

Oktetten användes första gången som grundläggande informationsenhet i Stretch, en transistoriserad dator med stor kapacitet, som byggdes av IBM 1961. Oktett kallas ofta på engelska (och svenska) för byte, något oegentligt, eftersom byte strängt taget innebär en bitgrup av enhetlig längd vilken inte nödvändigtvis behöver vara just åtta. Oktetten används idag allmänt för att representera ett tecken. Datorns minneskapacitet anges vanligen i KiloOktetter (KiloByte).

År 1962 Framlades de första generella simuleringsspråken: SIMSCRIPT av Rand Corporation och GPSS av IBM.

Andra generationen

Transistorn uppfanns år 1947 av fysikerna William B. Shokley, John Bardeen och Walter H. Brattain vid Bell-laboratorierna. År 1956 fick de Nobelpriset för sin uppfinning. Från en början tillverkades de av germanium men runt 1960 började de även tillverkas av kisel.

Den första helt transistoriserade universaldatorn var TRADIC (Transitorized Airborne Digital Computer), utvecklad av J. H. Felker och hans forskarlag för flygteknikernas behov.

Metusalemdatorn, IBMs AN/FSQ-7 som togs i bruk den 1 Juli 1958, är trots sin höga ålder är den fortfarande i bruk hos NORAD i North Bay, Ontario. Den utför beräkningar i den nordamerikanska kontinentens nät för anfallsvarningssystem. Den har 55000 lampor, 135000 transistorer, 7000 bladminnen och tolv trumminnen. Alltsammans upptar ett trevåningshus.

Den första integrerade kretsen tillverkades av amerikanen Jack S. Kilby vid Texas Instruments. Den första kommersiella användningen ägde rum år 1964. Det var en integrerad krets som Texas Instruments använde i en hörapparat.

IN- & UTMATNING

Det första in- och utmatningsmediet var troligen det som i USA år 1954 utvecklades av Bob Evans på en IBM 704. Och fungerade genom att data skickades genom elektriska pulser i kablar. De första av den här typen började marknadsföras år 1958, det var IBM 709.

Första gången man använde en terminal var 1940 i Bell-laboratorierna. Datorn befann sig i New York och terminalen i Dartmouth College, New Hampshire. Man hade dragit en ledning mellan datorn och terminalen. Där gjorde man även det första försöket med en fjärransluten terminal, som även utgjorde den första riktiga erfarenheten av datakommunikation. Det var till Whirland den första interaktiva bildskärmsterminalen anslöts, dvs en bildskärmsterminal som tillåter dialog mellan människa och maskin medan data behandlas. Datorn användes till att simulera flygningar.

Den svenske professorn Erik Stemme (f. 1921) utvecklade 1951 en remstans och remsläsare som byggde på elektrodynamiska lösningar. Remstansen var vid den tidpunkten världens snabbaste och möjliggjorde bättre utnyttjande av datamaskinerna. Stemme hade även konstruerat snabba skrivare för datamaskiner bl a en piezoelektrisk bäckskrivare 1950.

Den första snabbskrivaren värd namnet var den som tillverkades av Remington Rand år 1953 för UNIVAC. Den hade en skrivhastighet av 600 rader med 120 tecken/rad på en minut. År 1957 kom IBM med en skrivare som klarade 1000 rader/minut. Det första in- och utmatningsmediet var troligen det som i USA år 1954 utvecklades av Bob Evans på en IBM 704. Ursprungligen var medierna enkla elektroniska pulser. De första maskinerna utrustade med detta började marknadsföras år 1958 på en IBM 709.

Termen datalista användes för första gången av W. S. Burroughs år 1886 för att beteckna de resultat som matades ut från Burroughs Adding and Listing Maschine.

Den första ljuspennan presenterades år 1963, den ingick i ett grafiskt system som kallades Sketchpad och tillverkades av MIT av I. E. Sutherland.

Den första kurvritaren fick Charles Babbage iden till 1883 för att uttrycka de resultat som hade beräknats av hans analytiska maskin.

Två interaktiva system utvecklades samtidigt, DAC-1 från General Motors och Sketchpad från Lincolnlaboratoriet från MIT. De gör det möjligt att på bildskärm i form av bilder presentera numeriska data som matas ut från dator.

År 1964 presenterade M. R. Davis och T. D. Eillis för första gången Rands tablett, tillverkad av Rand Corporation, USA. Davis och Eillis tablett såg ut som ett litet ritbord. 1966 kom Lincoln-laboratoriet med sin "wand tablet" (trollstavstablett), som tack vare mikrofoner kunde känna av i tre dimensioner (läget), hos en penna som avgav ljudvågor.

Tredje & Fjärde Generationen

IBM 360, av vilken sex olika modeller lanserades år 1964, var resultatet ett projekt som startade år 1961 av en grupp vid IBM ledd av G. Amdahl och M. Blaann som ville tillverka en familj datorer som var kompatibla sinsemellan helt och hållet. Beteckningen 360 på denna serie påminner om dess syfte, definierat år 1961 av T. J. Watsom jr såsom en familj med allroundinrikting ("runt-om 360 grader").

CDC 6000 hör till en datorfamilj med hög kapacitet, sedan 1964 tillverkad av Control Data Corporation (CDC) grundat i Minnesota år 1957. Denna serie datorer använder dataord om 600 bitar. Modellen 6600, den kraftfullaste av dem, innefattade en centralprocessor kopplad till primärminne och ett flertal specialiserade processorer, var och en utrustad med sin egen minnesenhet. Denna uppbyggnad gjorde parallellbe-

handling möjlig och sålunda mycket hög räknehastighet.

År 1959 började en grupp vid det amerikanska företaget Digital Equipment Corporation (DEC), ledd av Benjamin Curley, arbete med datorn PDP1. Det blev den första minidatorn, och började säljas år 1960.

År 1971 presenterade Marcian E. Hoff chef för Intel Corporation den första mikroprocessorn, döpt till Intel 4004. En mikroprocessor, ofta kallad chip, är egentligen en dator, uppbyggd på en yta av några kvadratmillimeter. Kretsen i mikroprocessorn Intel 4004 har 2 300 transistorer. Idag kan en krets av samma storlek t ex Iapax 432 innehålla 225 000!

Den första elektroniska miniräknaren tillverkades av J. S. Kilby, J. D. Merryman och J. H. Van Tassel vid Texas Instruments. Patentansökan lämnades in år 1972 och beviljades år 1978. År 1973 fick Hewlett Packard iden till den programerbara miniräknaren (för speciella användningsområden t ex finans, ekonomi). År 1976 kom samma tillverkare ut med de första programmerbara miniräknarna, egentligen små datorer i fickformat.

År 1980 beslöt IBM, jätten på databehandlingsområdet, att ge sig in på mikrodatorområdet. Företaget installerade ett ungt arbetslag i Florida och gav dem fria händer. Tvärtemot vad som är brukligt inom företaget är PC-n inte byggd kring ett chip utan kring de 16-bitars Intel 8088 som har större kapacitet än konkurrenternas. Andra företag uppmuntrades att skriva program och succen var given. 185 000 sålda exemplar 1982, 850 000 år 1983. Av bara farten lanserade IBM i januari 1983 sin XT, en professionell mikrodator, och i november PC Junior, inriktad på hemmabruk.

Den första kommersiella superdatorn, Cray 1, tillverkades under ledning av Seymour Cray år 1976. Den innehöll 200 000 integrerade kretsar, hölls nedkyld med freon och kunde utföra 150 miljoner operationer per sekund. Den följdes av Cray X/MP med en kapacitet av 400 miljoner operationer per sekund. Ett exemplar av denna modell inköptes 1978 för 21,6 miljoner dollar för att skapa filmer med animerade inslag, bl a "Tron". Men Cray datorn har överträffats av Cyber 205, från Control Data, som utför 700 miljoner operationer per sekund.

MINNEN

Fransmannen Joseph-Marie Jaquard uppfann hålkortet, som han år 1805 använde för sin mekaniska vävstol. Engelsmannen Charles Babbage använde det likaså som datamedium för in- och utmatning till sin analytiska maskin. Men i praktiken var det Hermann Hollerith som lanserade det i samband med den amerikanska folkräkningen år 1890.

Trumminena var förutsedda sedan år 1946 som ett medel för att utöka minneskapaciteten och blev det första magnetiska dataminnena som tillverkades av flera företag (omkring 1950). År 1949 använde Jay Forrester ferritkärnor som beståndsdelar i Whirlwinds primärminne. Ferritkärnor är små ringar av ferrit, ett material som äger permanenta magnetiska egenskaper som gör det möjligt för dem att "minnas" en elektrisk signal. Ferritkärnor började försvinna år 1964 för att lämnna plats åt halvledarminnen.

De första magnetbanden dök upp ganska tidigt; de var tillverkade av stål och mycket ömtåliga, tills IBM år 1948 uppfann en maskin som hanterade dem i vakuum. De första plastbanden användes år 1951 av Luis Fem för Raytheons Faydac. Den första datorn som var utrustad med magnetbandsenhet var UNIVAC I (USA). Detta band fungerade som buffertminne mellan datorn och dess användare. UNIVAC byggdes i ett trettiotal exemplar, den installerades den 14 juni 1951 för att användas i den amerikanska folkräkningen.

De första försöken att använda magnetiska skivor som massminne i en dator, i detta fall EDVAC, ägde rum år 1949. År 1960 kom den första löstagbara skivan i form av en skivpacke som alltså lätt kunde flyttas. År 1962 började IBM saluföra den första skivpacken (1311), vars kapacitet var tre miljoner tecken. Omkring 1970 kom de första flexskivorna, som idag används för de flesta mikrodatorer. Återkom skulle man kunna säga, för efter att ha uppfunnits på 50-talet vid Laboratory for Electronics föll de i glömska.

Atlas, konstruerad i England av Ferranti och presenterad år 1961, erbjöd tack vare en sidväxlingsfunktion möjligheten att använda ett adressbart minne som var större än det reella minnet: det var det första virtuella minnet.

Det första bubbel-minnet byggdes år 1967 av en grupp vid Bell-laboratorierna. Bubbelminnet är inte flyktigt dvs det behåller informationen även om strömmen slås av. Det började marknadsföras i slutet av 70-talet. Minnet med stegvis laddningsförflyttning (Charge-Coupled Device, CCD), en variant av halvledare enligt bubbelminnesprincipen uppfanns år 1969 av W. S. Boyle och G. E. Smith USA. De första kapslarna marknadsfördes år 1975.

ANVÄNDNINGSOMRÅDEN

Kompositörerna var de första som tog hjälp av datorn för konstnärligt skapande. Allra först var det år 1956 i amerikanarna Hillers och Isaacsons verk " Suite Iliac". Vid 1970-talets slut hade datorn gjort sitt inträde på musikens alla områden (från variter till undervisning). Att göra grafisk konst med hjälp av dator går tillbaka till C. Burnett i Californien, som år 1937 på detta sätt framställde målade bilder. De var baserade på Lissajou-kurvor. Men det var Ben F. Laposky som år 1950 på allvar grundlade den datoriserade grafiska konsten. De "konstgjorda" bilderna, är en ren produkt av datateknologin, börjar omge oss på alla håll. Vissa sätts samman av element som redan finns lagrade i datorminnet, men andra uppstår ur matematiska funktioner eller slumpartade processer. I åtskilliga länder, i synnerhet i USA och Frankrike, är man intresserade av att skapa "bilder av framtiden". De arbeten som utförs av Robert Abel & Associates, Digital Equipment Corporation (USA) och Sogitec (Frankrike), har rönt stor uppmärksamhet, liksom verken gjorda av David Elm, Herv Huitric, Monique Nahas och Mike Newmans.

De första ideerna rörande programmerad undervisning beskrevs av den amerikanske psykologen B. F. Skinner år 1954. Det tillvägagångssätt som Skinner på sin tid föreslog. Det gav inga möjligheter att gradera elevernas svar i nivåer. En annan amerikansk psykolog, Norman Crowder uppfann en metod med flerval. Omkring år 1955 började man försöka använda datorn på detta område. År 1958 utfördes ett första experiment vid IBMs Watson Research Center.

Det var år 1951 som de första försöken till animation (simulering) med hjälp av datorer gjordes vid MIT. Men det dröjde ända till i början på 1960-talet innan man förstod följderna av denna tillämpning. Inom medicin, arkitektur (med tredimensionella modeller), rymdforskning och kemin har man sedermera fått stor användning för tekniken.

Numerisk styrning av maskinell utrustning tog sin början år 1956 då språket APT skapades på uppdrag av US Air Force. I viss mening skulle man kunna söka den numeriska styrningens ursprung i Jaquards vävstolar, eller ännu längre tillbaka, hos tsar Peter den store, vars mekaniker A. C. Narton år 1712 uppfann den första automatiska svarven för kopiering av föremål.

Svensken Gerhard Westerberg (f. 1931) har uppfunnit en handburen terminal för datainsamling vid t ex inventering och kompletteringsbeställning i detaljhandel och lager. Iden fick Westerberg redan i början på 1960-talet men var tvungen att vänta till år 1975 för att teknikmognad och marknad skulle hinna ifatt innan terminalen började serietillverkas.

Den datorstödda konstruktionen började uppstå under 1960-talet inom det amerikanska militära flygprogrammen.

Som avslutning kan nämnas att när ABC80 kom var det en mycket bra dator, den var konkurrenskraftig till både pris och prestanda. Många upplevde det som bra att BASICen låg fast i ROM-minnet. Det var även en kraftfull BASIC-dialekt för sin tid. Datorn i helhet var lättanvänd och likaså mycket populär. Om LUXOR oxo varit snabb att lägga sin tillverkning i Taiwan när andra gjorde det hade de säkert inte behövt lägga ner tillverkningen. Nu kan vi bara hoppas att DIAB inte slutar med sin BASIC-PC, så vi får ha något minne av våra kära ABC-maskiner. Dessutom så blir övergången till PC lättare om man använder sig av deras BASIC-PC.

<6530>
Kerstin Jansson

ABC-Graphics Display Controller

Vad?

I några tidigare artiklar har jag beskrivit en utbyggnad av ABC80 till bättre grafik. På grund av det stora uppseende dessa artiklar väckte (inte en enda människa har sagt ett enda ord) kommer här en fortsättning.

Varför?

Jag har länge funderat på det här med 80 tecken per rad. Den vanliga tillsats som finns passade mej inte riktigt, jag ville ha bättre grafik samtidigt. Först försökte jag i samma anda som TKN80 alltså med ett nytt större bildminne i tangentbordet men det blev alldeles för mycket extra elektronik och en massa ingrepp i ABC80. Videoprocessorerna blir bättre och bättre. Det verkade bättre passa mina behov. Med videoprocessor menar jag en integrerad krets som innehåller funktioner för att rita figurer på bildskärmar. Man kan skilja på tre generationer videoprocessorer:

1:a generationen som kan sköta ett dynamiskt bildminne och producera synkroniseringspulser till bildskärmen. Hit kan man räkna t ex processorerna MC6845 som finns i ABC800 och TMS9929 som jag använde i min tidigare grafiktillsats till ABC80.

2:a generationen som dessutom kan rita linjer och cirklar mm på egen hand (vilket gör att det går oändligt mycket fortare). Hit kan man räkna t ex processorerna Intel 82720 den jag använt i ABC-GDC (GDC=Graphic Display Controller) som denna artikel handlar om. Priserna på dessa processorer har sjunkit kraftigt på senare tid vilket gör dem intressanta för amatörer. Intel 82720 kostar nu cia 20 US dollar.

3:e generationen som dessutom kan ge flera färger och hantera fönster och naturligtvis ritar ännu mycket fortare. Hit kan man räkna tex.processorerna TMS34010 och HD63484 Dessa har just börjat tillverkas och är nu mycket dyra, hundratals US dollar per styck.

Blockschemat

ABC-GDC består av ett kort som sätts i en expansionslåda av något slag t ex ABC 890. Sladden mellan tangentbord och bildskärm ska passera kortet, vilket sker genom att sladden från bildskärmen sätts i en kontakt på ABC-GDC och och en ny sladd går från ABC-GDC till tangentbordet. På kortet blandas bilden från ABC-GDC med den vanliga ABC80-bilden. Man kan naturligtvis visa bara en av bilderna också, t ex kan ABC80-bilden tas bort men PRINT CHR$(12).

Teknik

Bildminnet består av 4 st dynamiska minnen 41464 som vardera innehåller 64k*4 bit, totalt 128 kbyte. Timing består av cia 10 vanliga TTL-kretsar.

ABC-GDC har i denna första version upplösningen 608*276 alltså 608 punkter horisontellt och 276 linjer vertikalt. Det beror på att med ABC80 klocka (12 MHz) täcks hela skärmen med denna upplösning. Annars går videoprocessorn att programmera till vilken upplösning som helst upp till 1024*1024 Den ritar linjer och cirklar med 400 000 punkter per sek (i bästa fall). Det finns instruktion för att rita tecken i 8*8 punkter matris. Man skapar alltså sina egna bokstäver och andra krumelurer, ingen begränsning i teckenuppsättningen.

Framtiden

Några möjliga utvecklingar:

En liten ändring i elektroniken och man får färgsignal RGBI. Men då ska man helst ha en färgmonitor förstås.

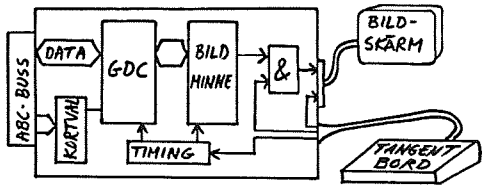
Programmera processorn att ge bild med radsprång, då blir det dubbelt så många linjer och bättre upplösning i vertikal led.

Minska bildhöjden i ABC80-monitorn och sedan programmera processorn att ge fler linjer per bild så att skärmen åter blir fylld. Resultat större upplösning i vertikal led.

Höja klockfrekvensen genom att använda en separat oscillator vilket ger snabbare ritning och högre upplösning i horisontell led.

<6555>
Lars Karis

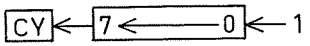
BLOCKSCHEMA ABC-GDC VER 1.



En "bortglömd" instruktion till Z80

Maskinkoden för Z80:s rotate-, shift-, BIT-, RES- och SET-instruktioner består av två bytar, där den första är 203 (decimalt) och den andra anger typen av instruktion och operand. De tre minst signifikativa bitarna i byte 2 representerar instruktionens operand (alltså (HL) eller ett av 8-bitsregistren), medan resterande bitar anger instruktionstyp (t ex kodar byte 2-värden mellan 0 och 7 för RL med olika operander).

För byte 2-värden mellan 48 och 55 finns dock ingen assemblerinstruktion definierad; detta trots att processorn i detta fall faktiskt utför följande variant av SLA på operanden:



Denna "överhoppade" instruktion tycks i alla avseenden fungera analogt med de andra instruktionerna i samma grupp - adressering av argument med indexregistren går alltså också bra. (Detta går också till precis som vanligt; om t ex (IX+3) är operand blir alltså hela maskinkoden 221, 203, 3, 54.)

Flaggorna sätts på följande sätt:

S	Sätts om negativt resultat
Z	Nollställs
H	Nollställs
P/V	Sätts om resultatet har jämn paritet
N	Nollställs
C	Sätts till bit 7 i operanden

Instruktionen är alltså det snabbaste och minnessnålaste sättet att multiplicera ett 8-bitstal med två och därefter lägga till ett. För att få samma effekt med standardinstruktioner krävs följande sekvens:

```
SLA  m
INC  m
```

Detta tar dock både en byte extra och längre tid; dessutom erhålls ingen information om resultetets paritet - P/V signalerar istället overflow.

<1336>
Fredrik Ullén och Peter Lönnerberg

Vi skrev till Karl Lindström och bad om utlåtande och han svarade:
"Detta funkar vad jag vet på alla Z80A och B. Eventuellt på Z80."

Se vidare den röda inlagan i ABC-bladet 4, 1984.

<1182>
Claes Schibler

```
Op.kod    Op.kod          KÄLLKOD
HEX       DECIMAL         MNEMONIC

CB30      203 48          SLL  B
CB31      203 49          SLL  C
CB32      203 50          SLL  D
CB33      203 51          SLL  E
CB34      203 52          SLL  H
CB35      203 53          SLL  L
CB36      203 54          SLL  (HL)
CB37      203 55          SLL  A

DDCB0536  221 203  5  54  SLL  (IX+IND)

FDCB0536  253 203  5  54  SLL  (IY+IND)
```

Byte & bit

På många håll förekommer olämpliga förkortningar av enkla saker. I samband med datorer berörs man bl.a. av de egendomliga bitar och av Kb för kilobyte.

Med den förstnämnda avses bit som är en förkortning av binary digit och är en enhet, som inte kan förses med engelsk eller svensk pluralis (bits, bitar) liksom man på god svenska inte använder meters eller kilogramar. Inte heller bör man förkorta förkortningen en gång till och skriva bara b.

Meter- och kilogram-prototyperna i Paris var nästan 100 år gamla när metersystemet infördes i Sverige år 1878. Sedan dess förkortas allmänt (även på engelska, tyska, franska etc.) exempelvis kilometer och kilogram med gemena km och kg.

Enligt aktuell standard är det bara de tre "stora" prefixen tera, giga och mega, som i förkortad form betecknas med versaler, alltså T, G och M. Alla övriga prefix, exempelvis k (kilo), h (hekto), d (deci), c (centi) och m (milli) m.fl., betecknas med gemena bokstäver ur det latinska alfabetet. Ett undantag är grekiska alfabetets my för mikro. Dessutom är hos oss åtminstone de flesta av nämnda enligt standard förkortade prefix lagligt fastställda i kombinationer med exempelvis m och g för mått och vikt.

Förkortningen Kb för kilobyte är sålunda olämplig och möjligen olaglig medan gement k i kB överensstämmer med svensk standard.

För kilobit torde kbit vara den enda möjliga förkortningen.

<5595>
Stig Thorén

FACIT Twist

Till DS90 har jag skaffat en FACIT Twist-terminal med vit bildskärm. Det är samma typ av bildskärm som finns på Nokia 1200:an. Den har vit bakgrund och texten är svart. Skärmen är helt flimmerfri och har mycket fin upplösning.

Varför det blev just Twist beror på att det går att vända på skärmen och istället för det normala 24 rader köra med 72 rader. I båda fallen är det 80 tecken per rad och en statusrad. Detta är mycket bra vid textbehandling. Just nu skriver jag så här när jag skriver den här texten. Jag ser då text för ett helt A4-ark i stället för att bara se en del av texten som det blir när man kör på en vanlig skärm som bara kan köras i landskapsmode. FACIT kallar det för poträttmode när skärmen står på högkant. Nästan alltid kör jag terminalen i poträttläge.

Det är möjligt att från programvaran fråga Twistterminalen i vilket läge skärmen står och få svar på detta och därefter sätta parametrar. Körs skärmen i landskapsmode kan man växla mellan tre olika sidor.

Tangentbordet är av hög klass. Jag skulle önska ett sådant tangentbord till min AT! Inte minst av det skälet att jag då skulle hitta tangenterna på samma ställe. Noteras kan att DTC-2 är baserat på samma hårdvara som Twist om jag inte minns fel.

DIAB:s ordbehandlingsprogram till DS90, LEX68, gör sig mycket bättre om man kör det på Twist än på en vanlig PC/ABC med VT100-emulator eftersom Twisten har extra funktionstangenter som inte är standard VT100. Dessa tangenter behövs när man kör LEX. Dessutom är det bra att kunna se en hel 72-raders sida på en gång. Det finns 12 extra tangenter och dessa kan kombineras med SHIFT och CTRL varför de blir 36 olika varianter. Kombinationen med CTRL kan dock ej laddas från program utan enbart från setupläge.

Twist saknar fläkt, vilket inte bör finnas i en modern terminal! Vad jag vet är Twist den denna terminal på markanden som går att köra med skärmen på högkant! Det förutsätter så klart att programvaran är anpassad för en bildskärm med 72 rader.

<1789>
Bo Kullmar

Folk och rövare

I somras visade TV en film om Mästerdetektiven som bygger på en av Astrid Lindgrens böcker. En poäng i berättelsen var att man lyckades fånga in en mördare genom att barnen kunde tala "rövarspråket" med varandra utan att mördaren förstod vad de sa, ja t o m utan att han förstod att de sa något till varandra.

Rövarspråkets ide är att man förvränger orden genom att upprepa varje konsonant med ett "o" emellan. "Jag" blir "jojagog" och "kan" blir "kokanon".

I slutet av historien har gänget "vita rosen" bestämt att de måste lära också sina antagonister "röda rosen" att tala rövarspråket. Vem vet när det uppträder en mördare nästa gång? Inlärningen går trögt, som det troligen skulle göra för oss alla.

Här kommer ett litet ABC80-program till vår hjälp. Det är en liten övning i stränghantering.

PROGRAMMET RÖVARE

På rad 70 skriver du in den text som skall "översättas" till rövarspråket. Man måste använda INPUTLINE, annars tas alla mellanslag bort, och de är ju viktiga i en löpande text.

INPUTLINE behåller också tecknen för RETURN i slutet av strängen, dvs radbyte (ascii=13) och "åter till början av raden" (ascii=10). Dem vill vi normalt inte ha kvar, så de måste opereras bort. Det kan man göra med

```
A$ = LEFT$(A$,LEN(A$)-2)
```

LEN(A$) ger längden på strängen A$. De två sista ascii-koderna skall bort, så den nya strängen A$ innehåller de tecken som står till vänster i den gamla, fram till längden minus 2.

Men nu skall vi ändå bearbeta tecknen för att få reda på vilka som är konsonanter och skall fördubblas, så man kan kombinera nytta med nöje.

MID$(A$,x,y) ger oss en delsträng av A$, räknat fr o m tecken x och ytterligare y tecken framåt.

På rad 110 går vi igenom strängen A$ ett steg i taget. I slingan får I löpa från 1 till LEN(A$)-2 - då blir vi automatiskt av med "svansen" RETURN. Det tecken vi för tillfället plockat ut läggs i B$.

VOKAL ELLER KONSONANT?

Hur skall vi veta om B$ är konsonant eller inte? Datorn förstår inte det, utan måste ha fullständiga instruktioner. Eftersom vokalerna är färre än konsonanterna är det enklare att testa på vokal.

På rad 130 står

```
INSTR(1,'eiyäöaouå',B$)
```

vilket betyder: titta om du kan hitta strängen B$ i strängen 'eiy...å', och om du gör det, tala om hur många steg in du får träff!

Om B$='y' skall således uttrycket ovan bli 3, och om bokstaven är 'å' blir det 9.

Om det INTE blir träff, blir uttrycket noll.

Nu är vi inte intresserade av vilket värde uttrycket får, bara om det blir träff eller inte. Rad 130 säger att om det blir träff (alltså INSTR(...)<>0) då är B$ en vokal och den skall vi inte ändra på. Vi samlar ihop den nya frasen i C$:

```
C$ = C$ + B$
```

Vi lägger alltså bara till B$. Men om det blir bom, då vet vi att B$ INTE är en vokal. Är det en konsonant så skall den dubbleras med ett "o" emellan:

```
C$ = C$ + B$ + 'o' + B$
```

Men är det verkligen en konsonant om det inte är en vokal? Dess värre inte alltid. Det kan ju t ex vara ett blanktecken mellan orden. Det alternativet måste vi fånga upp. I programexemplet har vi lagt till både blanksteg, punkt och frågetecken bland de tecken som skall undantas från dubblering.

Sedan är det bara att tuta och köra!

KOMPLIKATIONER

Givetvis kommer du snart att upptäcka att programmet behöver förbättras. Hur är det med versaler (stora bokstäver)? Har vi fått med alla vokaler? Skall man verkligen köra dubbeltecknade konsonanter två gånger? (Det är ju trots allt ett talspråk.) Hur gör man med ord som "och"?

Ojoj, här finns mycket utrymme för egna experiment och förbättringar. Ha så kul!

<1384>
Sven Wickberg

```
10 REM Rövare
20 REM
30 REM liten övning i stränghantering
40 REM på grundval av rövarspråket
50 REM hos Astrid Lindgren
60 REM
70 ; 'Skriv din text: ' : INPUTLINE A$
80 ; : ;
90 REM
100 FOR I=1 TO LEN(A$)-2
110 B$=MID$(A$,I,1)
120 IF B$=' ' THEN C$=C$+' ' : GOTO 140
130 IF INSTR(1,'eiyäöaouå',B$)<>0 THEN
    C$=C$+B$ ELSE C$=C$+B$+'o'+B$
140 NEXT I
150 REM
160 ; 'På rövarspråket blir det: '
170 ; C$
```

Facit till övningsuppgiften i programmet DELAFIL

För att varna för överskrivning av en befintlig fil kan man tillägga exempelvis:

```
145  ON ERROR GOTO 160
150  OPEN <filnamn> ASFILE 99:CLOSE:
     GOSUB 500
155  ON ERROR GOTO 1000
```

Om datorn inte hittar <filnamn> på rad 150 finns filen inte och felhanteringen från rad 145 skickar programmet vidare till 160.

Finns filen då lyckas rad 150 och man hamnar i stället på 500, där det lämpligen skrivs en varningstext som låter användaren välja mellan att skriva över den gamla filen eller börja om på rad 110.

Variationer finns!

SW

"Godkänd" leksak

En dator är något onödigt dumt som de vuxna måste ha, speciellt när dom vill fortsätta att vara barn. För tänk vad ska de annars leka med, innan de får sina egna barn? Och vad ska de leka med när barnen växt upp, när bilbanan är borta tåget trasigt osv? Datorn ersätter barnens leksaker och till på köpet en "godkänd" leksak åt vuxna barn. En dator sköter de om som den goaste nallebjörn, köper mer "juveler" till den än till tjejen/sambon/frun, för vem skulle inte vilja bli bortskämd på samma sätt som en dator? Det köps nya kort med alla möjliga finesser, ta till exempel alla dessa disketter! Vad nu dom är bra för, en liten tunn svart plastbit som inte ens är snygga. Och inte kan man använda dom till något vettigt, skulle duga perfekt om man vek dom på mitten som bokmärken.

Det värsta börjar när de skaffar små "syskon" som de döper till printer och Winchester som får smeknamnet Winch. Då vet man att de är förlorade för omvärlden långa perioder åt gången. Försöker man då vara med på ett hörn blir man bortsnäst vänligt men bestämt, råkar man då vara envis blir det lätt ett gräl. Ytterligare en sak som säger att en datorägare är ett vuxet barn. Delar inte med sig av sina leksaker precis som småbarn.

<6830>
Kerstin Jansson

ABC80 i ny skikkelse

Jeg er sikkert mellem de få der stadig finder anvendelse for den gamle ABC80, måske skyldes det nostalgi eller muligvis også det forhold, at der efterhånden er investeret ikke så få penge i den gamle computer. Den rigtige sandhed er nok den, at de mange års viden i ABC80'ens konstruktion og virkemåde, har givet en sikkerhed og tryghed i anvendelsen og udbygning, at alt dette kan nulstilles, såfremt man går over til ny PC'er.

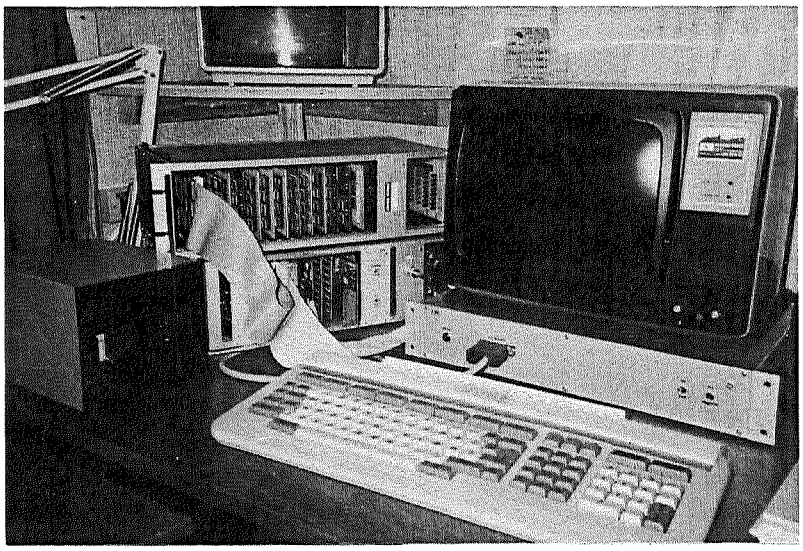
Jeg har gennem de sidste år udbygget min ABC80, at det eneste der er tilbage, er skärmen. Denne er dog også suppleret med en höjoplöselig monitor til min grafik.

Udbygningen er kommet i flere etaper, i afhängig af mit behov. Strömforsyningen er switch-mode power supply 5 volt/15 amp, +-12 volt/8 amp, dette at strömforsyningen er switch-mode vil sige, at det ikke bliver en stegepande, som en almindelig analog strömforsyning. Denne forsyningen däkker t behov både i den nuvärende konfiguraon og til flere udvidelser.

ABC80 er ombygget og ligger nu i sin egen kasse, med tilpasning til et intelligent keyboard, 2 serielle udgange, 2 karaktersät og videoudgang til monitor. I kassen er der plads til diverse ändringer f.eks CPM og flere Basic-tolke.

Keyboardet kan köre 8-bit + strobe, men körer 7-bit som normalt keyboard. Det har sin egen Z-80 og E-prom til ASCII-koder. Denne E-prom har jeg ändret til mit eget behov for nummerisk tastatur, skärmeditor tastatur og med funktionstaster, så man kan glemme alt om control-koder.

Min FD2D er savet over til halv störrelse og strömforsyningen er smidt langt väk. Kontrollerkortet er isat min ekspansionsboks sammen med printerinterface. Prommerne med DOS og printerrutinen er fjernet og overfört til et specielt Ram/Eprom kort fra adr. 16384 til 32768, således at behov for ändringer f. eks til UFD-DOS og andet kan afprövet i ram för endelig bränding af -prommer. Det samme er udfört med et Ram/Eprom kort med adresserne 0 til 16384, så der mulighed for en vis form for eksperimenter. Når prom med DOS og printerrutine,



som adresseres af ABC80 fjernes, skal chipselect til disse kredse også ändres, ellers sker der dobbelt adressering, og det er noget, der ikke tillades, derfor skal alle kort og adresseringer väre fuldständige.

Printere og plotter er tilsluttet via omskifter og fälles buffer på 256 K-byte. En buffer er uundvärlig, når man först har prövet at vente på lange printerudskrifter, derfor er der rart at sidde og arbejde videre med sin computer, selvom printeren körer videre i nästen 1/2 time.

Mit modem er et auto dial, d.v.s. softwaremässigt styret telefonopkald, derfor har jeg fundet det interessant af bygge et telefonkatalog op på et kort med EE-prommer. Jeg havde först lavet det på normale E-prommer, men de skulle for ofte ändres og opdateres, at jeg valgte, at pröve EE-prommer. Jeg har lavet nogle skriverutiner (SAVE) til dette kort.

Dette kort sidder i en af mine 2 ekspansionsbokse, den ene er forbeholdt normale ABC80-bus kort (64 pol AB), og den anden er min egen standard (64 pol AC).

I AB-boksen sidder strömforsyninger, kontrollerkort, printerkort, E-prom bränder kort og mit grafikkort. Grafikkortet körer med sin egen processor og monitor, og denne lösning er god at arbejde med, idet ABC80 er arbejdsterminal og resultatet af arbejdet og instruktionerne ses på grafik-monitoren (512 x 256 punkter x 4 sider).

I AC boksen har jeg lavet min egen standard til et antal 64 K-bytekort. Kommunikationen med disse kort sker via et styrekort, tidl. beskrevet i bladet. Dette kort sätter banknr. mellem 0 og 255 samt adresseområde mellem 0 og 65535, således at önskede data kan overföres til ABC'ens memmory. Denne boks består efterhånden af 1 Mega-Byte E-prommer, 320 K-Byte ram, 64 K-byte EE-prom, hvorpå det endnu kun er de 32 K der er isat, de er dyre sådan nogle EE-prommer. Desuden har jeg lavet det ene 64 K-ram kort med dipswitche, således at dette kort kan omstilles til 16 forskellige banker, det har betydning når jeg vil afpröve nogle programmer med de eksakte calladresser, inden jeg bränder E-prommer.

Fremtidige önsker er at udvide med CPM, som jeg forventer at väre färdig med inden jul, desuden er jeg begyndt at pröve med UFD-DOS, idet jeg har hentet programmet fra Monitoren i Sverige. Derudover er det mit store önske at få tid til at lave nogle maskinprogrammer for Z-80, specielt et maskinprogram til skärmprint af grafik og tekst, det jeg har lavet i basic, er alt for langsomt.

Desuden kunne jeg tänke mig ...., ja sådan kunne man blive ved, men det spändende er vel at ABC80 stadig giver behov for interessante opgaver.

<198>
Flemming Baagöe
Söndergade 16
DK-4130 Viby Sj.