

radio &
television

6

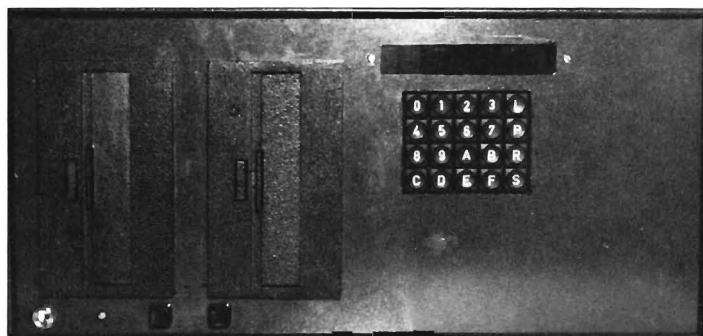
BYGG SJÄLV

DATORER

- Bygga själv eller köpa färdigt? • Fullständiga datorbeskrivningar
- Datorns funktion och programmering • Klubbar, litteratur och andra tips



Vi har byggsatserna till RT-datorn



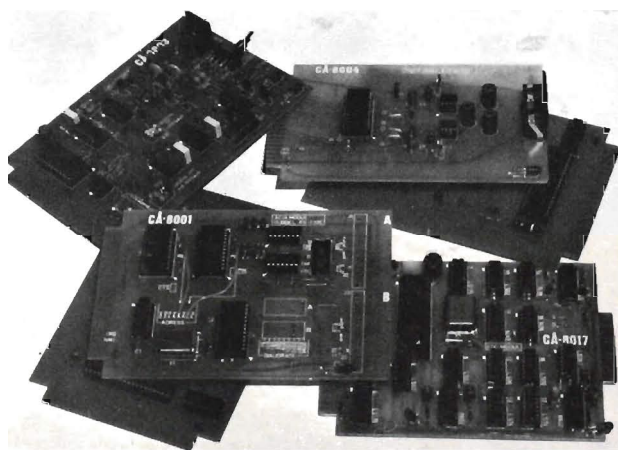
RT-datorn från CÅ-Elektronik är utvecklad för att användas som hobbydator och avancerad privatdator.

Används även av många företag som utvecklingssystem för 6800 och 6809.

Till RT-datorn finns ett stort urval periferikort för olika applikationer.

Exempelvis

- 8001 2 st ACIA med V24-snitt
- 8002 2 st PIA
- 8003 KCS-modem 300/1200 baud
- 8004 Programmerbar timer 6840
- 8005 Digitalklocka MM 5318
- 8006 Hex-display & tangenter
- 8008 16 ing. A/D-omvandl. 8 bit.
- 8010 Eprom-programmerare
- 8013 Korthållsmodem 300 baud
- 8016 Ljudeffektkrets SN 76477
- 8017 Flexkontrollkort
- 8018 IEEE-488-bus anpassning
- 8020 Debug-kort
- 8091 Labkort för PIA
- 8092 Labkort för ACIA/SSDA/ADLC
- 8095 Förlängningskort



Programvara till RT-datorn finns för de flesta tillämpningar: Monitorprogram för 6800/6808: ECBUG, TBUG1, TBUG2 och TBUG3. Monitorprogram för 6809: FOBUG, CBUG2 och CBUG3.

Flexskiveoperativsystem från TSC: FLEX 2 för 6800/6808 och FLEX 9 för 6809.

Assembler, editor, basic, utökad basic, textprocessor, sorteringsprogram mm finns för både 6800/6808 och 6809.

CÅ-TELKO_{AB} EH:s

ELEKTRONIK AB
Box 2010
135 02 TYRESÖ
Tel. 08-742 34 01

S:t Eriksgatan 15
Box 12011
102 21 STOCKHOLM
Tel: 08-54 18 40

S. Promenaden 78
Box 6097
200 11 MALMÖ
Tel. 040-723 90

Thorburnsgatan 5
Box 14120
400 20 GÖTEBORG
Tel. 031-83 03 10

ELEKTRONIK
Kungsängsgatan 10
721 30 VÄSTERÅS
Tel. 021-12 03 65

**radio &
television**
BYGG SJÄLV
NR 6

© Radio & Television 1980
Omslagsfoto: Claes-Göran Flink, Kamerabild
Redigering och bearbetning: Gunnar Lilliesköld
Formgivning: Håkan Widestedt
Utgiven av Specialtidningsförlaget AB, Stockholm 1980
Tryckt hos Kristianstads Boktryckeri AB, Kristianstad 1980
ISBN 91-7274-108-2

SPECIALTIDNINGSFÖRLAGET AB

Innehåll

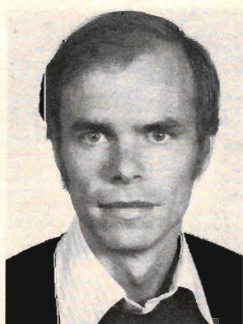
Vad kan datorn utföra?	4
Datasystemets delar. Alternativa lösningar.	8
Bygga själv eller köpa färdigt?	11
Mikrodatorns talsystem.	20
Hur fungerar mikroprocessorn.	22
Datorns monitorprogram.	26
Kassettband och flexskiva som massminnen.	32
Presentation av RT-datorn.	37
Några tips inför bygget.	39
Datorlåda, moderkort och nätdel.	40
Processorkortet – datorns hjärna.	47
Enkel terminal för maskinkodprogrammering.	50
Kassettpassning för den enkla terminalen.	54
48 kbyte minneskort.	56
Seriekommunikation med videoterminal.	59
Intelligent videoterminal med färg och grafik.	61
Använd tv:n som dataterminal!	73
Programmerbar räknare.	75
Labkort för PIA och ACIA.	77
Analog/digitalomvandlare.	79
Flexskivsystemet.	82
Skrivare för datorn.	86
Basicdator på ett kort.	89
Maskinkodprogrammering.	98
Assemblerprogrammering.	100
Bli bekant med basic!	102
Färdig programvara för RT-datorn.	108
Högnivåspråk för smådatorer.	114
Ordlista.	116
Datorlitteratur.	119
Datorklubbar.	122
Appendix: Kretskortmönster, ASCII-kodtabell, processkoder m m	124



Gunnar Lilliesköld är bokens huvudredaktör och tillika författare av en rad kapitel om bl a mikrodatorns funktion, talsystem, dess användningsområden, högnivåspråk för mikrodatorer och en sammanställning av marknadens datorer att köpa eller bygga. Sedan 1971 är han anställd vid RT och där verksam som andre redaktör.



Åke Holm har, med ett undantag, tagit fram alla konstruktioner till datorerna i boken. Till grund för kapitlen ligger en uppmärksam byggserie i RT. I 8 års tid har han medverkat med byggeskrivningar av olika slag för tidningen med bl a avancerade instrument som frekvensräknare och färgbalkgenerator.



Tommy Bladh orienterar om programmering i maskinkod eller assembler samt lagring av data i massminnen. Han har även tillsammans med Åke Holm tagit fram avancerade monitorprogram i PROM, för datorbygget. Till vardags är han verksam med programmering av mikrodatorer vid Ellementel.



Bengt Grahn beskriver i boken hur man på ett enda kretskort kan bygga en dator för ett slags högnivåspråk, kallat NIBL. Han har specialiserat sig på datorer med SC/MP-processorn. Verksamheten är idag inriktad på industritillämpningar.



Bertil Hellsten ligger bakom kapitlet om basic-programmering. Förutom hängiven basic-programmerare, i egenskap av RT:s facktekniske redaktör, är han även mycket engagerad i bandinspelning av ljud- och bild.

Förord

Datatekniken har under det senaste decenniet genomgått en så snabb utveckling att den saknar motsvarighet i historien. Ett datorinköp var för tio år sedan en affär som ekonomiskt sett bjöd på en rejäl omskakning åtminstone för det medelstora företaget. För småföretagaren var en egen dator något helt otänkbart.

Idag är prisbilden en annan tack vare mikrodatorns tillblivelse. Nu kan även privatpersoner köpa en dator.

Den elektronikintresserade bygger naturligtvis sin dator själv. Därmed öppnar sig intressanta möjligheter att bygga datorsystem för specialtillämpningar av olika slag och i en takt som passar den egna kassan. Totalt sett kan det bli billigare än att köpa en färdig dator och så ger det en mängd kunskaper om hur datorn fungerar och kan användas. Kunskaperna är värdefulla inte minst om man behöver felsöka i datorn eller bygga ut den i framtiden.

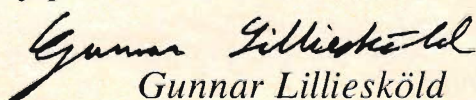
Den här boken handlar just om hur man själv bygger sitt datasystem. Tre grundvarianter beskrivs, ett litet nybörjar system, en stor basic-dator och en basic-dator på ett kretskort. En uppdelning i moduler gör att datorerna kan byggas ut efter önskemål.

Man börjar lämpligen med att bygga ett prisbilligt system som arbetar i maskinkod. Det kan sedan byggas ut för basic och andra programspråk.

Konstruktionerna är presenterade med kretskortsmönster så att den som själv vill göra sina kort har möjlighet till det, men färdiga mönsterkort säljs även till samtliga presenterade konstruktioner. Till de två datorer som arbetar med 6800/6809-processorerna finns byggsatser och till den dator som har processorn SC/MP finns monterade och provade kort.

Boken innehåller även kapitel som orienterar om hur datorn fungerar, hur den programmeras i olika språk, vad den kan användas till och vad för slags specialprogram som finns till en av de beskrivna datorerna. Vilka möjligheter det finns att bygga upp ett datorsystem går vi också in på med en kort genomgång av byggsatser och färdiga datorer på marknaden.

Den här boken är avsedd som en introduktion i ämnet, ägnad åt framför allt självbyggare, men för den som vill utveckla hobbyn vidare ger vi dessutom anvisningar på lämpligt litteraturval. Vi hoppas att boken skall ge dig stimulans till att tränga djupare in i datorernas fantastiska värld.


Gunnar Lilliesköld

Vad kan datorn utföra?

AV GUNNAR LILLIESKÖLD

Vad kan man göra med datorn man byggt eller köpt? Kan datorhobbyn kombineras med andra områden? Vad går att utföra med datorer i olika utbyggningsfaser? Vi ger en kort orientering i frågorna.

■ Många menar att datorns genombrott är lika kraftigt som ångmaskinens var under 1800-talet. Kanske får den lika stora sociala konsekvenser som den industriella revolutionen med kraftiga vinstökningar hos företag, nyetableringar, stora tekniska framsteg men kanske också problem med omskol-

ning genom att nya yrken kommer och gamla försvinner med utslagning som följd. Säkert kommer en rad lagar att stiftas av ekonomisk, fördelningsteknisk natur. Svartmålningen har dock gått till överdrift i många fall där man bortsett från fördelarna med det nya redskapet. Ja, datorn kan faktiskt betraktas som ett redskap för hjärnan på samma sätt som mekaniken blev ett redskap för musklerna under den industriella revolutionen. Det finns idag många exempel på hur datakraften tillämpas, men ändå är detta bara början på en utveckling . . .

Det som kom att revolutionera hela datatekniken var mikroprocessorn. Den första såg sitt ljus re-

dan 1971, framtagen av Intel i Californien. De första mikrodatorerna var tänkta att användas för diverse styrändamål men med tiden har de blivit så snabba och har så hög kapacitet att de ingår i minidatorer. Därmed har gränsen mellan mini- och mikrodatorer blivit flytande. Man talar hellre om smådatorer idag. Privatdator och persondator är andra begrepp för samma sak.

Datorn som styrdon

Mikrodatorn är som sagt från grunden tänkt att användas som styrdon. Industrin använder så mikrodatorn som programverk i t ex tvättmaskiner och andra hushållsmaskiner, i programstyrda verktygsmaskiner och robotar i så-



Fig 1:1. En av de mera spektakulära tillämpningarna av datorn är skapande av musik. Anläggningen på bilden tillverkas av ALF, USA. Musiken genereras i ett kretskort som passar i datorer med S-100 buss. Märk presentationen av noter på bildskärmen.

kerhets- och övervakningssystem.

Vad har då styrdatorn för tillämpningar i amatörsammanhang? Ett intressant projekt är reglering av värmeelementen i villan. Datorn samlar in uppgifter om temperatur i de olika rummen, kanske även yttertemperatur. Den tar del av de inprogrammerade uppgifterna om vilken temperatur de olika rummen skall ha och klockslag så att man kan få olika temperaturer vid olika tidpunkter. Datorn bearbetar uppgifterna och ger kontinuerligt signaler till de olika elementen för reglering av temperaturerna. En invändning mot ett sådant system är att S-märkning kan bli ett problem och till följd av det ett försäkringsproblem.

Ett annat projekt är styrning av modelljärnvägen. Datorn styr växlar, signaler, bommar och lok efter ett program. Kanske är det ett lämpligt projekt för förälder och barn?

I USA har robotarna blivit populära att bygga, medan intresset kanske inte varit lika stort på den här sidan Atlanten. Inspiration till byggena har man oftast fått från Star trek och liknande science fiction-filmer.

Ett område är datorn som spel. På ett litet enkelt utvecklingssystem kan man faktiskt utföra en hel del spel, även om kapaciteten naturligtvis är högre i de större data-systemen.

Datorer med tv-terminal

De enklaste datorerna har bara en indikator för några siffror och ett tangentbord avsett för siffror från 0 till F. Det låter kanske litet konstigt, men det innebär 16 tecken i det s k hex-kodsystemet. Vi kommer till olika talsystem i ett senare kapitel. Den typen av dator är avsedd för maskinprogrammering och den kan användas främst som styrdon.

För litet mera komplicerade program bör man ha en mera utbyggd dator. Då kan man arbeta i andra språk som assembler, basic m m. Programmeringen underlättas därmed avsevärt.

Tangentbord och indikatorer måste då kunna arbeta med både

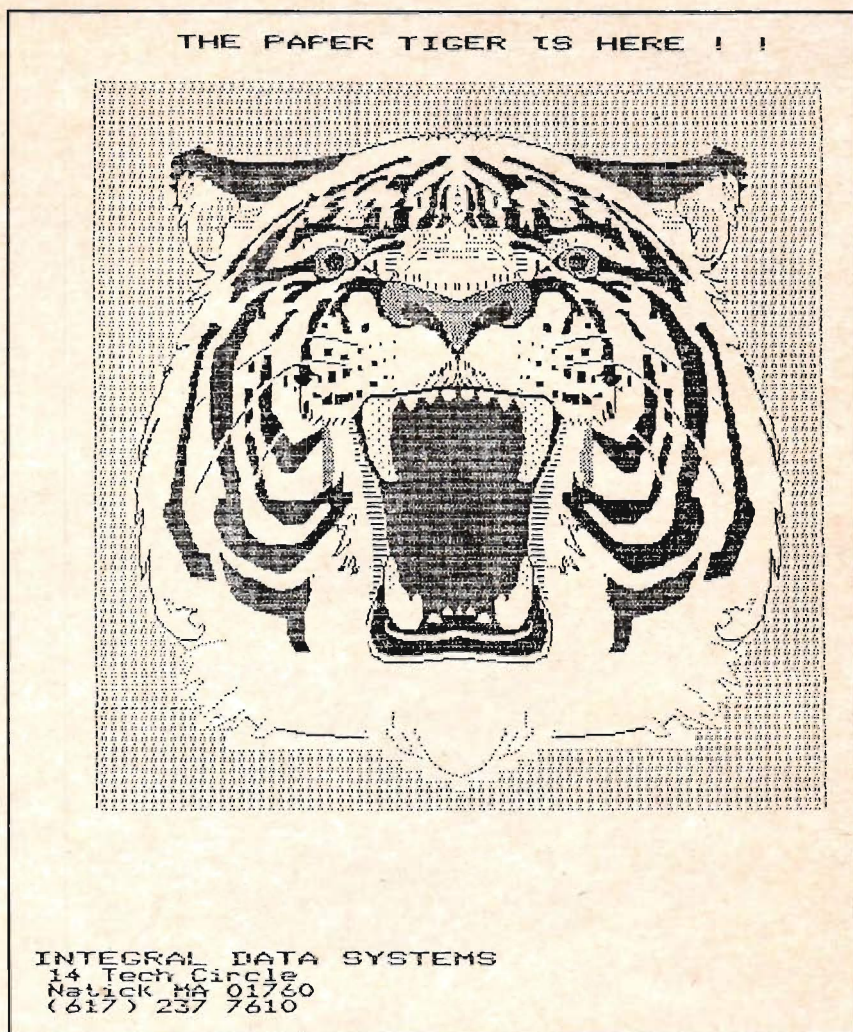


Fig 2.2. Ett exempel på grafisk återgivning med dator. Skrivaren (inte x-y plottern) är av fabrikat Epson och bilden är genererad av en Apple II-dator. Upplösningen är som synes ganska god. Ursprunget är som synes gjort för skrivaren "Paper Tiger".

siffror, från 0 till 9 och alfabetet jämte specialtecken. Minnet i datorn måste vara större eftersom åtminstone basicprogrammen blir längre. Ibland låter man interpretatorn ligga i maskinens minne (RAM). Den laddas då in efter start från en hålremsa, bandkassett eller s k flexskiva. Vissa datorer har interpretatorn lagrad i ROM. Hur en dator av denna kategori jämte en dator för maskinspråk är uppbyggda framgår i boken i utförliga byggbeskrivningar.

En dator för basic kan användas för mycket. Assemblerspråket är också användbart men det är svå-

rare och mera tidsödande att programmera. Å andra sidan kräver assemblerprogrammen mindre minnesutrymmen och kan göras snabbare varför de företrädesvis används i styrsystem. Den som programmerar mycket väljer också hellre assembler än maskinspråk.

Basic-språket för nybörjare

Låt oss så återgå till basic. Tack vare att språket är enkelt har det blivit allmänt accepterat och omtyckt av nybörjare. En dator med ett högnivåspråk som t ex Basic kan utföra en rad uppgifter:

- Söka i register. Man kan tänka

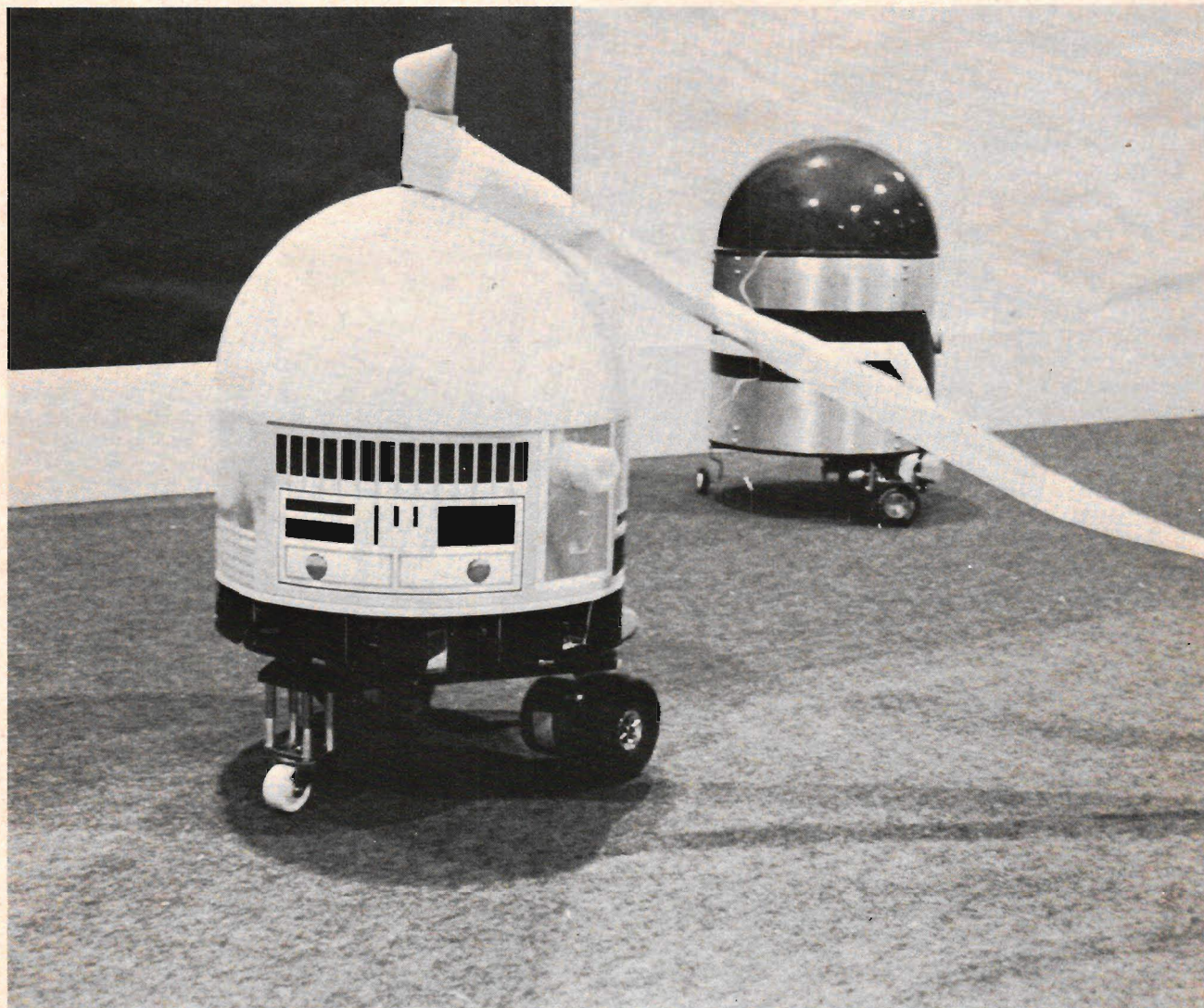


Fig 3:3. En robot har självfallet en mikrodator som hjärna. Bilden är tagen vid en amerikansk hobbydatormessa.

sig ha en lista över skivsamlingen med sökning på artister och melodier. Programmet utformas lämpligen så att det även kan sortera. En elektronisk telefonkatalog är en annan intressant uppgift. Rent tekniskt skulle man kunna koppla ihop datorn och telefonen för att få automatisk uppringning sedan man skrivit in namnet på den person eller det företag man skall ringa. Tyvärr är det inte möjligt i Sverige med de lagar som idag råder, men rimligen sker en utveckling av det slaget så småningom. Sökning i personregister kan vara lockande för föreningar, men där måste man tänka på att ha datainspektionens

godkännande av registret. Eventuellt kan registret kombineras med andra funktioner för att t ex få automatiska utskrifter av adressetiketter och därmed är vi inne på nästa punkt:

- **Ordbehandling** är en datatillämpning som blir allt vanligare. Det innebär att man använder datorn i stället för en skrivmaskin. På ett alfanumeriskt tangentbord (dvs med både siffror och bokstäver) skriver man sin text som syns på en bildskärm. Eftersom texten ligger i datorns minne kan man redigera i redan skriven text med utbyte och inlägg av bokstäver eller

ord. Till och med hela stycken kan byta plats och man kan få jämna vänster- och högermarginaler. Ren ordbehandlingsmaskiner kan mer än så. Arbetar man dagligen med text är ett sådant system att föredra framför ett ordbehandlingsprogram till en mindre dator. Har man dock redan en dator kan det vara ide att skaffa programmet för att få ytterligare en tillämpning av sin dator.

När texten är färdigredigerad ger man datorn instruktioner för att med en skrivare få sin text på papper.

- **Beräkningar** skall man lämpligen göra med sin dator. Med litet vana

är det faktiskt enklare att göra komplicerade beräkningar med en dator än med en programmerbar räknedosa. En bra Basic klarar inte bara de fyra räknesätten utan även trigonometriska funktioner, logaritmer (10-logaritmer eller e-logaritmer eller ibland båda), exponenter och kvadratroter. Resultaten kan ev presenteras i tabellform eller som kurvor. Med de vanliga tecknen kan en grov grafik åstadkommas. Vissa datorer har även grafiska symboler som medger relativt god detaljupplösning. Flyttal har de flesta basic-datorer. En del har även heltalsvariabler vilka kan användas i speciella fall där man vill ha snabba beräkningar. Med mycket stora krav på snabbhet väljer man kanske ett kompilerande högnivåspråk som t ex fortran eller pascal. Det finns även kompilerande varianter av basic som ger snabba exekvering.

● **Datorspel** har kommit att bli en stor tillämpning. Det finns gott om litteratur som innehåller listade spelprogram i basic. Det är inte alltid man kan skriva av programmen direkt. Ofta måste man göra några modifieringar för sin dator eftersom det finns många olika dialekter av basic-språket. För den som har lärt sig sin dator grundligt bör det dock inte vara några problem. Ibland kan programmet bli en smula längre då någon smart instruktion saknas i datorn. Månländningsprogram, star trek samt schackspel hör till de mest uppskattade datorspele. Spelprogram som komplement till datorn kan hela familjen ha glädje av. Inte minst därför att det finns spel som riktar sig speciellt till de yngre ålderna.

● **Amatörradio** har idag fått ett naturligt komplement i datorn. Som sändaramatör måste man ju föra loggbok över utförda radiokontakter. Det är en utmärkt uppgift för datorn. Kompletterar man datorn med en skrivare kan man få ut listor ur loggboken när så behövs. Med sorteringsprogram kan man t ex ta ut vissa koder och lista dem. Man kan t o m tänka sig att datorn skriver ut QSL-korten!

Datorn kan även användas som en SSTV-terminal (Slow Scan Television). Ett omvandlingsprogram tar emot de i låg hastighet översända bilderna och omvandlar dem till vanligt tv-format så att de kan presenteras på datorns tv-skärm.

Radioteletype, dvs RTTY, kan också sändas och mottas via datorn. Man slipper de slammande skrivarorna och arbetar ljudlöst med presentation på tv-skärmen. Telegraf kan också sändas och mottas men det ställs naturligtvis krav på jämna, normalenliga morsetecken vid mottagningen. Telegrafprogrammet kompletteras lämpligen så att det kan fungera som övningsprogram.

Automatisk antenninställning är ett projekt som kan intressera de sändaramatörer som sysslar med månstuds- och satellitkommunikation. I ett program kan man beräkna banor hos satelliterna (Ja, månen är ju faktiskt även den en satellit!) och beräkna den rätta antenninställningen för varje tidpunkt och förutsättningarna vid olika tillfällen.

```

*****
* COMPUTERIZED QSL-CARD BY ABC00 *
*
* FROM *
* SK0BY *
*
* SSS *
* S S K K 000 AAA Y Y *
* S K K 0 00 A A Y Y *
* S K K 0 00 A A Y Y *
* S K K 0 00 A A Y Y *
* S S K K 0 00 A A Y Y *
* SSS K K 000 A A Y *
*
* TO *
* SM0DIS *
*
* DATE GMT RST FREQ MODE *
*-----*
* 790930-2035-567-145 525-FM *
*-----*
* E L F A 777777 333333 *
* RADIO & TELEVISION AB 7 3 *
* 171 17 SOLNA 3 *
*
* S W E D E N 7 3 *
*
* QRA-LOC JT 51 F 7 333 *
*****

```

Fig 4:4. Sändaramatörerna har många möjligheter att kombinera sin hobby med datorverksamhet. Man kan t ex göra en loggbok i form av ett basic-program och låta datorn göra utskrifter av QSL-kort, som på bilden.

I repeaterstationer har man redan nu börjat använda datorer som sköter om på- och avstängningar vid vissa tidpunkter och sänder ut stationskod och meddelanden. Kanske kan man i framtiden tänka sig att kommunicera med en dator via etern. Datorn får information från en mottagare och det bearbetade resultatet sänds åter ut. Datorn kan ge svar på frågor om olika tester, diplom, regler, radiofyrfrekvenser, dx-tips, adresser till andra sändaramatörer, rutiner för QSL-kort, klubbmöten och andra aktiviteter. Möjligheterna är många!

● **Bokföring** är en uppgift för datorn som främst småföretagaren har nytta av, men även privatpersonen har glädje av ett dylikt program. Det kan i så fall vara utformat för att passa hushållskassan med allt vad det innebär av inbetalningar av räkningar, räntor och amorteringar. Ett deklarationsprogram skulle vara idealet för många, speciellt om inkomsterna kommer från många olika arbetsgivare. Ett program skulle kunna hålla reda på att rätt skatt inbetalas under året för att slippa obehagliga kvarskatter. Det finns en hel del färdig programvara av det här slaget på marknaden. Räkna dock med att kostnaderna för program många gånger kan överstiga kostnaderna för själva datorsystemet! Som dataamatör får man i stor utsträckning göra sina program själv. Ett gott tips är dock att gå med i någon dataklubb. Därigenom kan man ofta få programvara till låg kostnad eller t o m gratis.

Användningsområdena är många för datorn. Vi har här bara nämnt en del. Eftersom de flesta bedömare hävdar att datautvecklingen nu bara befinner sig i sin början kan man följaktligen hävda att vi idag ännu inte vet alla tänkbara tillämpningar för en dator. Ett gott tips är att skaffa sig datorerfarenheter så snart som möjligt. Det ger goda personliga framtidsmöjligheter. Att bygga datorn själv ger naturligtvis för en tekniskt intresserad person extra värdefulla erfarenheter.

Datorsystemets delar

Alternativa uppbyggnader

AV GUNNAR LILLIESKÖLD

Här skall vi titta innanför skalet hos datorn på de huvudbeståndsdelar som ingår.

Vi skall också se på datorns förbindning med omvärlden, med terminaler, skrivare, kassettbandspelare.

■ Ett datasystem består förutom av en dator, av en terminal av något slag. Terminalen är den enhet med vilken man direkt kan kom-

municera med datorn. Det enklaste systemet, avsett för maskinkod-programmering är gjort för hexadecimal kod (Se talsystem), dvs för siffror från 0 till F. Följaktligen finns det bara 16 tangenter för siffrorna jämte ett antal tangenter för specialfunktioner. Den har två sifferindikatorer för data och fyra för adresskod. Hur de används kommer vi till senare i boken. För att kunna lagra våra program har vi anslutit en kassettspelare till datorn. Det finns olika metoder att lagra datainformation på band. I amatörsammanhang används vanligen en standard som kallas "Kassas City". Det finns vidare en rad olika standards för formattering av data. Mer om det i avsnittet lagring av data.

Datorns uppbyggnad från grunden

I fig 2:2 ser vi ett förenklat block-schema för ett datasystem. Härnan i datorn är *mikroprocessorn*.

Den utför ett antal instruktioner efter ett givet program, t ex gör matematiska beräkningar, hämtar och lämnar data. Vår enklaste dator består av fyra grundfunktioner: Mikroprocessor, minne, anpassningskretsar för terminal och för kassettbandspelare. De står sinsemellan i förbindelse via en *sk buss*. Det är ett antal parallella dragna (elektriskt sett) trådar som går till alla enheter. Från terminalen kan vi lägga in ett program i minnet som datorn sedan utför eller exekverar som man även säger. För att spara programmet lämpar man över det till *kassettbandspelaren*.

Ett komplett datasystem

Ett fullt utbyggt system ser ut som i fig 2:3. Till datorn har vi här anslutit en tv-terminal bestående av ett terminalkort som ofta är placerat i datorn), en tv-mottagare och ett alfanumeriskt tangentbord. Den typen av terminal är en förutsättning för att vi skall kunna arbeta i högnivåspråk eftersom det fordrar såväl siffror som bokstäver och andra tecken. Den kod man arbetar med kallas ASCII-kod.

Till datorn har vi som i det lilla systemet en kassettbandspelare ansluten. Ett betydligt snabbare medium är dock *flexskivan*. Det program som tar flera minuter att ladda från kassetten tar kanske bara några sekunder att ladda från en flexskiva. En flexskivenhet är ganska dyr i anskaffning, men många tar ändå den investeringen efter en tid eftersom datahanteringen förenklas så radikalt.

En *skrivare* kan vara bra att ha för att få programmen listade, utskrift av register m m. De billigare skrivarna har tecken uppbyggda med punktmatriser. Tecknen ser inte särskilt vackra ut men resultatet duger i många sammanhang. En *skönskrivare* arbetar med typer på hjul eller kula och ger lika god skrift som en skrivmaskin. Den är ett måste om man arbetar med ordbehandling. Tyvärr är priserna höga för skönskrivare: Mellan 10 000 och 20 000 kr får man räkna med. Det avskräcker nog de flesta

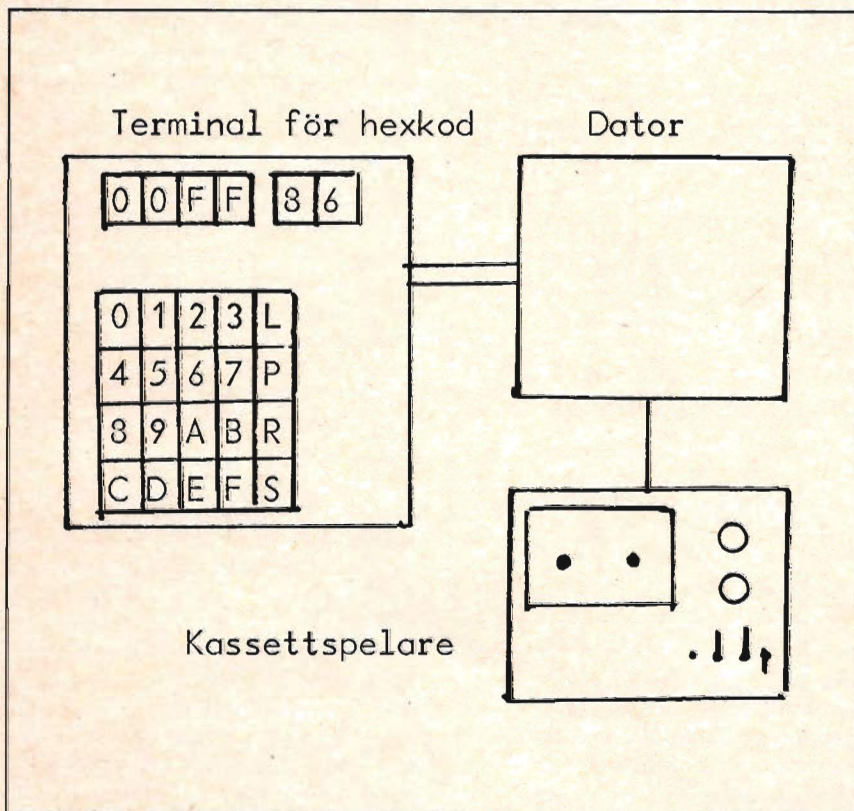


Fig 2:1. Det enklaste datorsystemet består av tre delar: Terminal för kommunikation med datorn och en kassettspelare som massminne.

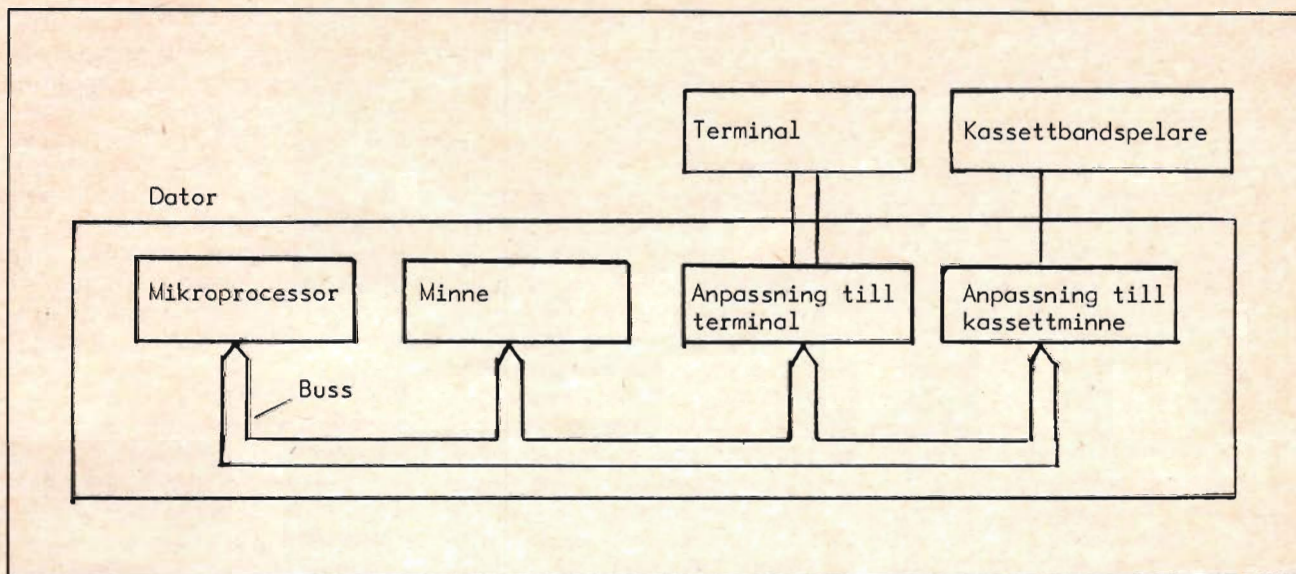


Fig 2:2. Ett förenklat blockschema för systemet i fig 1 ser ut så här.

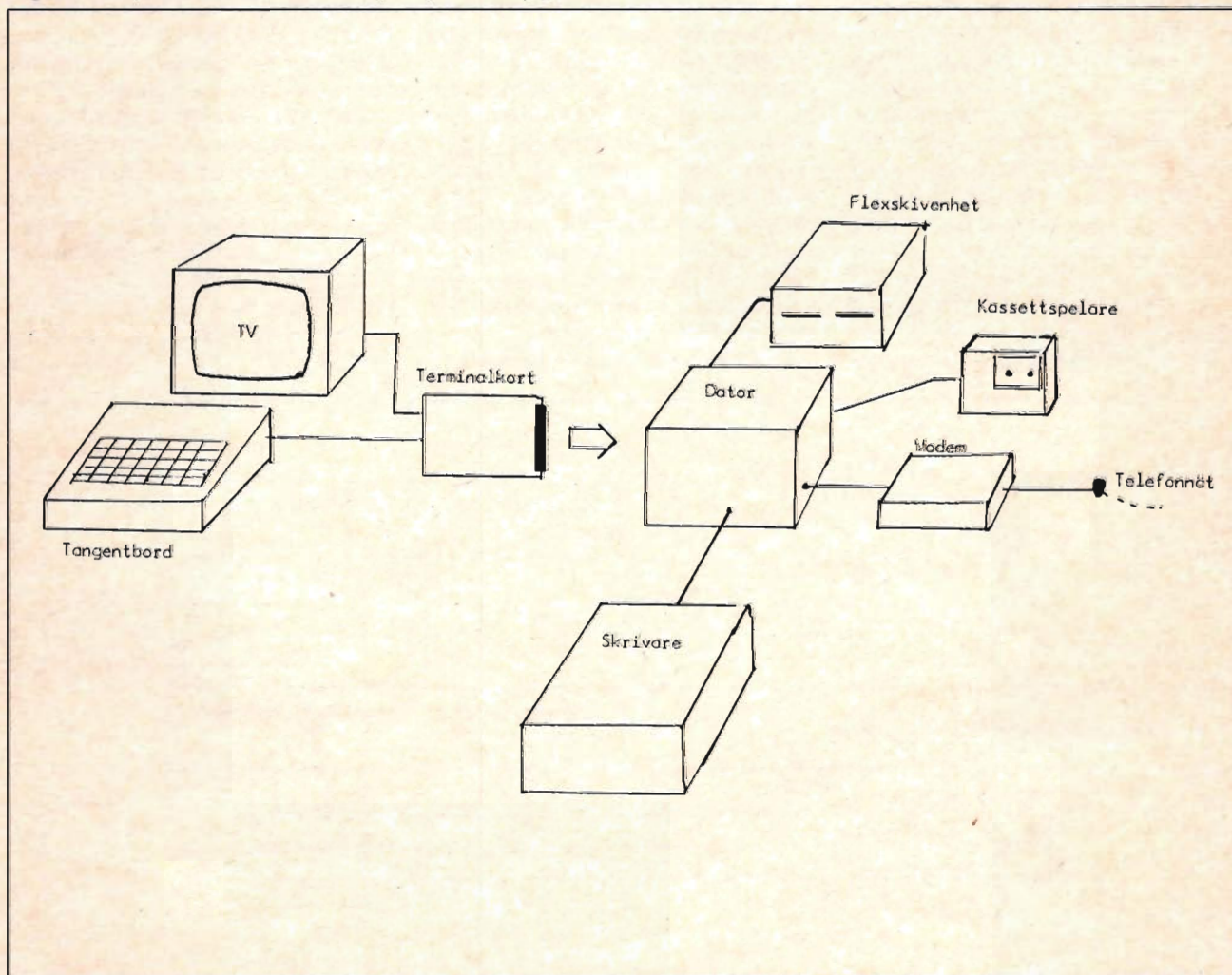


Fig 2:3. Ett fullt utbyggt system kan se ut på detta sätt.

dataamatörer. Ett alternativ är att bygga om en elektrisk skrivmaskin till skrivare. Det är dock inte så lätt som det låter. Anpassningskretsarna kan bli omfattande och därtill fordras en hel del programvara. Men det kan bli en billig lösning för den kunnige.

Modem är den sista biten i systemet vi skisserat. Det är en anpassningsenhet mellan dator och telefonnät. Med dess hjälp kan vi ringa upp t ex en stor datacentral och köra våra program på den om vår egen dator inte är kapabel för uppgiften, t ex om vi behöver arbeta i ett programmeringsspråk som vår dator inte har. Modem-enheten måste vara godkänd av televerket. Man hyr den lämpligen. Ett annat alternativ är s k akustiska modem. Även det kräver typgodkännande. Den är så utförd att akustisk koppling sker till vanlig telefonlur. Televerket godkänner dock bara att endast en av de kommunicerande abonnenterna använder akustiskt modem. Kommunikationen skall även avse "strikt portabelt bruk". Det utgör givetvis en begränsning för dataamatörer som knappast är beredda att lägga ut pengar för hyra av modem. Hur reglerna kan komma att se ut i framtiden är dock ovisst. Troligen kommer just

datakommunikation att bli ett intressant specialområde inom amatördatorverksamheten.

Begagnad teletype som terminal

I vårt utbyggda datasystem har vi använt en tv-terminal. Det är nog den bästa läsningen i amatörsammanhang. När de första mikrodatorkretsarna började komma ut var det dock många som använde teletype-terminaler. De består av skrivare och tangentbord. Har man tur kan man komma över en sådan begagnad men priserna är dock höga trots att maskinerna många gånger är slitna eller kräver översyn. De slarvar kraftigt och skriver ofta bara stora bokstäver. Anslutningen till dator sker med serieöverföring och 20 mA strömslinga. En annan vedertagen standard för terminaler är RS 232 eller V24-snitt. Att maskinen har det snittet innebär dock inte att den direkt kan anslutas till vilken dator som helst. Datorn måste ställas in för rätt överföringshastighet, udda, jämn eller ingen paritetsbit (kontrollbit vid dataöverföringen som ligger i varje ord). Man måste läsa specifikationerna för resp maskiner för att kunna koppla ihop dem.

Teletype-maskinerna arbetar i

ASCII-kod som har blivit en standard för datorer. Skilj på dessa maskiner och gamla RTTY-maskiner (Radio Tele *TY*pe). De senare arbetar med fembitars *Baudot*-kod. Principiellt sett skulle vi kunna ansluta en sådan till vår dator, men då måste den utrustas med ett fast liggande program (placerat i ett ROM) som omvandlar Baudot- till ASCII-kod. Nackdelen är då att vi får en begränsad teckenrepertoar.

Hållremsläsare ytterligare tillbehör

Vill man lagra program sker det idag vanligen på kassett eller flexskiva. Förr var det vanligen att använda hållemsa. Remsan är gjord av papper. Raden är osymmetriskt placerad så att remsan inte kan vändas fel. På vardera sidan på den finns det plats för tre resp fem hål, dvs totalt 8. Vi lagrar här alltså 8 bitars ord efter varandra.

För att kunna arbeta med remsa behöver vi dels en stans som gör hål i pappret, dvs en remsläsare. En läsare kan vara ett lämpligt komplement till datoranläggningen om man har tillgång till programvara på remsa. Hållemsan är annars ett medium som är på väg att försvinna till förmån för de snabbare flexskivorna.

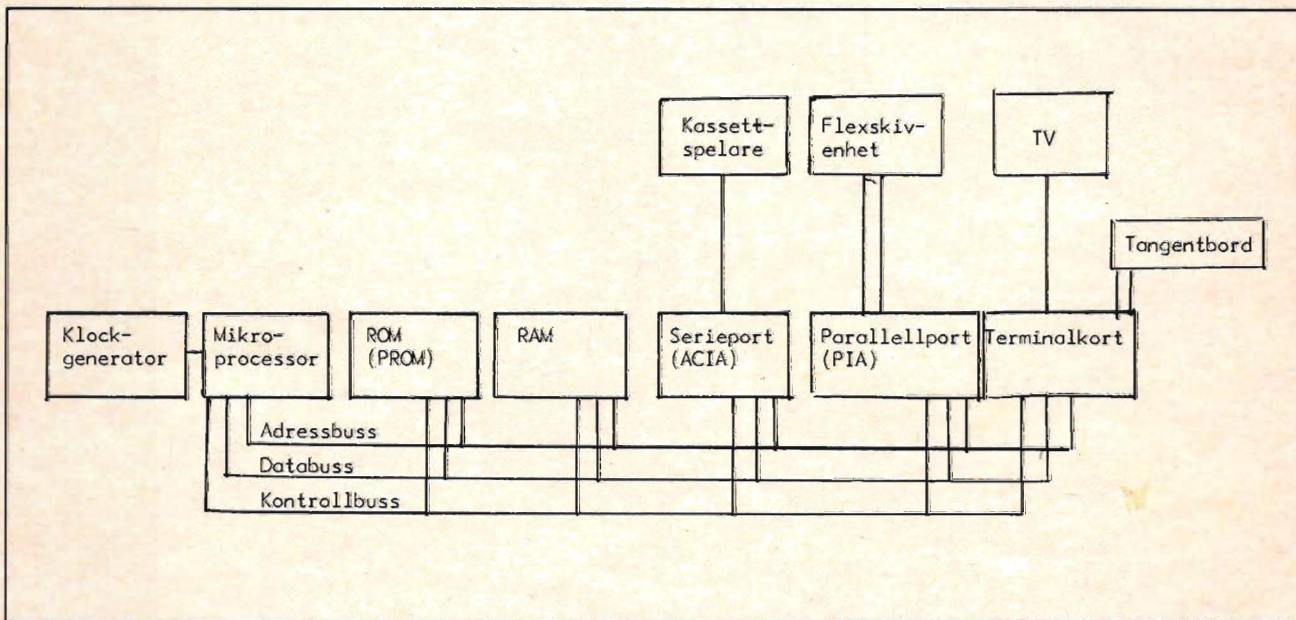


Fig 2:4. Blockschema för ett "stort" datorsystem. Ett sådant kan du bygga med utgångspunkt i bokens beskrivningar.

Bygga själv eller köpa färdigt? Många datorsystem att välja på

AV GUNNAR LILLIESKÖLD

Vare sig du tänker bygga datorsystemet själv, köpa en byggsats, ett färdigt datakort för inbyggnad eller en färdig, komplett dator måste du först analysera dina behov: Vad skall du göra med datorn, skall den kunna byggas ut, vad får det kosta?

Här ger vi en översikt av datorer i urval som kan vara lämpliga för amatör- eller semiprofessionella tillämpningar.

■ Det finns många sätt att gå tillväga då man vill bygga en dator. Först och främst får man bestämma sig för vad man vill göra med den. Räcker det med en liten mikroprocessor som arbetar med maskinspråk? Eller skall datorn arbeta i ett högnivåspråk? I så fall måste vi ha en terminal för alfanumeriska tecken? Hur stort minne behövs för de tänkta applikationerna? Skall man kunna ansluta kringutrustning och i så fall av vilken typ?

En annan aspekt som man bör ta ställning till är om man avser att bygga hela datasystemet efter ritningar, bygga en byggsats, köpa en dator på kretskort och bygga in eller köpa en komplett dator som senare kan kompletteras med tillsatskort och kringutrustning. Mycket är det en kostnadsfråga, men det är också fråga om vad man vill med sin hobby. Är man enbart intresserad av att programmera och saknar kunskaper om elektronik bör man naturligtvis satsa på en färdig dator. Med en smula kännedom om dator och kretsteknik kan något av de övriga alternativen vara mera attraktivt. Bygger man datorn själv, lär man sig funktionen hos de olika delarna. Därmed blir det lättare att felsöka i kretsarna eller

komplettera och bygga ut datorn. Vad som är väsentligt är att man kan satsa på ett relativt litet system från början och sedan bygga ut det undan för undan efter tillskott i kassan. Låt oss se på de olika alternativen i tur och ordning:

Att bygga efter ritning

Det är ett alternativ som den här boken tar sikte på. Datorbeskrivningar i elektroniktidskrifter världen över förekommer rätt sparsamt. En bärande orsak är naturligtvis tillgången till sådana beskrivningar. Det tar en oerhörd tid och det krävs självfallet mycket kunnande för att konstruera en dator och sedan dokumentera det hela.

Vi har här samlat ett antal beskrivningar av kretskort och delar som tillsammans bildar datorsystem. Konstruktör är Åke Holm som har utarbetat beskrivningar för Radio & Television under nära ett decennium när detta skrivs. Datorn, som alternativt arbetar med processorerna 6800 och 6809, är anpassad för bli en färdig programvara från TSC, Technical Systems Consultants Inc och från Motorola. En del programvara har också Tommy Bladh bistått med. Två grundkonfigurationer kan sammanställas med kretskorten i byggserien: En liten utvecklingsdator för maskinspråk och en större med tv-terminal som kan arbeta i assembler och högnivåspråk (och naturligtvis även maskinspråk). Systemet är så upplagt att man kan börja med det lilla systemet och sedan bygga ut det. Det stora systemet är mycket kraftfullt genom att det kan kompletteras med TSC:s programvara för *assembler/editor, basic, sortering och ordbehandling*. Ett holländskt programpaket för Pascal passar även datorn. Mer fakta om systemet finns i presentationen av RT-datorn.

Ett annat alternativ är att bygga det lilla datorkortet som Bengt Grahn beskriver. Här finns alla funktioner på ett kort. Datorn ansluts till en teleprinter-terminal för ASCII-kod (*Teletype*) och överföringen sker med 20 mA strömloop. På kortet finns en interpretator för språket NIBL. Det är ett *basic*-liknande språk som National Semiconductor tagit fram för sin mikroprocessor SC/MP. Språket har en hel del specialrutiner som förenklar programmeringen i de fall datorn arbetar som styrdon.

Till 6800/6809-datorerna finns kompletta materialsatser. Man kan även köpa kretskort och vissa specialkomponenter (PROM) löst. Till SC/MP-datorn och skrivaren finns ej materialsatser. De får man antingen bygga på egen hand eller köpa som färdigbyggda, provade enheter.

Byggsatser

Byggsatser i ordets rätta bemärkelse tillverkar Heathkit. De är upplagda för att klaras av rena noviser. Dokumentationen är synnerligen god med steg-för-steganvisningar. Den modernaste konstruktionen i programmet är H89. Det är en dator, sammanbyggd med en intelligent videoterminal och en flexskiveenhet. Två processorer ingår: En för datorn och en för terminalen. Den senare kan även köpas separat och passar då till de separata datorerna H8 (med 8 bitars ordlängd) och H11A (med 16 bitars ordlängd).

Till H89 kan anslutas en enhet Z87 som innehåller två ytterligare flexskiveenheter. Totalt har man då snabb tillgång till 300 k byte data. För datorerna H8 och H11A finns flexskiveenheterna H17 resp H27. Den senare är gjord för 8" flexskivor. Till samtliga enheter passar skrivaren H14.

Språken omfattar assembler och

basic för alla datorerna. Som tilllägg kan man få Fortran. Till *H11A* passar **Digital Equipments** programvara till deras dator *DEC PDP-11/3*. Datorerna har nämligen identisk funktion, men priset skiljer avsevärt. Till den nya *H89*, eller dess färdigbyggda variant *Z89*, finns nu även *cobol* och *Pascal*.

Heathkit tillverkar även en liten experimentdator, *ET3400*, för dem som vill lära sig mikrodatorns funktion från grunden. Till den finns en mycket omfattande dokumentation, lämplig för självstudier.

Datorer byggda med S-100 buss

En metod att bygga en dator är att sammanställa den av kretskort enl S-100 buss standard. Man skaffar en kortlåda med nätaggregat, eller bygger och förser den med lämpliga kort som finns färdigbyggda eller i byggsats. Ursprungligen kom S100 bussen till i *MITS* dator *Altair* som kom att starta hela "personal computing"-utvecklingen. Firman finns inte kvar, men bussen lever vidare och är idag en IEEE-standard. Den skiljer sig något från den ursprungliga bussen, något man måste ha i åtanke då man gör en kombination av kort och kortlåda. Från början var nämligen inte vissa stift standardiserade och de kom att användas till olika funktioner av olika kretskortstillverkare.

Exempel på S-100 buss datorer är *Cromemco*, *TEI*, *InterSystems* (tidigare *Itacha Intersystems Inc*), *No name computer* (!). Vi skall inte glömma två svenska konstruktioner. Den ena göra av *Frika elektronik* i Pajala och heter *Zampo 80*. Den andra tas fram av *Hobbydata*, Malmö som också säljer och importerar de nämnda datorerna. Andra märken är *Xitan*, *Horizon* och *Vector* med import av *Bita AB*, *Wernor AB* och *Dicro* dator för att nämna några av de många försäljningsställen av smådatorer som finns idag i Sverige.

En mängd lösa kort finns att välja mellan tillverkade av ovanstående firmor jämte ett antal andra: *Artech*, *Digital Research Corporation*, *Miniterm Associates*, *Dateli-*

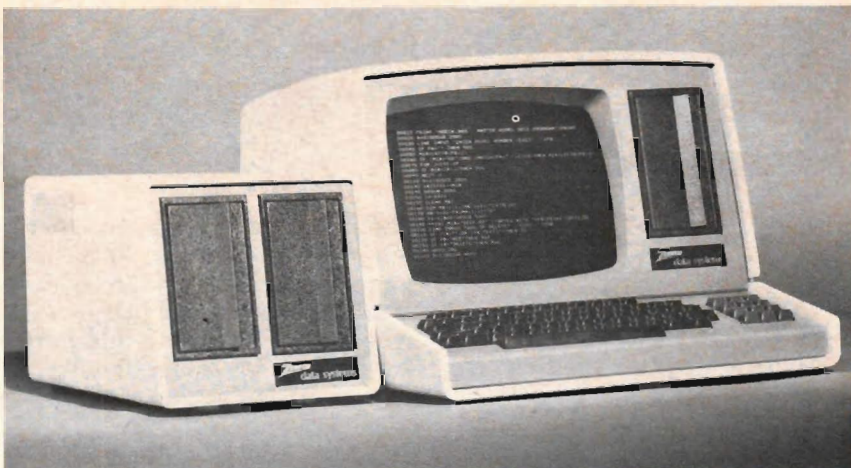


Fig 3:1. Heathkit H89/Z89 med dubbel flexskivenhet Z87.



Fig 3:2. InterSystems dator med S-100 buss.

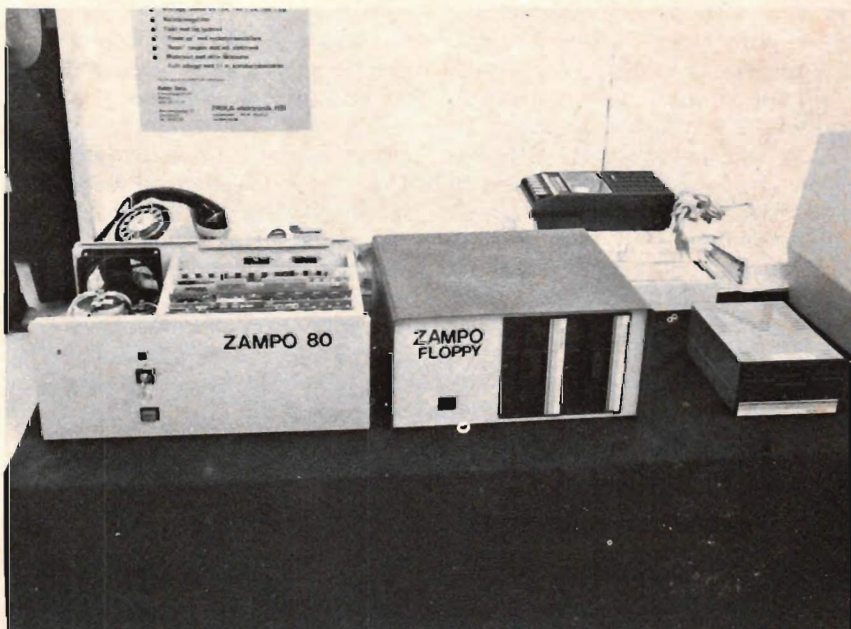


Fig 3:3. Zampo 80 S-100 buss dator från Frika Elektronik, Sverige.

ne, Vanderberg, Dynabyte och Thinker Toys (*Morrow's Micro Stuff*). En annan intressant grupp kort genererar musik. Här finns kort från *Solid State Music System*, *Newtech*, *Schertz computer music system*, *ALF system* och fler.

Datorkort för inbyggnad

En metod är att köpa en dator i form av ett kretskort som man sedan på lämpligt sätt bygger in i en låda och eventuellt kompletterar. Kortet kan vara färdigbyggt eller levereras i byggsats. Det finns några olika alternativ av det här slaget att välja mellan:

- **Telmac 1800** heter ett datorkort som några år har funnits på den svenska marknaden. Som processor används CPD 1802 från RCA, byggd i CMOS-teknik. Kortet ansluts till en vanlig tv och tangentbordet är i form av ett kretskort med beröringsavkänning. Programmen lagras på kassetband. Datorn var den första med *basic* till ett pris av ca 1700 kr med program när den kom. Ett språk, kallat CHIP-8, var till hjälp vid uppbyggnad av grafik på skärmen. Apparaten är på väg ut ur marknaden och en variant 2000 kommer som ersättning. Telmac säljs av **BHIAB Eletronics**, Norrtälje.

- En annan dator är **Oscom Nano**. Samma person ligger bakom konstruktionerna, finländaren *Osmo Kainolainen*. Konstruktionen skiljer sig från *Telmac 1800* i det att den är avsedd för 50 Hz PAL-s signaler till tv:n vilket möjliggör färggrafik för kommande expansion, en ny typ av kassettpassning vilken skall fungera bra även på billiga kassetbandspelare, tangentbord med riktiga tryckknappar och kortet är helt färdigbyggt. Oscom nano kan användas i kombination med en tv-mottagare. För att arbeta i *basic* måste man komplettera nano med ett tangentbord för ASCII-tecken och ett speciellt terminalkort för video eller ett för skrivare. Videokortet ger 64 tecken på 24 rader på skärmen och skrivkortet är avsett för 24 teckens bredd. Det ansluts till en *Olivetti-*

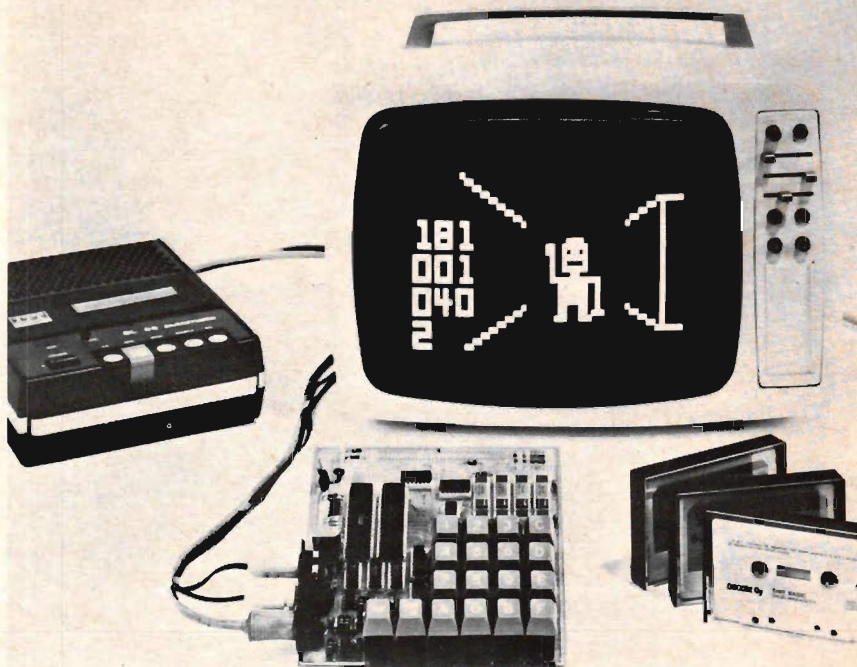


Fig 3:4. Oscom Nano är det lilla kortet i förgrunden, här anslutet till tv och bandspelare.

skrivare. Tre olika *basic*-dialekter kan väljas där den största är 12 k byte stor. Minnet kan givetvis expanderas och teckengeneratoren kan kompletteras för små bokstäver jämte matematiska tecken och grafiska symboler. Minnet i datorn kan expanderas i steg om 8 k och för videokortet finns minnesmoduler i storleken 12–16 k.

Programdistributionen sker i stor utsträckning via dataföreningen Cosmac Digital Group. Försäljningen av Oscom sker av **Teklab**, Norrtälje.

Ohio Superboard, som ingår i något som kallas Challenger 1 systemet, är ett komplett datorkort med tangentbord (ej Å, Ä och Ö), videoutgång och *basic* från **Microsoft** ligger i ROM och plats finns för upp till 8 k RAM på kortet. Datorn använder mikroprocessorn 6502.

Videosignalen har 60 Hz bildfrekvens och de flesta tv-monitorer

kan med enkel justering fås att fungera. Har man en tv-mottagare måste en tv-modulator kopplas till.

Basic-tolken räknar med $6\frac{1}{2}$ siffror och har talområdet 10^{-32} till 10^{32} . Vidare finns grafiska möjligheter. Texten på skärmen har 25 tecken per rad. Raderna, även de 25 till antalet, ligger dock så tätt att texterna går i varandra. Det finns dock både små och stora bokstäver. Som standard har datorn 4 kbyte RAM. Ytterligare 4 kbyte kan pluggas in på kortet. Ett expansionskort för 8 kbyte finns att tillgå och det inkluderar två kontrollkretsar för flexskivenheter med vardera 90 kbyte och med plats för totalt 24 kbyte RAM. Svensk representant är **Intersonic AB**, (Täby) som även säljer större system avsedda för framför allt kontorstillämpning: Challenger C3P, C4P och C8P system.



Fig 3:5. Compukit 101. Basic-dator på ett kort för anslutning till tv.

● **Compukit** heter en dator som starkt påminner om Ohio Superboard. Den tillverkas dock i England och säljs i byggsatsform, med svensk bygganvisning. Likheter med Superboard ligger i att processorn är av typ 6502, den har 8 k **Microsoft Basic** i ROM och ett ASCII-tangentbord. Förutom stora och små bokstäver har den grafik.

Den skiljer sig från Superboard i det att den är gjord för 50 Hz bildfrekvens, presentationen på skärmen sker på 48 kolumner och 16

rader, den har uhf-modul för direkt anslutning till tv-mottagare som ett alternativ till en monitor och nätdelen är stabiliserad med transformator. Med 4 k RAM kostar den 2 990 kr och med 8 k RAM 3 468 kr. För ett passande plasthölje får man betala 450 kr. Svensk representant **Marketing Trim AB**, Stockholm.

● **Rockwell AIM 65** är en enkortsdator med flera intressanta egenskaper. Den har ASCII-tangentbord, en rads skärm med 20 tecken och en liten skrivare för 20 teckens

bredd. Kortet kräver +5 V för halvledarna och +24 V för skrivaren. Monitorprogrammet, i ett PROM, är ovanligt stort: Hela 8 k. Här ligger förstås skrivarrutinerna, men också en disassembler och editorfunktioner. Vid maskinkodprogrammering behöver man inte mata in hexkod utan kan mata in mnemonics direkt, såsom i assemblerprogrammering. En riktig *assembler* finns att tillgå, 4 k stor, liksom en 8 k *basic*-tolk från Microsoft. Andra språk som finns är *PL65* (Liknande Intel's *PL/M*),

forth och pascal. Korten i Rockwells stora mikrodatorsystem passar. Man kan komplettera AIM 65 med t ex bubbelminnen vilket ger mycket stor kapacitet.

Dokumentationen till AIM 65 är föredömlig. User's guide omfattar 400 sidor och ger svar på det mesta. Den övriga litteraturen beskriver programmering, hårdvaran, dvs 6500-systemet, programlistning och köper man till ett basic-ROM ingår en bok som beskriver basic-språket. Kortet levereras med 1 kbyte RAM som kan expanderas upp till 4 kbyte. Utgångar finns för anslutning till kassettspelare och teletypskrivare. Ett video-interface kommer som tillbehör.

Svensk representant: **Betoma**, Solna.

Färdiga datorer

Färdiga datorer är ett annat alternativ. Priserna har rasat de senaste åren och de ligger inte så särskilt mycket högre än för byggsatser eller för helt hembyggda datorer i likhet med de i boken beskrivna. Det är dock bara halva sanningen. När man bygger ut de färdiga datorerna blir det som regel dyrt. Här tar man igen på gungan av vad man förlorar på karusellen. Flexskiveenheter, serie- och parallellportar kostar. Som vi sade tidigare bör man tänka sig för vad man vill göra med datorn. Är man enbart intresserad av programmeringen och har måttliga krav på framtida utbyggnader är en färdig dator det bästa alternativet. Låt oss se på något av vad som bjuds på den svenska marknaden:

■ **ABC 80** är en svenskbyggd Z80-baserad dator, framtagen i samarbete mellan **Luxor**, **Scandia Metric** och **Dataindustrier AB**. Sedan lanseringen hösten 1978 har den snabbt blivit en bestseller, mycket tack vare riklig svensk litteratur och programvara. Skriv och läsminnena (RAM) är dynamiska och omfattar 16 kbyte. *Basic*-tolken (interpretator) ligger i 16 kbyte läsminne (ROM). Den är ovanligt omfattande och snabb. Vid uträkningar kan man välja mellan heltal, flyttal och strängvariabler med

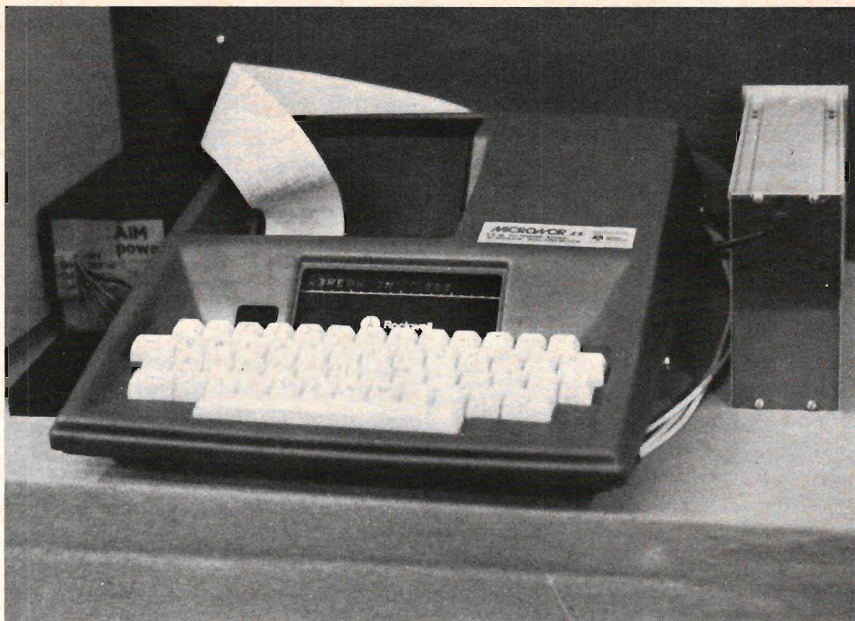


Fig 3:6. Rockwell AIM 65 dator på ett kort med en rads display och skrivare, här med kåpa.



Fig 3:7. Luxor ABC 80 – svenskbyggd och såld i stora antal.

ASCII-aritmetik. Stränguttrycken gör det möjligt att använda ordbehandling och ett särskilt program finns för detta (som dock inte kan prestera vad de specifika ordbehandlingssystemen kan, men här ligger också en stor prisskillnad). Ett intressant kommando utför rättelser i redan skrivna programrader. Grafik är en annan intressant egenskap. Därvidlag används view-data standard för de grafiska elementen.

I datorsystemet ingår en 12" bildskärm och en kassettbandspelare för lagring av data. En serieport enl V24 finns för anslutning till skrivare. Till en busskontakt på baksidan kan man ansluta expansionsenheter som flexskiveenheter, serie- och parallellportar, IEC-busskort, A/D-omvandlare. Bland periferiutrustningen finns fyra olika typer av skrivare plotter och digitaliseringsbord för överföring av grafiskt material till digital form.

Förutom ABC80-korten finns här ett kortsystem från Sattco, kal-

lat 4680 eller *databoard*, som tillåter vidlyftiga expansioner. Korten är främst avpassade för industrin vad gäller kvalitet och prisnivå. En tjock applikationsbok i A4 format visar ett stort antal exempel på expansion av en ABC-80 dator med 4680-korten. För applikationerna beskrivs såväl kopplingsscheman som program.

Programutbudet är svenskt och omfattande.

■ **Acorn Atom** heter en engelsk dator som nu kommit till Sverige. Till dess förtjänster hör ett lågt pris, den har en speciell *basic* med möjlighet att blanda *basic* och assembler, grafiska möjligheter med 256×194 punkters upplösning och den kan dra linjer mellan två punkter på skärmen med *basic*-kommandon.

Datorn är byggd på ett kort, men genom en yttre expansionslåda kan man använda övriga kort i Acorn-sortimentet som finns sedan tidigare.

Datorn är tänkt att kunna användas i undervisning. Den kan då

"klasskopplas" på så sätt att maskinernas bussar kopplas samman. Därigenom kan man utnyttja en gemensam flexskivenhet. Varje station har då sin egen kod.

Den har hela 32 bitars heltalsbasic (16 bitar är vanligt i sammanhanget) och matematiska rutiner med flytande decimalkomma och 10 siffrors upplösning.

Datorns RAM är bara 2 kbyte stort i standardutförande, men det kan byggas ut i 1 kbyte steg upp till 1 kbyte. Den senare minnesstorleken behövs för att rymma grafik med hög upplösning. Två ytterligare ROM finns det plats för på kretskortet. Kretsarna kan utgöras av "Acorn floating point scientific ROM" och ytterligare ett ROM vilka tillsammans har 16 kbyte kapacitet. En extra modul medger färgåtergivning antingen med monitor eller tv-mottagare. Processorn är av typ 6502. Datorn drivs med 8 V 800 mA vid minsta systemlösningen. Då utnyttjas en intern stabilisator. Det maximalt utbyggda systemet kräver 5V och 1,8 A från yttre



Fig 3:8. Acorn Atom heter en prisbillig engelsk dator.



Fig 3:9. *Apple II har här fått "stordatorkapacitet" med anslutning till winchesterminne (40 Mbyte!), flexskivenhet och skrivare.*

stabilisator. Datorn importeras av **Marketing Trim**, Sthlm. Förs. **Beckman Innovation AB**, Sthlm.

■ **Apple** är en av de största på den amerikanska marknaden. Dator och tangentbord ligger i samma hölje. Bildskärm ingår ej. För europeiska tv-mottagare med 50 Hz bildfrekvens och PAL färgsystem måste man plugga in ett extra kretskort som ger upp till 16 olika färger. Som standard har den flytalsbasic med 9 siffror, operativsystem och disassembler i ett 12 kbyte ROM, men man kan från flexskiva ladda in en heltalsbasic.

Grafiken är ovanligt högupplösande med 280×192 inkrement. Då kan man använda 6 färger. Upplösningen vid 16 färger är 40×48 punkter.

Enheten är liten och lätt tack vare switchat nätaggregat och plats finns för 8 extrakort. De kan utgöras av A/D-omvandlare, D/A-omvandlare, IEEE-buss, serie och parallellanslutningskort, aritmetisk processor, realtidsklocka m m. Som standard har datorn 3 TTL-in-

gångar, 4 TTL-utgångar och 4 analoga ingångar (8 bitars upplösning).

Programmeringsspråken till Apple är många. Förutom nämnda två basic-tolkar finns *fortran* och *pascal*. *Visical* heter en ny typ av användarprogram för problemlösning som verkar mycket användbart. APL kommer i framtida maskiner. Ett mycket stort antal amerikanska användarprogram finns att tillgå. Apple har gjort en del, men de övriga kommer från firmor som specialiserat sig på program. Alla är dock inte tillämpliga för svensk marknad, men det finns idag en rad program för svenska företagare med bl a bokföring, reskontra, löner, fakturor. Programmen är utvecklade i Sverige.

Spel med färg och grafik samt röst- och musiksyntetisering hör till paradnumren och enheter för röstgenerering och röstigenkänning finns även.

En speciell finess är ett litet kort med Z80-processor som tillåter att man använder programvara från **Microsoft**. Diskoperativsystemet

är då CP/M till vilket finns stora mängder programvara. Bland annat förekommer Cobol.

Datorn kan faktiskt förvandlas till en "stordator" genom att anslutas till ett skivminne av typ Winchester med 4×10 Mbytes kapacitet! Det kan fungera som massminne i ett system där 64 Apple II-datorer är sammankopplade. En dator krävs för själva hanteringen av data.

Ett annat tillbehör är ett grafiskt digitaliseringsbord på vilket man kan rita upp sina figurer för vidare datorbehandling. Bl a kan ritningarna storleksförändras och göras i 6 olika färger. Presentationen kan sedan ske med t ex en **Epson**-skrivare.

Prismässigt ligger Apple II något över sina närmaste konkurrenter ABC80 och PET 3008 med tanke på att bildskärm inte ingår, men det gäller grundversionen. Minnesutbyggnaden i Apple II är dock ovanligt billig så totalkostnaden för ett större system blir i klass med de övriga eller kanske t o m billigare

beroende på utbyggnad.

En större dator, *Apple III*, tillverkas nu även. Med bl a 128 kbyte stort minne är den främst avsedd för industri- och kontorstillämpningar. Svensk representant: **Systemelektronik AB**, Stockholm.

■ **Exidy Sorcerer** är en amerikansk dator, sammanbyggd med ett tangentbord för anslutning till videomonitor eller tv med en modulator mellan enheterna. Presentationen på skärmen sker med 30 rader och 64 kolumner. Vidare finns 64 grafiska tecken med vars hjälp man själv kan bygga upp bilder. Alternativt kan man definiera 128 grafiska element. Å, ä och ö finns inte men man kan själv programmässigt definiera valfria tecken varvid frågan löses. De 128 ASCII-tecknen omfattar såväl stora som små tecken.

Processor är Z 80 och basic-tolken är skriven av **Microsoft**. Den ligger i en kassett som innehåller ett ROM. Kassetten kan bytas ut så att man får andra språk som assembler och APL. En ordbehandlingskassett finns även.

På baksidan av datorn finns en kortkontakt enligt S-100 buss standard. Datorn kan alltså expanderas med de många moduler som finns att tillgå i S-100 buss standarden. Vidare finns en RS232 serieport in/ut för 300 eller 1200 bauds överföringshastighet. En parallellport in/ut för 8 bitar med handskakning finns även. RAM är 8 kbyte stort som standard men kan expanderas upp till 32 kbyte. Svensk representant: **Hobbydata**, Malmö.

■ **PET Commodore 2001** var den första färdigbyggda, kommersiellt tillgängliga hemdatorn. I USA kostade den 1000 dollar, dvs ungefär 5000 kr och blev snabbt en succé. I Sverige fick vi den inte lika entusiastiskt mottagande då den med moms kom att kosta över 12 000 kr. Idag är prisbilden en annan och den ligger prismässigt ungefär som sina konkurrenter. Den första modellen, 2001, innehöll såväl bildskärm, tangentbord som kassettbandspelare. Ett bättre alternativ är dock nya 3008 som kostar lika mycket, utan bandspelare, men

med ett betydligt bättre tangentbord. Det minimala tangentbord som den första typen hade var just dess egentliga nackdel. Nu när datorn har ett rejält tangentbord har PET Commodore blivit något att räkna med inte bara för amatörer utan även för professionella användare. Idag satsar man främst på dem med ett massivt stöd av svenskframtagna programvara för bl a företagare.

En intressant detalj är att PET-datorerna har IEEE-buss för direkt anslutning till mätinstrument med detta bussuttag. Industrin använder bussen alltmer och man ser ofta en PET ansluten till automatiska mätsystem eller andra övervakningssystem.

3008 heter alltså den minsta datorn i den nya serien. Den har 8 k RAM. Modellerna 3016 och 3032 har följaktligen 16 k och 32 k stora RAM. ROM är 14 k byte stort vilket omfattar 8 k basic, 4 k diskoperativsystem och 1 k byte diagnostiska rutiner.

Basic-tolken har heltalsaritmetik med flerdimensionella matriser, matematiska standardfunktioner, fullständig stränghantering, möjlighet till subrutiner i maskinspråk, klocka, slumtalsgenerator och filhantering via basic.

Som tillbehör finns dubbelt flexskiveminne med 2×170 kbytes kapacitet, kassettstation, skrivare med friktions- eller traktormatning, utbyggnadsminne upp till 32 k, plotter, tv-anpassning, ljuspenna, teckengenerator för svenska tecken m m.

Svensk representant: **Datatronic AB**, Stockholm.

■ **Sharp** var den japanska firma som först började sälja smådatorer i Sverige. Först kom MZ-80K som är en basic-dator integrerad med tangentbord, bildskärm och kassettspelare. Processor är Z80. Som standard har maskinen 20 k RAM i vilket basictolken laddas från band eller flexskiva. Det senare rekommenderas. Att ladda in en basictolk från band varje gång maskinen startas kan vara enerverande i längden, även om överföringshastigheten är hela 1200 baud. Små och stora tecken finns liksom grafiska symboler. Minnet kan byggas ut till 48 k byte. Bland övriga egenskaper kan nämnas klocka och musikfunktion. Förutom basic finns assembler och maskinspråk.

Den andra datorn från sharp är mera revolutionerande. Det är den första fick-datorn för basic på marknaden. Med storlek som en räknedosa har den givetvis sina be-



Fig 3:10. Den första fickdatorn: PC1211 från Sharp. Språket är basic.

gränsningar men dess basic omfattar bl a trigonometriska funktioner, logaritmer m fl matematiska funktioner. Presentationen sker på en flytande kristallindikator med en rad och maximalt alfanumeriska tecken. Dataprogrammen kan lagras i en kassettbandspelare, men det är den enda "expansion" som datorn tillåter. Inte bara storleken påminner om en räknedosas utan även priset. Svensk representant: **Addo försäljnings AB**, Åtvidaberg.

■ **Sinclair ZX80** är en intressant lågprisdator. Den kostar en fjärdedel av de vanliga persondatorerna, den arbetar i basic och kan anslutas till en tv-mottagare och kassettspelare. Datorn har 4 kbyte ROM i vilken en liten basic ligger och vidare finns ett 1 kbyte stort RAM. Processor är Z80. Den är liten till formatet. Tangentbordet är helt stumt och utfört som ett kretskort med beröringsomkoppling, något som kanske inte tilltalar alla, men naturligtvis en eftergift för att hålla priset nere.

Datorn räknar bara med heltal med talområdet $\pm 32\,767 (\pm 1/2 \cdot 2^{16})$. Det är naturligtvis en klar begränsning liksom det faktum att trigonometriska och exponentiella funktioner saknas. Vissa grafiska möjligheter har den och skärmen rymmer 23 rader med 32 tecken.

Datorn har alltså tydliga begränsningar i sin basic, ett litet användarminne och skärmen är släckt när någon operation försiggår. Däremot har den goda redigeringsmöjligheter vid programskrivning och den är som sagt mycket billig. Svensk representant: **Beckman Innovation AB**, Stockholm.

■ **SWTPC** hör till de äldsta på hobbydatorsidan. Deras första dator kom ut inte långt efter det att Altair-datorn debuterade, men till skillnad från den satsade SWTPC på en egen buss kallad SS-50 och Motorolas 6800-processor. Bussen har man behållit, men idag tillverkas enbart datorer med 6809-processor. I Sverige säljs de bl a med TSC:s programvara med operativsystemet *Flex*. Systemprogram finns för textbehandling, basic, assembler, avlusning, sortering, ba-



Fig 3:11. Radio shack TRS 80 tillverkas av Tandy, USA.

sic, extended basic, multiuser basic. Pascal och fortran är i antågande när boken skrivs.

Datorerna finns i tre serier. 69/A med 8 kbyte RAM, 69/56 med 56 kbyte RAM och S/09 med 128 kbyte RAM utbyggbart till 768 kbyte! Som kringutrustning finns matris-skrivare, skönskrivare, 5 1/4 och 8 tums flexskivenheter samt 16 Mbyte hårddisk. Svensk representant: **Swedish Electronics hb**, Uppsala.

■ **TRS 80** från **Radio Shack** är en av de största och första på den amerikanska marknaden. Märkligt nog har introduktionen på den svenska marknaden kraftigt fördröjts, men nu skall det bli ändring. **Nordisk Micro**, Stockholm är ett företag i Johnson-koncernen som skall ta hand om försäljningen och apparaterna skall enligt utsago levereras

för fullt när den här boken kommer ut.

TRS 80 är en dator med processor Z80. Den kommer i flera versioner. Den enklaste modellen får inte å, ä och ö vilket skall finnas i de dyrare modellerna. Minnet (RAM) är 4 k byte stort. Datorsystemet består av enheterna tangentbord, bildskärm och kassettspelare. Presentationen på skärmen sker med 64 teckens bredd och 16 rader. Grafik är även möjlig. Basic-tolken ligger i 4 k ROM (Level 1). Den något större modellen har en 16 k Basic, med betydligt fler instruktioner och kommandon än sin mindre kollega. Minnet (RAM) omfattar 16 k byte. Den kan även fås med en 5" flexskivenhet vars skivor rymmer 200 k byte eller av 8" storlek. En Winchester-disk kommer även. ■

Mikrodatorns talsystem

AV GUNNAR LILLIESKÖLD

En mikrodator arbetar med binärkod. För att underlätta programmeringen skriver vi datorns koder med hexadecimala eller oktala tal.

Vad binärkod, hexkod och oktalkod är framgår i det här avsnittet. Lär dig begreppen innan du tränger vidare in i mikrodatorns funktion!

■ Det talsystem vi använder i dagligt bruk, det decimala systemet, är inte användbart i datorer, åtminstone inte i dagens datorer. Orsaken till detta är att varje siffra i ett tal då måste representeras av 10 nivåer, 0 till 9. Om vi i stället väljer det binära talsystemet med 2 som bas, behöver vi endast en hög och en låg nivå, vilket är lätt att utföra praktiskt. För att kunna omvandla decimala tal till binära och tvärt om, måste vi känna till hur resp talsystem är uppbyggda.

I korta drag kan detta förklaras enligt följande:

Antag att vi har decimaltalet 134. I detta system är den sista siffran värd fyra ental eller $4 \cdot 1$; i andra positionen är siffran värd tre tiotal, d v s $3 \cdot 10$ och i tredje positionen (första siffran) ett hundratal eller $1 \cdot 100$.

Man kan också skriva talen som $1 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$. (Observera att $10^0 = 1$). Rent allmänt kan ett decimaltal skrivas som:

$$D = a_{n-1} \cdot 10^{n-1} + a_{n-2} \cdot 10^{n-2} + \dots + a_1 \cdot 10^1 + a_0 \cdot 10^0$$

Koefficienten a kan därvid väljas ur decimalsystemets 10 siffror 0 till 9. Basen i systemet är 10.

Våra talsystem är positionssystem

Som vi såg är decimalsystemet ett positionssystem, där varje steg till vänster innebär en multiplikation med basen ($1, 10 \cdot 1, 10 \cdot 10, 10 \cdot 10 \cdot 10$ osv eller $10^0, 10^1, 10^2$ osv). Detta gäller även de binära, oktala och hexadecimala talsystemen.

Det binära talsystemet är, som nämnts, antaget i digitala tillämpningar. Generellt för detta gäller att basen är 2 och att det innehåller två symboler; 1 och 0.

Inom digitaltekniken talar man om logisk etta eller logisk nolla som oftast representeras av spänning eller icke spänning, ström eller icke ström, ljus eller icke ljus (i ett ljusöverföringssystem).

Ibland är förhållandena tvärt om d v s spänning kan representera en nolla och "icke" spänning en etta. Huvudsaken är att man klart har definierat vad som är vad.

Det tal vi tidigare nämnde, 134, heter i binärsystemet (= tvåsystemet) 10000110. Hur kommer man då fram till detta?

Låt oss se på det binära systemets grunduppbyggnad. Det kan allmänt skrivas som:

$$B = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

Sätter vi in vårt tal i formeln, blir resultatet:

$$1 \cdot 128 + 0 \cdot 64 + 0 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1$$

(En kontroll ger $128 + 4 + 2 = 134$).

Från decimalt till binärt tal

Omvandlingen görs enklast genom att man provar med att subtrahera största möjliga potens av 2 och skriver en etta i motsvarande position, om subtraktionen går att utföra utan att resultatet blir negativt. Skriv för övrigt en nolla. Det betyder i detta exempel att vi provar först med 128, som går att subtrahera. Därför sätter vi 1 i binärtalets första position. Rest blir 134 - 128 = 6.

Vi provar därefter med nästa tal nedåt i serien, nämligen 64. Detta ger 0.

Nästa steg är att pröva med 32. Det ger 0.

Vi provar sedan med 16, vilket ger 0.

Inte heller subtraktion med 8 går. Vi skriver 0.

Nästa tal i serien, 4, kan subtraheras från 6. Vi skriver 1. Rest $6 - 4 = 2$.

Resten 2 kan subtraheras med nästa tal i serien, nämligen 2. Vi skriver 1. Rest 0.

Resten 0 subtraheras med det minsta talet i serien, $2^0 = 1$. Det går ej att utföra och vi skriver 0.

Sammanställer vi de givna talen får vi 10000110, vilket också var det tal vi sökte.

Omvandling genom division

Det finns ytterligare en metod för omvandling från decimalt till binärt tal: Dela kontinuerligt med 2 och notera resterna. Den första divisionen ger den minst signifikanta siffran.

Låt oss åter ta exemplet 134.

	Kvot	Rest
134/2 =	67	0
67/2 =	33	1
33/2 =	16	1
16/2 =	8	0
8/2 =	4	0
4/2 =	2	0
2/2 =	1	0
1/2 =	0	1

Vi får även här resultatet 10000110.

Omvandling av tal mindre än ett

Tal mindre än ett kan naturligtvis även förekomma i binär form. Tag som exempel det binära talet 0,101. Det betyder $1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$ eller $1 \cdot 1/2 + 0 \cdot 1/4 + 1 \cdot 1/8$. Detta ger alltså $0,5 + 0,125 = 0,625$.

Går vi åt andra hållet, d v s om vi har talet 0,625 (decimalt) och vill omvandla det till den binära motsvarigheten förfar vi enligt följande:

Vi börjar med att multiplicera med 2; $0,625 \cdot 2 = 1,250$. Ettan till vänster tar vi tillvara, och den kommer då att stå närmast till hö-

ger om kommat i det binära talet. Resten av talet, d v s 0,250, multiplicerar vi åter med 2 och får 0,500. Nollan sätter vi till höger om vår etta och multiplicerar sedan 0,500 med 2, och får 1,000. Vi sparar ettan och sätter den till höger om nollan i vårt binärtal. Eftersom vi inte kan multiplicera med 0,000 är operationen avslutad.

Sammanställer vi delresultaten får vi 0,101.

Binära tal otympliga Hexadecimala tal bättre

Datorer arbetar som sagt med binära tal, och skall vi programmera med dessa blir det hela ganska svårhanterligt. Därför har man i många datasystem hexadecimal inmatning. I vårt tidigare exempel hade vi decimaltalet 134. Det heter alltså 10000110 i den binära versionen.

Den hexadecimala versionen lyder 86. För att inte denna skall förväxlas med ett decimalt tal brukar man skriva 86₁₆, H'86 eller \$86. Genom att hexadecimalsystemet innehåller 16 symboler, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E och F, blir det fråga om korta tal, kortare än i decimalsystemet. En fördel framför detta är att hexadecimala tal enkelt kan omvandlas till binära tal och tvärt om.

Låt oss ta vårt tidigare exempel, talet 1000 0110. Vi grupperar talet i grupper om fyra siffror. Varje 4-grupp motsvarar ett hexadecimalt tal. Talet 1000 motsvaras av 8 och 0110 av 6.

Vårt hexadecimala tal heter alltså 86. Det bör vi läsa som "åtta sex, hexadecimalt". Läs ej "ättiosex", det kan ge upphov till förväxlingar med decimala tal. För säkerhets skull skriver vi \$86 (86₁₆ eller H'86).

Tabell 1, sid 141, visar de olika talsystemen i jämförelse. Vi ser här också att det oktala systemet här har 8 som bas. Lägg märke till att 10 i oktalsystemet betyder 9 i decimalsystemet och att 10 i hexadecimalsystemet betyder 16 i decimalsystemet, \$20 betyder 32₁₀ osv.

Decimal kodning i sextonsystemet

Avkodar vi 4 bitar, får vi alltså en hexadecimal siffra från 0 till F. Ibland använder man avkodning enligt BCD (Binary Coded Decimal). Det innebär, att 4 bitar avkodas, men man tar då bara tillvara siffrorna 0–9. Resten, d v s A, B, C, D, E och F används icke. Med 1 byte (8 bitar) kan man alltså representera decimala tal från 00–99. Man förlorar som synes ganska mycket, men in- och utmatning av tal föranklas.

Addering enkelt förfarande Subtrahering mera speciellt

Addering av binära tal sker ungefär som för decimala tal. Låt oss ta ett exempel: De decimala talen 9 och 1 skall adderas, d v s binärtalen 1001 och 0001. Vi ställer upp dem enligt följande:

$$\begin{array}{r} 1 \\ 1001 \\ + 0001 \\ \hline 1010 \end{array}$$

1 och 1 är 10, därför skriver vi 0 i den högra kolumnen och får 1 som minnessiffra.

Ett ytterligare exempel visar att det inte är svårt att addera binära tal:

Decimaltalen 9, 1 och 3 skall adderas. Vi skriver:

$$\begin{array}{r} 1 \\ 1001 \\ 0001 \\ + 0011 \\ \hline 1101 \end{array}$$

Subtraktionen är i och för sig inte heller svår att genomföra:

$$\begin{array}{r} 1001 \\ - 0001 \\ \hline 1000 \end{array}$$

Det är bara en nackdel med detta förfarande, vårt datorsystem kan inte lagra eller behandla minus-tecknet. I stället tillämpar man principen komplementära tal.

Principen innebär i stort att man adderar två tal där det ena talet är negativt (komplementärt).

Med tre bitar kan vi presentera åtta olika binära tal:

$$\begin{array}{l} 000 = 0 \\ 001 = 1 \\ 010 = 2 \\ 011 = 3 \\ 100 = 4 \\ 101 = 5 \\ 110 = 6 \\ 111 = 7 \end{array}$$

En annan variant är att lägga talet kring 0 och låta dessa vara positiva eller negativa:

$$\begin{array}{l} 100 = -4 \\ 101 = -3 \\ 110 = -2 \\ 111 = -1 \\ 000 = 0 \\ 001 = 1 \\ 010 = 2 \\ 011 = 3 \end{array}$$

Den första biten i detta tal kallas teckenbit. Denna är 0 om talet är positivt och 1 om talet är negativt.

Omvandlingen av positiva tal till negativa är enkel att utföra. Man byter ettor mot nollor och nollor mot ettor och får då något som kallas ettkomplement. Därefter lägger man till en etta i den minsta signifikanta siffran (d v s den siffran som ligger längst till höger – den minst värda siffran). Vi får då ett tal som kallas tvåkomplement. Det är detta vi skall använda i vår dator.

När vi adderar två tal, t ex 011 (3₁₀) och 111 (–1₁₀) kommer vi ju att få 1010 som svar. Eftersom vi arbetar med bara tre bitar i exemplet, tar vi oss friheten att "kasta bort" den uppkomna ettan.

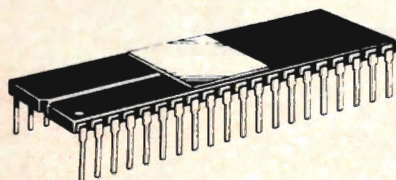
Addition av två BCD-tal kan inte utan vidare göras. Vi skall inte gå in på metodiken för detta, utan i stället konstatera att det sköter vår dator (6800) genom en speciell instruktion som heter DAA (= decimal justering).

Vi skall inte fördjupa oss mer i talsystemen, även om mycket mer finns att nämna. Viktigt är att vi lär oss **grunderna** för talsystemen, mikrodatorns funktion och litet om dess programmering. Nästa steg blir att bygga vårt utvecklingssystem och använda det. Då lär vi oss snart hur det hela fungerar – och då är vi mogna att gå vidare. ■

Hur fungerar mikrodatorn?

AV GUNNAR LILLIESKÖLD

Vi skall här försöka att ge svar på frågan om hur en mikrodator fungerar med en förenklad bild. Först bör man dock studera avsnittet tal-system.



■ Man talar oftast om en *mikrodator* som en liten krets, men det rätta är att denna utgör ett helt system. I det finns sedan en mikroprocessor, som för det mesta rymmer i en integrerad krets. Det finns dock

kretsar som inte bara innehåller en mikroprocessor utan faktiskt rymmer en komplett mikrodator! Hjärtat i mikroprocessorn är kontrollenheten. För att förklara dess funktion kan vi studera en analogi i form av en teckning, *fig. 1*.

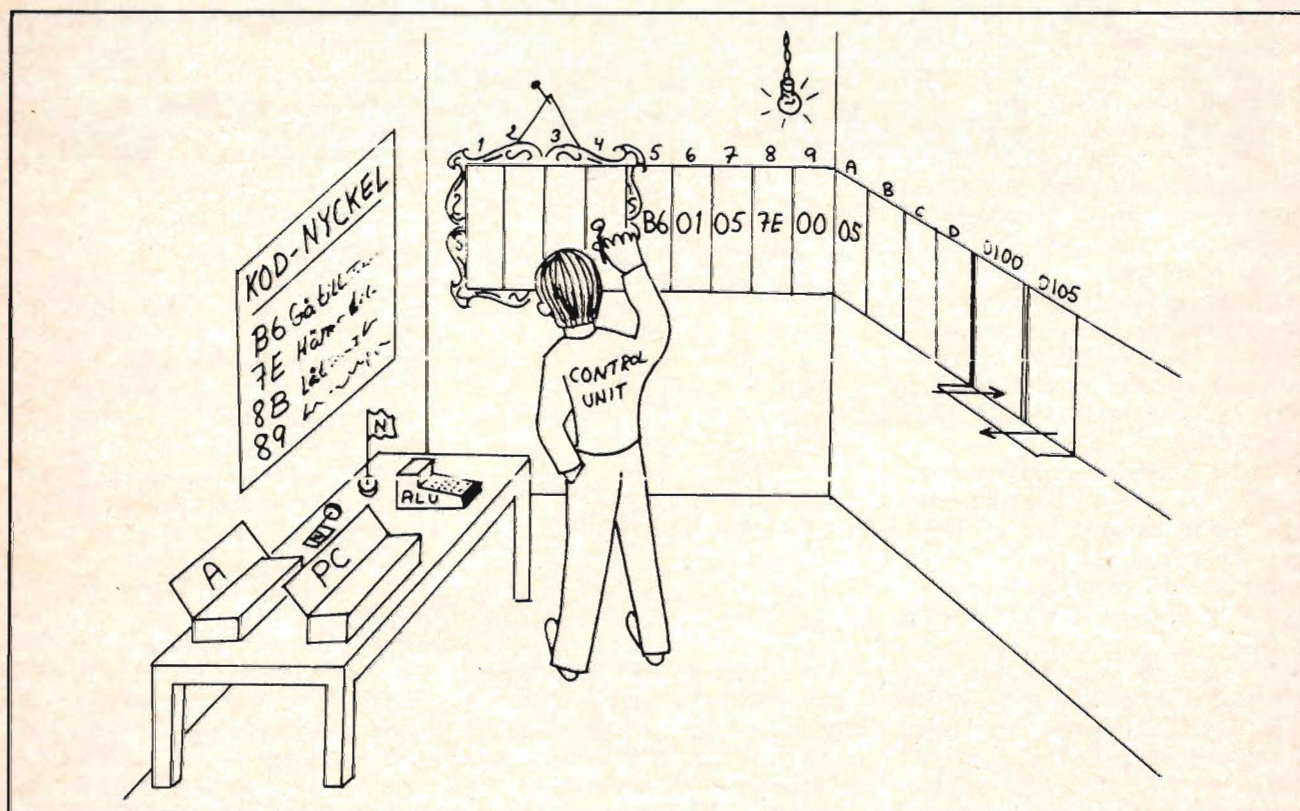
Mannen på bilden, "Control unit", hämtar resp lämnar data i in- och utluckorna till höger på bilden. Data består av en instruktion, en kod, som talar om vad han skall göra. Där står t ex B6. Vad det betyder, finner han på väggen i kodnyckeln. Där står att han i låda A (Ackumulatoren) skall lägga den adress som framgår av följande två bytes (=ord) som han hämtar i inluckan. För varje instruktion han utför, noterar han detta i programräknaren, d v s i den låda som är utmärkt PC.

När lampan i taket blinkar skall han börja om från början (nollställning). Början sker från en bestämd adress, från vilken han räknar upp med ett varje gång en instruktion

utförs och noterar detta i PC (Programräknaren).

På väggen hänger ett antal tavlor som är numrerade från 0 till D. Detta representerar adresser för olika enheter. Den "svarta tavlan" med adresserna 1 till 4 kan man skriva data på och radera. I verkligheten representerar detta ett skriv/läsminne eller RAM (Random Access Memory). Till höger om denna finns ett antal numrerade tavlor, där man bara kan läsa vad som står. Detta representerar alltså ett ROM (Read Only Memory). Jämför gärna med *fig 3* som visar ett mikrodatorsystem av minsta tänkbara storlek i 6800-serien. Vi återkommer till detta senare och återgår till att betrakta *fig 1*.

Kontrollmannen har som synes ytterligare några redskap till sitt förfogande. På skrivbordet finner vi en räknedosa, märkt ALU, som innebär aritmetisk, logisk enhet. Med denna kan vi utföra olika operationer som addition av tal, OCH,



*Fig 5:1. Vad vi ser på bilden är faktiskt en förenklad bild av ett mikrodatorsystem! Se texten och jämför med *fig 3*. Illustrationen är hämtad ur ett kursmaterial ursprungligen sammanställt av Bo Andersson, f:a Inometer, för Rifa.*

ELLER, exklusiv ELLER samt vänster- och högerskift av olika slag. Här finns också två flaggor som hissas för vissa speciella händelser: Z-flaggan hissas om innehållet i A blir noll och N-flaggan hissas om A blir negativt. (I verkligheten finns fem "flaggor" i processorkretsen 6800.)

Från analogi till blockschema

Låt oss efter denna bild av hur det generellt går till i mikroprocessorn se på dess blockschema, fig. 2.

Vårt program kan se ut enligt följande:

Adress	Minnesinnehåll
123B	86
123C	10
123D	7E
123E	12
123F	3B

Förloppet börjar med att kontrollenheten sänder ut signalen "Läs", varvid 86 kommer in i

CPU:ns informationsregister, IR, via databussen. Data kommer in i instruktionsregistret genom att CU aktiverar strömbrytare 1.

Information sparas alltså i instruktionsregistret och styr CU:s fortsatta verksamhet. (Koden 86 betyder att nästa ord skall lagras i ackumulatören, A.)

CU följer instruktionen och ökar programräknaren, PC, med ett steg via strömbrytare 2. (De i schemat utmärkta strömbrytarna styrs alla av CU. Förfarandet kallar vi mikrooperationer.) Strömbrytare 3 sluts sedan och för över programräknarens innehåll till adressregistret, AR.

Nu ligger en ny adress ute. Adressen, som i exemplet är 123C, pekar på en minnescell i ett ROM (eller PROM) som innehåller informationen 10. CU sänder ut signalen "Läs" och därmed kommer 10 in i mikroprocessorn via databussen.

Föregående adress innehöll en

instruktion som innebär att CU skulle lägga nästa inkommande data i ackumulatören. Den kod vi nu får in, 10, läggs därför i ackumulatören genom att strömbrytare 4 sluts. CU ökar PC med 1 via strömbrytare 2. Med strömbrytare 3 slussas innehållet ut i adressregistret, AR. Därmed är första instruktionen klar.

Nästa adress som ligger på adressbussen är 123D. När CU sänder ut "Läs", kommer 7E att ligga på databussen och denna information slussas till IR över strömbrytare 1. CU tolkar denna kod som betyder hoppa till den adress som nästföljande två ord (bytes) anger. PC stegas fram ett steg (strömbrytare 2). Innehållet i PC läggs i AR genom strömbrytare 3.

CU ger därefter signalen "Läs" (d v s läs i den adress som adressregistret anger, d v s 123E). På databussen kommer att ligga 12. Vår instruktion 7E innebär att nästföljande två ord skulle överföras till

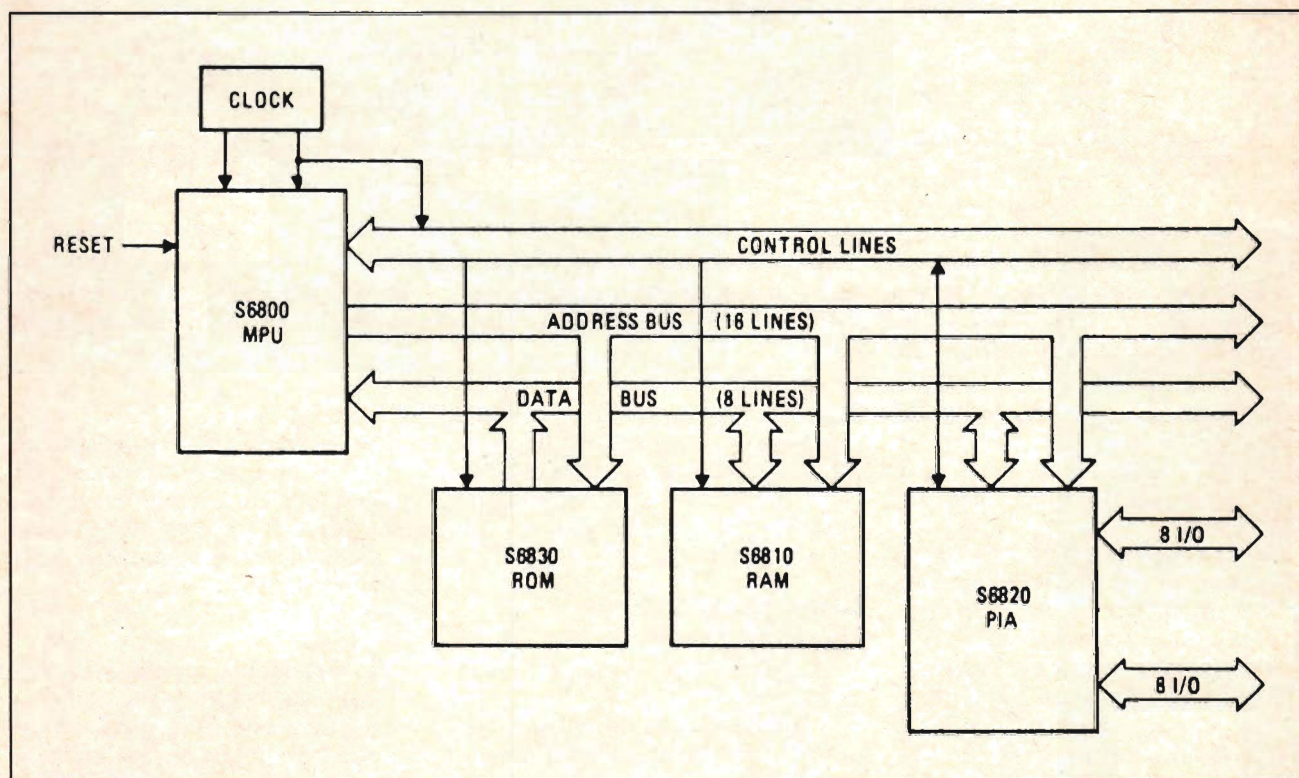


Fig 5:2. Här visas ett blockschema för minsta möjliga mikrodator med processorn 6800. Ännu enklare konfigurationer ger 6802 och 6801 med två resp en krets för en komplett mikrodator. Blockschemat gäller även för 6809, med den skillnad att klockan är inkluderad i mikroprocessorkretsen.

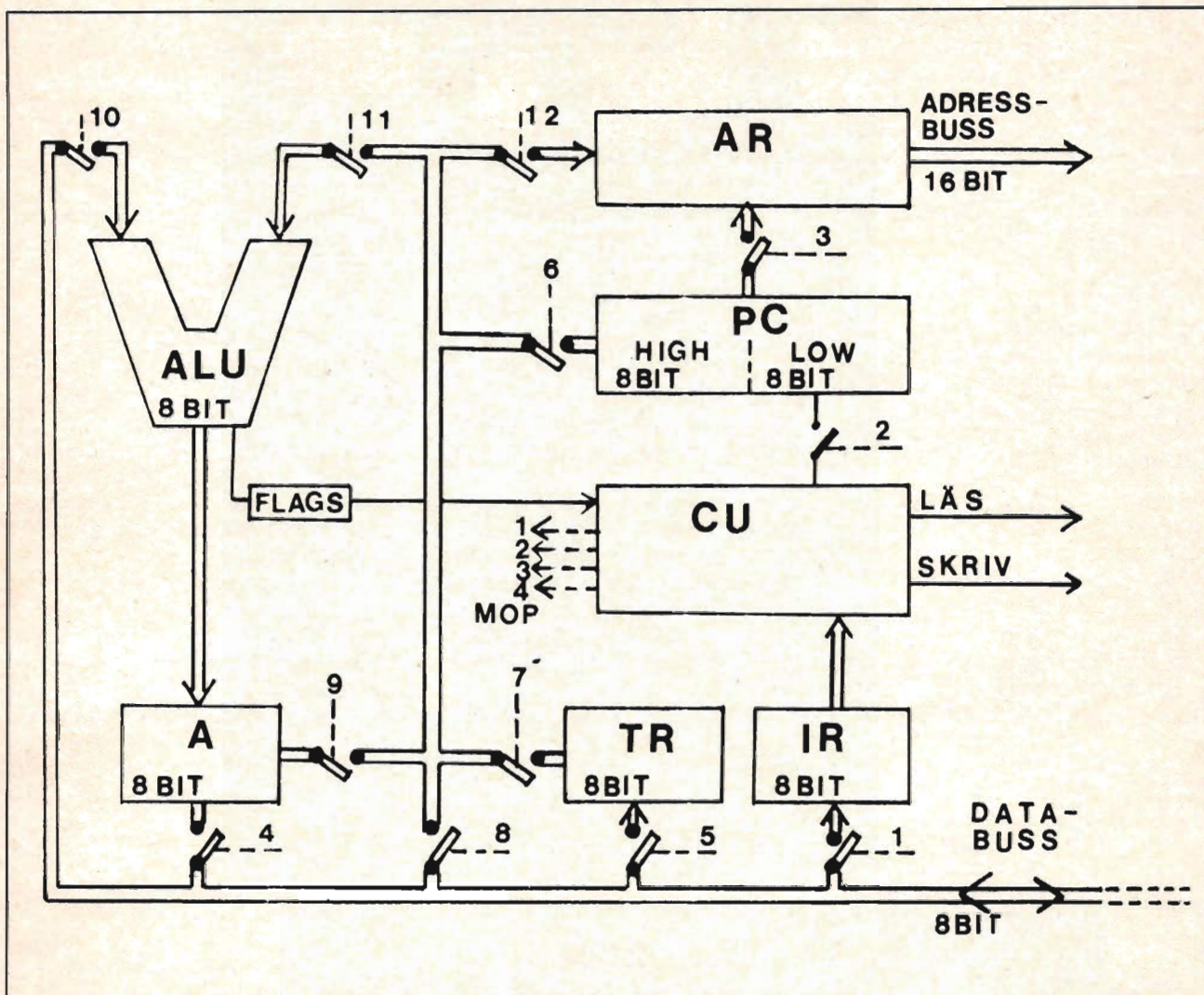


Fig 5:3. En förenklad bild av uppbyggnaden hos processorkretsen 6800: IR = Instruktionsregister · PC = Programräknare · ALU = Aritmetisk och logisk enhet · AR = Adressregister · A = Ackumulator · CU = Control unit · MOP = Mikrooperation · TR = Temporärt register.

adressregistret, eftersom vi skulle hoppa till denna adress. Det gör vi inte direkt, utan lagrar det första ordet först i ett temporärt register, TR.

Databussens innehåll, 12, slussas alltså till TR via strömbrytare 5. PC stegas fram ett steg och dess information överförs till AR.

Adressregistret står nu i position 123F och vi hämtar in nästa data genom att CU sänder ut "Läs" och sluter strömbrytarna 8 och 6, och överför data till PC. Här står nu 12, vilket skall utgöra de mest signifikanta talen i den nya adressen. CU ger därefter order till strömbrytar-

na 6 och 7, så att de två minst signifikanta bitarna, 3B, överförs till PC. Innehållet i PC är nu 123B. Via strömbrytare 3 överförs så innehållet till adressregistret. Instruksen är klar.

Långa program med maskinkod

Hela programmet kommer alltså att bestå av:

- Ladda ackumulatorn med 10
- Hoppa tillbaka till första adressen

Som framgår av tabellen ovan blir program skrivna i maskinkod ganska långa, vårt enkla program

krävde fem rader.

För att få kortare program och för att lättare hitta i program för felsökning och komplettering kan man, om maskinen tillåter, använda assemblerprogrammering. Vårt program skulle då lyda:

```
LDA A $10
JMP 123B
```

För att kunna arbeta med Assembler eller andra språk som Basic etc måste vi i vår dator ha ett omvandlingsprogram, en interpreter. Detta kan ligga i ett ROM, PROM eller matas in i RAM från kassettspelare, remsläsare e dyl. I vårt blockschema, fig 2, förekom-

MSB \ LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*	NOP (INH)	*	*	*	*	TAP (INH)	TPA (INH)	INX (INH)	DEX (INH)	CLV (INH)	SEV (INH)	CLC (INH)	SEC (INH)	CLI (INH)	SEI (INH)
1	SBA	CBA	*	*	*	*	TAB (INH)	TBA (INH)	*	DAA (INH)	*	ABA (INH)	*	*	*	*
2	BRA (REL)	*	BHI (REL)	BLS (REL)	BCC (REL)	BCS (REL)	BNE (REL)	BEO (REL)	BVC (REL)	BVS (REL)	BPL (REL)	BMI (REL)	BGE (REL)	BLT (REL)	BGT (REL)	BLE (REL)
3	TSX (INH)	INS (INH)	PUL (A)	PUL (B)	DES (INH)	TXS (INH)	PSH (A)	PSH (B)	*	RTS (INH)	*	RTI (INH)	*	*	WAI (INH)	SWI (INH)
4	NEG (A)	*	*	COM (A)	LSR (A)	*	ROR (A)	ASR (A)	ASL (A)	ROL (A)	DEC (A)	*	INC (A)	TST (A)	*	CLR (A)
5	NEG (B)	*	*	COM (B)	LSR (B)	*	ROR (B)	ASR (B)	ASL (B)	ROL (B)	DEC (B)	*	INC (B)	TST (B)	*	CLR (B)
6	NEG (IND)	*	*	COM (IND)	LSR (IND)	*	ROR (IND)	ASR (IND)	ASL (IND)	ROL (IND)	DEC (IND)	*	INC (IND)	TST (IND)	JMP (IND)	CLR (IND)
7	NEG (EXT)	*	*	COM (EXT)	LSR (EXT)	*	ROR (EXT)	ASR (EXT)	ASL (EXT)	ROL (EXT)	DEC (EXT)	*	INC (EXT)	TST (EXT)	JMP (EXT)	CLR (EXT)
8	SUB (A) (IMM)	CMP (A) (IMM)	SBC (A) (IMM)	*	AND (A) (IMM)	BIT (A) (IMM)	LDA (A) (IMM)	*	EOR (A) (IMM)	ADC (A) (IMM)	ORA (A) (IMM)	ADD (A) (IMM)	CPX (A) (IMM)	BSR (REL)	LDS (IMM)	*
9	SUB (DIR) (A)	CMP (DIR) (A)	SBC (DIR) (A)	*	AND (DIR) (A)	BIT (DIR) (A)	LDA (DIR) (A)	STA (DIR) (A)	EOR (DIR) (A)	ADC (DIR) (A)	ORA (DIR) (A)	ADD (DIR) (A)	CPX (DIR) (A)	*	LDS (DIR)	STS (DIR)
A	SUB (IND) (A)	CMP (IND) (A)	SBC (IND) (A)	*	AND (IND) (A)	BIT (IND) (A)	LDA (IND) (A)	STA (IND) (A)	EOR (IND) (A)	ADC (IND) (A)	ORA (IND) (A)	ADD (IND) (A)	CPX (IND) (A)	JSR (IND)	LDS (IND)	STS (IND)
B	SUB (EXT) (A)	CMP (EXT) (A)	SBC (EXT) (A)	*	AND (EXT) (A)	BIT (EXT) (A)	LDA (EXT) (A)	STA (EXT) (A)	EOR (EXT) (A)	ADC (EXT) (A)	ORA (EXT) (A)	ADD (EXT) (A)	CPX (EXT) (A)	JSR (EXT)	LDS (EXT)	STS (EXT)
C	SUB (IMM) (B)	CMP (IMM) (B)	SBC (IMM) (B)	*	AND (IMM) (B)	BIT (IMM) (B)	LDA (IMM) (B)	*	EOR (IMM) (B)	ADC (IMM) (B)	ORA (IMM) (B)	ADD (IMM) (B)	*	*	LDX (IMM)	*
D	SUB (DIR) (B)	CMP (DIR) (B)	SBC (DIR) (B)	*	AND (DIR) (B)	BIT (DIR) (B)	LDA (DIR) (B)	STA (DIR) (B)	EOR (DIR) (B)	ADC (DIR) (B)	ORA (DIR) (B)	ADD (DIR) (B)	*	*	LDX (DIR) (B)	STX (DIR) (B)
E	SUB (IND) (B)	CMP (IND) (B)	SBC (IND) (B)	*	AND (IND) (B)	BIT (IND) (B)	LDA (IND) (B)	STA (IND) (B)	EOR (IND) (B)	ADC (IND) (B)	ORA (IND) (B)	ADD (IND) (B)	*	*	LDX (IND) (B)	STX (IND) (B)
F	SUB (EXT) (B)	CMP (EXT) (B)	SBC (EXT) (B)	*	AND (EXT) (B)	BIT (EXT) (B)	LDA (EXT) (B)	STA (EXT) (B)	EOR (EXT) (B)	ADC (EXT) (B)	ORA (EXT) (B)	ADD (EXT) (B)	*	*	LDX (EXT) (B)	STX (EXT) (B)

DIR = Direct Addressing Mode
EXT = Extended Addressing Mode
IMM = Immediate Addressing Mode

IND = Index Addressing Mode
INH = Inherent Addressing Mode
REL = Relative Addressing Mode

A = Accumulator A
B = Accumulator B

*Unimplemented Op Code

Fig 5:4. Instruktionsschema för 6800.

mer ett block som är betecknat PIA. Beteckningen står för Parallell Interface Adapter, vilket innebär att data kan in- och utmatas i parallell form. Kontrollledningarna ger information om in- eller utläsning. En annan typ av in/utmatningskrets är ACIA som står för Asynchronous Communications Interface Adapter. Denna är avsedd för dataöverföring i serie med asynkron överföring. Det innebär, att data i sig triggar klockgeneratör för inmatning av databitarna. Vid synkron överföring har man en separat klockpuls vid överföringen. Kretsar för detta brukar be-

tecknas SSDA (Synchronous Serial Data Adapter).

Olika långa instruktioner:

Ett, två eller tre ord

Till varje mikroprocessortyp hör ett instruktionsschema. Som exempel visar vi i fig 4 instruktionerna för 6800. Låt oss sätta in 86, som vi hade i vårt program, i schemat. Den vänstra spalten anger den mest signifikanta biten (8) och den övre raden svarar mot den minst signifikanta biten (6). I korsningspunkten läser vi LDA A. Under detta står (IMM), som innebär att inläsningen sker omedelbart.

Den andra instruktionen i vårt program, 7E, finner vi i korsningspunkten för 7 och E där vi läser JMP (EXT). Denna har utsträckt adressering och omfattar totalt 3 bytes.

Man skall lägga märke till att instruktionerna alltså har olika längd. Kommandot följs av 2, 1 eller ingen adress. Tillverkarnas dokumentation anger längden och betydelsen av instruktionerna. Oftast måste man ha tillgång till mer uppgifter än vad som visas i fig 4.

Men med de här enkla exemplen hoppas vi att begreppen klarnat lite ifråga om det grundläggande. ■

Monitorprogrammet

— Datorns övervakare

AV TOMMY BLADH

Monitorprogrammet ligger fast i datorn i ett PROM. Det ser till att datorn direkt kan betjäna dig efter tillslag. Monitorprogrammet ger alltså rutiner för in- och utmatning av text på terminalen, skrivaren och kassettbandspelaren.

Ett sådant program kan också innehålla rutiner för flyttning av data i minnesceller och andra finesser. Flexskiveenheten kräver dock ett speciellt operativsystem (program). Se avdelning specialprogram.

■ Datorbygget i den här boken kan ske i två faser. Vi kan göra en enklare (och billigare) version där vi använder ett in/utorgan bestående av hexadecimala lysdiodindikatorer och likaså ett hexadecimalt tangentbord. I det senare ingår förutom de decimala tangenterna 0–9 och de hexadecimala A–F även ytterligare fyra funktionstanger. Tillsammans med dubbelutnyttjning av tangenterna A–E klarar vi med monitorprogrammets hjälp åtta funktioner. Presentationen av resultat sker givetvis på den hexadecimala indikatorn.

Efter ett tag vill vi kanske bygga ut datorn till en mer generell maskin och många tar förmodligen direkt detta steg. Det innebär anslutning av ett alfanumeriskt in/ut organ – en terminal som arbetar med ASCII-kod. Nuförtiden betyder det oftast att presentationsenheten består av en vanlig TV. Tangentbordet är fullt utbyggt så att det innehåller samtliga tecken i alfabetet, siffror, special- och kontrolltecken. Härvid finns en mer

eller mindre fullständig ASCII-repertoar att tillgå. Exempel på ett sådant organ är den intelligent terminalen vilken beskrivs på annan plats. Nu är vi inte längre begränsade till de sex hexadecimala indikatorerna. I stället finns 24 rader om 40 tecken att visa informationen på. Genom de många unika tangenterna kan fler funktioner införas i monitorprogrammet – ett program som till innehållet skiljer sig från motsvarande program i datorn med hexadecimalt in/ut organ, men vars funktioner är mycket likartade. Likheter är avsiktliga för att man inte skall behöva lära om vid övergången till en "riktig" terminal.

Innan vi går närmare in och beskriver detaljerna i ett "hexadecimalt" monitorprogram (ECBUG) och ett "alfanumeriskt" (TBUG), kan det vara lämpligt att räkna upp de olika varianterna. Vi har:

Namn	processor	terminal
ECBUG	6800	hex
F0BUG	6809	hex
TBUG2	6800	alfa
CBUG2	6809	alfa

De båda hexadecimala monitorprogrammen rymms i en lk byte EPROM-kapsel medan de andra fått växa ut i en 2k kapsel.

Bättre ett monitorprogram än en imponerande panel!

Till vad används då ett monitorprogram och varför skall det nödvändigtvis ligga i ett PROM?

Monitorprogrammet finns i ett fast minne enbart för att direkt kunna betjäna dig när du knäppt på nätströmbrytaren. Visst kan man förse datorn med en imponatorpanel, överfylld med lysdioder och strömbrytare, så som den första generationens hobbydatorer var bestyckade. Då kan man via strömbrytarna "knappa" in en pri-

mitiv laddare som i sin tur laddar en mer sofistikerad laddare som i sin tur . . . Men ack så omständigt! Det är i stället mycket mera praktiskt att alltid ha ett lämpligt urval funktioner till hands när man väcker datorn till liv.

Två rutiner bildar grunden

Vad bör man då kräva av dessa funktioner? Eftersom datorn förmodligen är tänkt att kunna uträtta något nyttigt, bör det finnas en rutin för snabb och säker inladdning av körbara program. Man behöver således en laddningsrutin som kan ta hand om data, vilket formaterats enligt en standardiserad metod.

När sedan väl programmet är infört någonstans i datorns RAM, måste datorn övertalas att börja utföra bearbetning (exekvering) av programmet. Man behöver då en rutin som får datorn att starta programexekveringen på ett noga angivet ställe.

Dessa två rutiner är egentligen allt som behövs. Har man bara dem, kan vilket program som helst hämtas in och köras. Det kan vara ett spelprogram som "Spacewar" eller ett operativsystem med flexskivor för programutveckling. Eftersom det brukar bli lite plats över i monitor-PROM:en, klämmer man in en del andra nyttiga rutiner. Ett PROM besitter ju den goda egenskapen att ej tappa sitt innehåll bara därför att ett program spårar ur eller "skenar".

Resten av monitorn brukar därför utgöras av program för manipulering av dataminnet, för avlusning (DEBUGing) av egna program samt för att kunna undanlagra innehållet i dataminnet på yttre medium, exempelvis en kassettbandspelare.

I mikrodatorn som vi bygger finns en serieport (ACIA) på ad-

ress \$8008 som används för terminalkommunikation. Det är en av förutsättningarna för TBUG. Samma adress använder f ö exempelvis MINIBUG II och III samt SWTBUG.

Genom att TBUG är kompatibel med MIKBUG på "systemrutin"-nivå, blir det enkelt att anpassa befintlig programvara. För dem som inte vet vad MIKBUG är, kan vi ge följande information: I mikrodatorns barndom då M6800 kom till lanserade Motorola ett utvärderingskort innehållande ett relativt enkelt monitorprogram med namnet MIKBUG. Kortet innehöll vidare en minimalt avkodad minnesdisposition som var lämpligt för det ursprungliga enkla ändamålet, men absolut vansinnigt för ett system som vårt. Men det är inget att göra åt. Eländet blev nämligen standard för M6800 system. Det medför att I/O portarna hamnade i mitten av minnesutrymmet, monitorns RAM vid A000 och monitorn själv vid E000. Som resultat har 6800-datorn gränsen 32k kontinuerligt RAM.

TBUG:s funktioner steg för steg

Ett system med TBUG presenterar sig självt efter återstart genom att skriva texten "TBUG2" på terminalen. Därefter skrivs på nästa rad en asterisk, "*", ut och datorn väntar på kommando. Varje kommando utgörs av en enställig bokstav efter vilken TBUG väljer rätt rutin. Det finns ett antal standardfunktioner plus några "bra att ha" rutiner. Standardfunktioner kan vi kalla dem som återfinns i de flesta på marknaden förekommande monitorerna. TBUG2 innehåller vanligtvis följande funktioner:

- A Hoppa till PROM för editor/assembler.
- B Hoppa till PROM för BASIC.
- C Fortsätt efter brytpunkt.
- D Dump i hex/ASCII 8 bytes/rad.
- E Extended instruktionssökning.
- F "Mini disk boot", dvs ladda diskoperativsystemet.
- G Gå till programadress och exekvera.
- H Beräkna checksumma adr1-adr2

- J Gå till subrutin. Skriv ut registren.
- K Kopiera minnesinnehåll.
- L Ladda minnet från S1-format.
- M Minnesmanipulering.
- N Stega en instruktion.
- O Fyll minnesblock med ett visst tecken.
- P Lagg ut minnesinnehållet i S1-format.
- Q Mät RAM-storleken.
- R Registerutskrift.
- S Sök 2 bytes kombination.
- T Ladda S1-format med offset.
- U Tag bort brytpunkter.
- V Sätt in brytpunkter.
- W Minnestest.
- X Sök och byt AA nnnn mot AA mmmm.
- Y Verifiera två minnesblock.
- Z Dump i hex/ASCII 16 bytes/rad.

Detaljbeskrivning av de vanligaste funktionerna

D, Dumpa minnet i hexadecimal och ASCII-format. Programmet frågar efter start och stoppadresser. Därefter skrivs adress och minnesinnehållet ut. Minnesinnehållet skrivs först som åtta hexadecimala bytes och därefter som åtta ekvivalenta ASCII-tecken. Varje byte översätts då till ett ASCII-tecken. Kontrolltecken byts dock ut mot en punkt.

Dumprutinen är uppdelad så, att dumputskriften sker i en subrutin. Denna kan därför med fördel användas av egna program under exempelvis avlusningsskedet. Intressanta data-areor kan skrivas ut på ett enkelt vis. I detta fall får start- och stoppadresserna i BEGA resp ENDA laddas av det egna programmet.

Exempel på dumprutinen:

***D**

START SLUT

>0162 016F

0160 52 45 53 54 4F 52 45 00

RESTORE

0168 10 8D 45 4E 44 00 0B 41

. . END. .A

* Trots att man angivit 0162 som startadress börjar dumpen på 0160. Detta hänger samman med att startadressen maskas till att sluta på 0

eller 8. Det ger snyggare och mer lättlästa listor.

F, FLEX boot. Används för att initialladda diskoperativsystemet FLEX. Kräver minst en drivenhet. En länkad systemskiva skall sitta i enhet noll.

G, Gå till minnesadress och starta exekvering av programmet. Rutinen används för att starta program från adresser som man själv ger. Efter G måste därför en 4-ställig hexadecimal adress ges. G-rutinen lägger då in den givna adressen på programräknarens läge i den stack vars stackpekare sparats i RAM-cellerna SP. Därefter laddas stackpekaren från SP och programmet gör en RTI, return from interrupt. Härigenom får man också med sig de i stacken undanlagrade processorregistren CC, B-ackumulator, A-ackumulator och X-registret förutom programräknaren. Dessa har kunnat ändras med M-kommandot om man velat. Vid utförande av G insätts även eventuella angivna brytpunkter i minnet. Se under V. Exempel på G-rutinen:

***G AD03**

H, Beräkna checksumma mellan två adresser genom bytevis addition utan hänsyn till spill med 16 bitars resultat. Funktionen är användbar vid kontroll att laddning från tape gått bra, alternativt att minnesinnehållet är opåverkat efter programurspårning.

L, Ladda dataminnet. Data skall vara formaterat enligt Motorolas S1-standard. I S1 är data representerat med ASCII-tecken. Varje inspelning inleds med tecken S1. Därefter följer uppgift om antalet dataord, adress till dataminnet, ett varierande antal bytes samt, slutligen, en checksumma beräknad på denna information. Detta laddningsformat är mycket säkert men tyvärr något långsamt. För att uppnå maximal säkerhet vid laddning bufferterar TBUG den inhämtade adressen och alla bytes. Först när checksumman mottagits och godkänts, överförs data till minnet; detta i motsats till MIKBUG och de båda MINIBUG-monitorerna.

Där lagras man data i minnet alltefter de mottas. Får man då ett checksummafel orsakat av att minnesadressen ej mottagits korrekt, kan data ha laddats med helt felaktiga adresser. Resultatet blir förmodligen att laddningen måste göra om. I TBUG medför ett checksummafel bara att data ej överförs från den temporära bufferten till minnet.

Laddningsprogrammet indikerar upptäckten av ett checksummafel genom utsändning av 'BELL' och '?' till terminalen. Därefter fortsätter den att ladda. Om man vill, kan bandspelaren stoppas och kassetten backas en bit. Sedan är det bara att sätta igång den igen och försöka få in det avsnitt som missades vid avspelnigen.

Laddningsprogrammet avslutas när "END of FILE" av inspelningen mottas. Detta består av enbart tecknen S9. Därefter återgår programmet till att vänta på nytt kommando. En asterisk, '*', utskrivs som indikation.

M, Minnesmanipulering. Detta kommando används för att läsa och ändra minnesceller. Efter M anger man därför alltid en 4-ställig, hexadecimal adress. Därvid skriver datorn ut innehållet i den utpekade cellen samt ett blanktecken. Nu har man att välja mellan ett par olika alternativ. Skall innehållet ej ändras utan bara läsas, kan ett '.'-tecken eller '-'-tecken (minus) ges.

I det förra fallet byter datorn rad, skriver ut nästa minnesadress samt innehållet däri.

I det senare fallet byter datorn också rad men "backar" minnesadressen ett steg, varefter minnesinnehållet skrivs ut.

Vanligtvis brukar tecknen LF (line-feed) och '^' (up-arrow) användas till dessa funktioner. Emellertid passar dock de här två tecknen inte så bra när vi har ett "svenskt" tangentbord. Därför har två andra oskiftade tecken tagits i stället.

För att ändra i den öppnade minnescellen krävs bara att två hexadecimala (0-F) tecken ges. Därefter byter datorn automatiskt rad

och skriver ut nästa minnesadress och dess innehåll. Denna automatik är praktisk då man önskar knappa in ett större antal bytes, exempelvis ett program från någon lista.

Varje gång man ändrar i en minnescell, kontrolleras att ändringen "fastnar". Antar minnescellen inte samma värde som angivits, skriver datorn ut ett frågetecken och piper (BELL). Den går dock ej ur M-moden, utan byter bara rad och skriver på nytt ut samma minnesadress och dess aktuella innehåll. Denna egenskap att inte lämna M-moden är avsiktligt inkonstruerad i TBUG. Har man suttit några gånger vid Motorolas utvecklingssystem *Exorciser* och försökt ladda ett antal konsekutiva write only-register och varje gång blivit utsparkad till kommandoslingan med ett 'No change' i ansiktet kan det inte hjälpa att man blir lite lömsk!

Ytterligare en värdefull egenskap har tillägnats M-programmet, nämligen "offset"-beräkning. Instruktionslistan för M6800 upptar ett antal hoppinstruktioner där hoppet utförs genom att innehållet, "offset", i byten som efterföljer operationskoden adderas till programräknaren. Eftersom offsetvärdet är i två komplementform och kan anta positiva och negativa värden, kan hopp utföras både framåt och bakåt.

Att räkna ut detta för hand är rätt besvärligt, speciellt vid hopp bakåt. I TBUG är det mycket enkelt: Efter att ha öppnat den cell som innehåller offsetvärdet skriver man sedan bara bokstaven O och därefter adressen dit hoppet skall gå. Datorn skriver ut det beräknade värdet, byter rad och skriver ånyo ut samma adress och dess innehåll. Vill man nu lägga in det offset-värde datorn har givit är det bara att skriva det. Skulle adressen, dit hoppet var tänkt att gå, ligga utom räckhåll skriver datorn ett frågetecken och piper. Därefter sker radbyte och utskrift av samma adress och innehåll igen.

I CBUG för 6809, levereras automatiskt ett långt (16-bit) offset-värde om det korta inte räcker.

6809 processorn har instruktioner för relativa hopp över hela minnesområdet.

M-rutinen avbryts genom att man slår ett otillåtet tecken, t ex 'CR'. Exempel på M-rutinen:

```
*M A042 10 öppna cell A042, ge '.'
*A043 22 00 ändra cell A043 till 00
*A044 1A ge 'CR' och avsluta M
*M E16D 24 OE190 22 beräkna.
*E16D 24 22 . och skriv offsetvärdet
```

P, Punch data. Innehållet mellan två adresser i datorns primärminne överförs till ett yttre medium som t ex kassettbandspelare. Data organiseras i poster (records) enligt S1-formatet. Varje post innehåller identifieraren S1 följt av en antalsuppgift på datamängden i posten. Vidare finns maskinadress dit data skall lagras samt slutligen en checksumma beräknad på innehållet i S1 posten.

Programmet frågar självt efter start och stoppadresserna. Efter att dessa givits kan en klartextrubrik skrivas. Den avslutas med vagnretur varefter utmatningen startar. Bandspelaren sätts därför lämpligen igång innan det avslutande retur tecknet ges.

Efter den sista datainspelningen matas automatiskt ett EOF (end of file) ut, som i det här fallet får betecknas av S9. Både L- och P-rutinerna lägger ut DC-tecken för automatisk start/stopp av bandspelare eller remsläsare/stans.

R, Registerutskrift.

Utskrift av processorregistren som de ligger undanlagrade i den stack, vars stackpekare lagrats i RAM-cellerna SP. Denna undanlagring görs efter varje stegning av programmet med N samt efter påträffandet av en brytpunkt. Vid dessa tillfällen sker automatisk registerutskrift. Eftersom man med M-funktionen kan ändra innehållet i denna "pseudo"-stack och vid utförandet av G, N eller C-funktionerna få med sig nya värden i processorregistren, behövs en rutin för kontroll av ändringarna. R skriver ut registren exakt som de ligger på stacken. Först kommer därför CC, sedan A-ackumulator, B-ackumulator, X-register, pro-

gramräknaren samt sist även stackpekaren. Då stackpekaren alltid pekar på nästa lediga position i stacken, krävs att man adderar en etta till dess värde när M-funktionen nyttjas för ändringar.

Exempel på hur R och M opererar på stacken:

***R**

C=CD B=44 A=12 X=0100

P=AD03 S=A038

***M A039 CD**

***A03A 44 20**

***A03B 12**

I ovanstående exempel ser vi att stackpekaren har värdet A038. Då ligger statusordet på A039, vilket syns med M-funktionens hjälp. Därefter ändras värdet som motsvarar B-ackumulatorn från 44 till 20. I 6809 monitorn skrivs självfallet även de nya Y, U registren ut.

V, Ange brytpunkt (break point) i minnet. Brytpunkter används under avslusning av program lagrade i RAM. Funktionen utförs genom att operationskoden i användarprogrammet på den angivna adressen byts mot en SWI operationskod. Den ursprungliga opkoden sparas undan i brytpunktstabellen tillsammans med minnesadressen. Detta utbyte sker dock först vid utförandet av G- och C-funktionerna. När sedan processorn påträffar en SWI avbryts körningen av målprogrammet och kontrollen återgår till TBUG. Bland det första som då sker är utbyte av de insatta SWI op-koderna mot de ursprungliga. Först därefter sker utskrift av processorregistren. Upp till fem brytpunkter kan samtidigt anges. En brytpunkt specificeras genom angivandet av en 4-ställig adress omedelbart efter V-kommandot:

*** V 2200**

U, Tag bort alla brytpunkter. Eftersom inga brytpunkter lagras i minnet så länge programkontrollen befinner sig i TBUG, är det mycket enkelt att "döda" dem. U-rutinen nollställer helt enkelt den räknare som håller reda på antalet brytpunkter.

Observera, att om ett program som innehåller brytpunkter sparas

ur eller går in i en slinga, och man blir tvungen att trycka på RE-START, förblir SWI (\$3F) kvar i minnet. De måste då plockas bort manuellt med M-rutinen och programlistan. Exempel på U-rutinen:

*** U**

Några användbara TBUG-rutiner

Följande generella (sub)rutiner kan med fördel användas i egna program. (M) anger adresskompatibilitet med MIKBUG.

BADDR (M) \$ E047 Bygg upp en 16 bit adress av 4 mottagna tecken.

BYTE (M) \$ E055 Skapa en hexadecimal byte av två mottagna tecken.

OUTH L (M) \$ E067 Konvertera vänster halva i en byte till ASCII och skriv ut den.

OUTH R (M) \$ E06B Konvertera höger halva i en byte till ASCII och skriv ut den.

OUTCH (M) \$ E075 Mata ut ett tecken till terminalen från A-ackumulatortorn.

INCH (M) \$ E078 Tag in ett tecken från terminalen i A-ackumulatortorn. Paritetsbiten nollställd.

PDATA1 (M) \$ E07E Skriv ut en textsträng

BEGEND \$ E09B

på terminalen. X-registret skall peka på strängen som skall avslutas med tecknet \$ 04.

Begär in två 4-ställiga hexadecimala adresser som lagras i BEGA och EN-DA.

ENDADR \$ E0A0

Begär in en 4-ställig hexadecimal adress som lagras i ENDA.

BADDRS \$ E0A6

Begär in en 4-ställig hexadecimal adress i X-registret. Lägger även ut ett blanktecken.

INHEX (M) \$ E0AA

Begär in ett ASCII-tecken från terminalen och omvandla detta till binärt 0 - F. Återvänd med tecknet i A-ack.

INHEX2 (M) \$ E0AC

Som IN-HEX men tecknet ligger från början i A-ack.

OUT2H (M) \$ E0BF

Skrivaren skriver två hexadecimala tec-

	ken från en minnescell. X-reg skall peka på minnescellen. X-reg ökas ett steg.	OUTS (M) \$ E0CC	Skriver ett mellanslag på terminalen.	INEEE (M) \$ E1AC	Tag in ett tecken från terminalen till A-ack. Nollställ pariteten. Eka ut tecknet till terminalen om flaggan EKO är nollställd.		
OUT2HA (M) \$ E0C1	Skrivaren skriver två hexadecimala tecken från innehållet i A-ack. X-reg ökas ett steg.	START (M) \$ E0DO	Kallstart i MIK-BUG, men varmstart i TBUG.				
		MIKCON (M) \$ E0E3	Entry-punkt för MIK-BUG:s CONTRL-slinga.	INCHP \$ E1BA	Tag in ett tecken från terminalen till A-ack. Behåll alla åtta bitarna.		
OUT2HT \$ E0C2	Skrivaren skriver två hexadecimala tecken från innehållet i A-ack. Påverkar ej X-reg.	PRREG \$ E11A	Subrutin för utskrift av processorregistren. Stackpekaren lagrad i SP.	OUTEEE (M) \$ E1D1	Skriv ut ett tecken på terminalen från A-acc. B-ack. påverkas ej.		
OUT4HS (M) \$ E0C8	Skrivaren skriver fyra hexadecimala tecken från två minnesceller vilka utpekats av X-reg. Därefter skrivs ett mellanslag. X-reg ökas två steg.	CRLF \$ E131	Subrutin för vagnretur och radbyte.	PIIP \$ E291	Skriv '?' på terminalen samt ge BELL-signal.		
		CMPADR \$ E183	Subtrahera en 16 bit-adress i ENDA från en 16 bit-adress i BEGA. Återvänd med statusflaggorna.	PDUMP \$ E384	Subrutin för dump-utskrift i hex och ASCII. Lägg upp startadress i BEGA och stopp-adress i ENDA före anropet.		
OUT2HS (M) \$ E0CA	Skrivaren skriver två hexadecimala tecken från en minnescell vilken utpekats av X-reg. Därefter skrivs ett mellanslag. X-reg ökar ett steg.	MCLOFF (M) \$ E19C	Textsträng innehållande kontrollord för avstängning av remsstans/läsare samt utskrift av CR + LF + '*'.	Monitorprogrammet behöver RAM-utrymme TBUG behöver liksom MIKBUG, MINIBUG etc ett eget RAM, beläget inom adressområdet \$ A000 – A07F. Följande celler används:			
		MCL (M) \$ E19D	Samma som ovan minus ett kontrollord	Adr	Namn	Ant	Förklaring
				A000	IOV	2	IRQ vektor
				A002	BEGA	2	Start adress

A004	ENDA	2	Slut adress
A006	NIO	2	NMI vektor
A008	SP	2	Stack pekare
A00A	CKSM	1	Checks. cell
A00B	BYTECT	1	Räknare
A00C	XHI	1	Temp X-reg Ms
A00D	XLOW	1	Temp X-reg Ls
A002	TW	2	Temporär slask
A00E	MCONT	1	S1 byte räkn.
A00F	SWIVEK	2	SWI vektor
A011	EKO	1	Eko flagga
A012	DMASK	1	Dump mask
A040	STACK	45	Maskinstack
A041	VFLAG	1	Ant. brkpt's
A042	BPTAB	15	Brkpt tabell
A051	TEMP	2	Temporär slask
A051	BUFF	50	Buffertarea

De hexadecimala monitor-programmens funktioner

Då hanteringen och datapresentationen skiljer sig ganska markant för ECBUG/FOBUG gentemot TBUG/CBUG skall vi ta en relativt detaljerad titt även på dessa.

De 20 tangenterna märkta 0-9 samt A, B, C, D, E, F, L, P, R och S används för att mata in hexadecimala tal för data och adresser samt för olika kommandon. För att ge en viss likhet med de monitorprogram vilka har en bildskärm som presentationsenhet har vissa funktioner införts för att förenkla handhavandet för användaren. Detta har även den fördelen, att det blir relativt lätt att vänja om sig vid senare övergång till en bildskärms-terminal.

De fyra första indikatorerna används av monitorprogrammet för att visa adresser och andra 16-bitars tal. De två övriga används för att visa data och 8-bitars tal. Båda monitorprogrammen har samma kommandon och identiska egenskaper. Till varje monitorprogram medföljer en programlistning med kommentarer.

När punkten på första indikatorn blinkar långsamt innebär det att programmet är i kontrollslingan och väntar på ett kommando. Det finns då åtta kommandon att välja mellan:

A, Alter Memory: Minnesmanipulering. Efter att ha tryckt på A blinkar punkten med högre hastighet och det är nu bara att knappa in en 4-ställig hexadecimal adress till den minnescell man vill undersöka. Adressen visas på de fyra första indikatorerna och data i adressens minnescell på de två sista indikatorerna. Nu finns det fyra vägar att fortsätta på:

Om data skall ändras på den aktuella minnespositionen gör man så enbart genom att ange två hexadecimala tecken (0-9, A-F). Om det nya värdet fastnar i minnet, dvs om det finns RAM-minne på den adressen, visas det nya värdet ett kort ögonblick varefter programmet stegar vidare till efterföljande minnesposition och visar datainnehållet i den minnescellen. Inmatning av maskinkod från en lista blir på det sättet enkel.

Om man bara vill undersöka vad som finns i de olika minnescellerna, kan man stega genom minnet med L-tangenten. Med P-tangenten kan man stega baklänges i minnet. För att återgå till kontrollslingan trycker man på R. Om man försöker ändra data på en adress där det inte finns något RAM; det kanske är ROM, EPROM eller ett "write only"-register i en periferikrets, ger datorn en BELL-puls och ingen ändring av minnesadressen sker. Det är användbart särskilt om man skall mata in olika tal i en periferikrets register för att prova dess utgångar.

Med S-tangenten kan man på ett enkelt sätt beräkna offset för relativ adressering. Det är annars ett tidsödande arbete, speciellt vid handskrivna program i maskinkod. Om vi exempelvis befinner oss i minnesposition A200 med instruktion 26 (BEQ), skall på den efterföljande minnespositionen stå offsetvärdet till den adress dit vi skall hoppa, om föregående instruktion gav ett resultat som blev noll.

Vi antar, att adressen är A1BE. På minnesposition A201 trycker vi på S-tangenten och knappar in adressen A1BE. På dataindikatorn står därefter BC, vilket är det ut-

räknade 2-komplements offsettal som datorn behöver för att hamna på rätt adress vid programmets körning. Eftersom det uträknade och visade offsetvärdet skall läggas in på den angivna minnesadressen så gör programmet det åt oss, varefter nästa minnesadress med innehåll visas på indikatorerna.

Problemet med 2-komplementoffset behöver inte längre bekymra den som gör program i maskinkod med funktionen. Om det skulle visa sig att den inmatade adressen ligger för långt ifrån, anges det med en BELL-puls utan ändring av minnesadressen.

B, Set Breakpoint: En brytpunkt är ett slags programstopp, vilket används under avlusning (eng "debugging") av program. Funktionen åstadkoms genom att man ersätter den ursprungliga opkoden med opkoden för SWI. När programmet kommer till SWI, avbryts det och registren skrivs ut (se nedan under R). Inlägg av SWI sker automatiskt vid E- eller C-kommando. Borttagning av SWI sker också automatiskt, såvida inte programmet avbryts med ett tryck på resetknappen. I så fall får man för hand ändra de minnesceller som var försedda med brytpunkter. Maximalt fem brytpunkter kan samtidigt anges. En brytpunkt inmatas genom att man trycker på B och en adress. Efter det återgår programmet till kontrollslingan.

C, Continue: Används för att fortsätta efter en brytpunkt. När registren har visats och programmet är i kontrollslingan, trycker man på C. Stackpekaren laddas då med det ursprungliga värdet, varefter en RTI utförs och programmet fortsätter.

D, Delete Breakpoints: Tar bort alla definierade brytpunkter från monitorprogrammets tabell.

E, Execute Program: Används för att starta program på en viss adress, t ex E 0100, vilken går till adress \$0100 och startar det program som finns där. Eventuella brytpunkter har dessförinnan placerats in på sina platser!

L, Load: Ladda minnet med data från kassett. Rutinen förutsätter att ett KCS-modem finns placerat i kontakten J3 på moderkortet. Data följer Motorolas S1-format. Under laddningen visas adress till varje datainspelning på adressindikatorn och varje dataord återges på data-indikatorn. Det innebär, att man vet var i minnet laddning sker och att data verkligen kommer in. Laddningen avslutas när koden S9 kommer in från kassetten, varvid programmet återgår till kontrollslingan.

P, Punch Data: Spela in data mellan två adresser på kassett. Förutsätter KCS-modem. Se kapitlet kassettpassning för den lilla terminalen. Funktionen initieras genom att man efter P anger start- och slutadress för det minnesblock som skall spelas in. Eftersom data sänds så fort sista tecknet har matats in, bör man starta bandspelaren så fort P-tangenten har tryckts ner. Under inspelningen visas adress och data på indikatorerna på samma sätt som vid laddningsfunktionen. När alla data är utmatade, avslutas det hela med koden S9. Att lagra data enligt det här formatet har den stora fördelen att kassetterna kan användas och spelas upp även över andra terminaler utan att man behöver tillgripa specialrutiner.

R, Register Display: Tangenten visar registren i processorns stack. Först visas på dataindikatorn CC-registret (condition code), med ett tryck på S-knappen visas sedan i tur och ordning B-ackumulatören, A-ackumulatören, X-registret, PC-värdet och S-värdet. Vid fortsatt tryckning på S-knappen upprepas det hela och programmet återgår till kontrollslingan med R-knappen.

För 6809-utförandet av monitorprogrammet presenteras registren i följande ordning: CC, A, B, DP, X, Y, U, PC och S. Till denna rutin kommer programmet automatiskt vid SWI, NMI och IRQ om RAM-vektorer för dem ej har ändrats av användaren. ■

Datorns massminnen: Bandkassetter och flexskivor

AV TOMMY BLADH

För att lagra våra program behöver vi ett massminne av något slag eftersom datorns minne snabbt blir för litet och därtill tappar informationen vid frånslag.

Lämpliga massminnen är kassetband och flexskivor. Här orienterar vi om olika tekniska lösningar.

■ Till varje dator behövs ett system för att kunna lagra stora datamängder. Vi brukar kalla detta för *massminne*. Till skillnad mot datorns primärminne, i vårt fall området 0–64k, är massminnet inte direkt åtkomligt av centralenhetens bussar. Massminnet består i stället av enheter vilka ansluts över speciella anpassningar. I hobbysammanhang var länge kassetbandspelaren och kompaktkassetten det vanligaste mediet för lagring av stora datamängder. Vissa pionjärer som började mycket tidigt med hemdata använde remsläsare/stans. Databärande media var en perforerad pappersremsa som kunde avkännas mekaniskt eller elektroniskt.

Nu börjar flexskivan dock bli alltmer vanlig i mikrodatorsystemen och det är kanske ett tecken på att dessa trampat ur barnskorna. En flexskiva, diskett eller floppy-disk, är en skiva av böjlig plast som belagts med magnetiskt material. Den förvaras permanent i ett fodral vars invändiga beklädnad suger upp dammpartiklar och lättare föroreningar. Skivorna finns i två olika storlekar, 8" och 5 1/4". Beroende på drivenhet kan ena eller båda sidorna på skivan använ-

das. Olika lagringssätt förekommer men så stora mängder som upp till en miljon tecken kan lagras oformatterat på den lilla skivan.

Bandspelare som massminne
Fortfarande är kassetbandspelaren det absolut vanligaste massminnet för privatdatorer. Åtminstone gäller det för system i det lägre prisområdet. Normalt duger nämligen en bandspelare i 200 kronors klassen även om resultatet oftast förbättras med mer avancerade apparater. Räkneverk och kontroll av in- och avspelningsnivåer är ett plus. Dolby eller andra brusreduktionsystem skall dock inte användas. Kassetter skall man välja med omsorg. De allra billigaste bör undvikas eftersom de brukar ge problem med bandföring och drop-outs. Beroende på inspelningsmetod kan data lagras och återvinnas med hastigheter av 300–4800 baud, dvs 30–480 tecken per sekund. Det sker utan att några ingrepp eller förändringar gjorts i bandspelaren. Med digital inspelningsteknik kan likströmspulser av olika polaritet matas direkt in i bandspelaren. Dess egna förstärkare kopplas bort och egna anpassningskretsar används i stället. Detta kan göras förvånansvärt enkelt. Några operationsförstärkare, motstånd och kondensatorer är allt som behövs. Överföringshastigheter upp mot 9600 baud blir då möjliga.

Standard finns

Är det nu så illa ställt att alla använder olika system och försöker "väsas" dem till maximal hastighet? Hur skall då något vettigt programutbyte kunna ske om vi inte kan läsa varandras kassetter? Det är faktiskt inte så illa. En standard för audio inspelade kassetter existerar nämligen. Den kallas *Kansas City Standard (KCS)* efter den plats där den definierades vid ett symposium arrangerat av tidningen **BYTE** 1975. Enligt KCS lagras data i följande audioformat: En nolla består av fyra perioder av frekvensen 1200 Hz. En etta av åtta 2400 Hz perioder. Formatet



Fig 7:1. Så här kan en flexskivenhet se ut, eller rättare sagt två enheter, monterade på varandra. Tillverkare är Micropolis, ett ungt framgångsrikt amerikanskt företag. Andra exempel är Shugart, Pertec, BASF, Siemens, MPI, Teac m fl. Vanligast är 5 1/4" flexskivor men 8" skivor förekommer även i amatörsammanhang. Tidigare har skivminnena varit för dyra i smådatorsystem, men där råder nu en intensiv utveckling och priserna rasar. Skivminne, i s k Winchesterteknik, finns nu att få med kapacitet upp till 20 MByte för anslutning till smådatorer.

ger en överföringshastighet av 300 baud. Långsamt kan man kanske tycka, men inget hindrar att man lagrar egna data med en snabbare metod och enbart kommunicerar utåt via det långsammare men säkra KCS formatet.

I ett 6800/6809 system har vi förmodligen ett serieport, ACIA, från datorn mot kassetanpassningskretsarna. Då blir detta relativt okomplicerat. Genom att ta ACIA klockan på 4800 Hz (16×300) delar vi lätt fram både 1200 och 2400 Hz. Mottagaren innehåller lite mer komponenter men klarar då även

att återvinna 4800 Hz klockan ur dataströmmen. Genom att mata in denna på ACIA:s klockingång på mottagningssidan får vi stor immunitet mot bandspelarens svaj. När data svajar kommer därför klockan att svaja lika mycket.

Fördelar – nackdelar

Den största fördelen med ett bandspelarbaserat massminnessystem är kostnaden. Den billiga bandspelaren eller det redan befintliga stereodäcket. Att få tag i prisvärda kvalitetskassetter är inte heller något problem. Det är därför vettigt

att starta med ett sådant system om man bara är allmänt intresserad av privatdata och/eller inte överlastad i kassan.

Nackdelarna visar sig dock tämligen fort då data skapas och läses i icke sekventiell (efter varandra) följd. Band och pappersremsa är till sin natur sekventiella. Dvs man startar i början och läser framåt tills den önskade datainformationen dyker upp. Måste man ha tag i något före detta, blir det nödvändigt att spola tillbaka och börja om igen. Random-access teknik är därför ingenting för vanliga band-

spelsystem. Åtkomsttiderna blir alltför stora. I det fallet ger flexskivan överlägsna prestanda. Men även i ett flexskivesystem kan en bandspelare ha sin givna plats: Dels som kommunikationslänk utåt, dels som back-up media för skivminnena. När man sparar undan filerna på band passar den sekventiella lagringen alldeles utmärkt.

För att få bästa möjliga säkerhet bör tonhuvudena rengöras ofta. Ett drop-out som knappt märks på en musikinspelning kan totalt förstöra viktig datainformation. Vettig märkning av kassetterna är också A och O. Annars flyter snart hela programbiblioteket runt i en salig röra.

Flexskivan som massminne

När man utrustat sin dator med flexskiveenhet(er) och därtill pas-

sande operativsystem, öppnas helt nya möjligheter. Borta är de tidsödande laddningarna från kassettbandspelaren samt den plågsamma upptäckten att något blev fel i det inhämtade. Korrigeringar i program och data som tidigare måste planerats noga eftersom de tagit så lång tid att överhuvudtaget få in i maskinen, görs nu på nolltid. I sanning – en stor del av datoristens vardag blir lättare tack vare flexskivan. Låt oss därför se en smula på detta massminne; vad det är, hur innehållet organiseras och identifieras m m.

Flexskivan består som vi sade i inledningen av ett böjligt plastmaterial, belagt med ett magnetskikt. I det skyddande fodralet som skivan permanent innesluts i finns öppningar för centrumhålet, synkroniseringshålet (indexhålet) samt slitsen där skriv/läshuvudet löper. Flexskivan finns som stan-

dard i två storlekar, 8" och 5 1/4". Den lilla skivan förekommer oftast i persondatorsammanhang medan den stora är vanlig i professionella tillämpningar, som t ex Motorolas *exorciser*. Skillnaden i diameter medför naturligtvis att mer datainformation kan klämmas in på den stora skivan vid samma packningstäthet. Genom att den lilla skivan är mekaniskt stabilare kan man dock med olika metoder lättare öka lagringskapaciteten.

Spår och sektorer

På skivan definierar man ett antal koncentriska cirklar, *spår*, över vilka ett huvud kan positioneras för läsning och skrivning. Numreringen börjar från 00 ytterst till 76 innerst (35 för 5"). Huvudet spelar in ett bredare spår än nödvändigt. Sedan trimmar två raderspalter vardera kanten. Därigenom skiljs

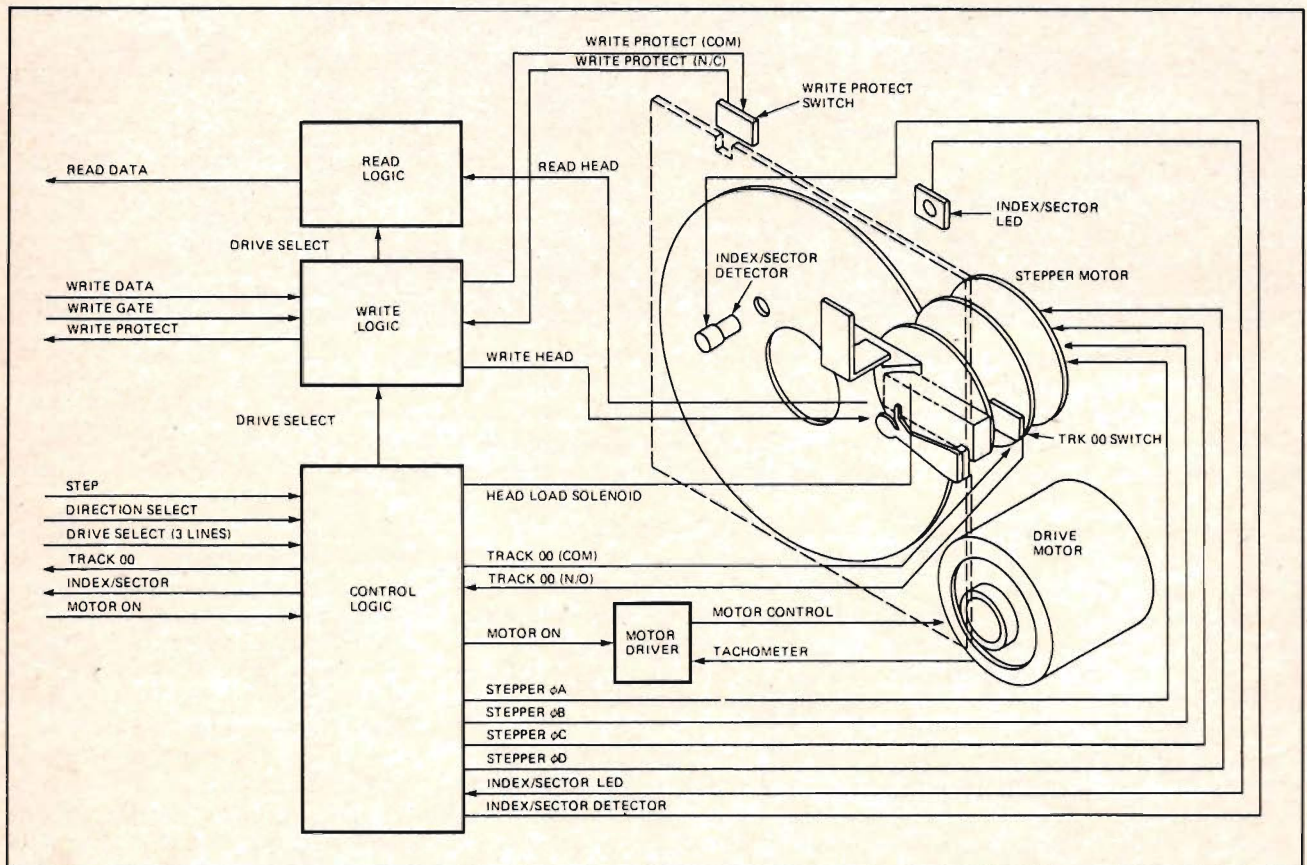


Fig 7:2. Här ser vi principen för ett flexskivsystem. En motor driver skivan med konstant hastighet och huvudet flyttas över skivytan med en stegmotor. Skivan har ett indexhål som indikerar var informationen på skivan ligger. I skivans hölje finns en flik som kan avläsnas då man vill ha skrivskydd.

spåren åt av omagnetiska tomma "gator".

Varje spår är i sin tur uppdelat i mindre enheter, *sektorer*. Dessa skiljs åt av gap. Längden av sektorerna definieras helt vid initieringen av skivan. Ett speciellt program sköter den saken. Metoden kallas *mjuksektorisering*. På skivor med *hårdsektorisering* hittas sektorstarterna i stället genom att de försetts med sektorhåll liknande innehållet. Pga bristande standardisering förekommer metoden främst i applikationssystem.

Varje sektor består av en identitetsdel och en datadel. I ID delen finns information om spårnummer, sida på skivan, sektornummer, sektorlängd samt CRC kod. Den senare består av två bytes. De övriga av en vardera. Eftersom sidan endast kan anta två tillstånd, innehåller den byten antingen 00 (sid 1), eller 01. Längden på sektorerna kan vara exempelvis 128, 256, 512 osv bytes långa. Detta anges med värdena 00, 01, 02 etc. CRC koden är en form av checksumma som beräknats över ID-innehållet. Vid läsningen beräknar man CRC på nytt och jämför med den på skivan lagrade summan. Detta garanterar innehållets kvalitet.

Datadelen av sektor består förutom av data självt också av en avslutande 16 bit CRC kod. Mellan ID och datadel samt före och efter gap. En speciell byte, adress- resp datamärke inleder sektorerna. Nu förstår vi varför en formaterad skiva alltid innehåller mindre användbart utrymme än vad reklambladet säger. Att hålla ordning på sektorerna kräver sin tribut i bytes. En sektor med större datadel ger också mer nyttigt utrymme på skivan. Tack vare färre ID delar.

Inspelning görs alltid av en hel sektor i taget. Huvudet positioneras över rätt spår. Sänks ned mot skivan och ID delarna läses i varje sektor tills den rätta påträffats. Efter c:a 60–70 μ S startar man inspelningen av datadelen. Liknande procedur sker för läsning. Fast här väntar man tills datamärket dyker upp. Hur många bytes som skall

läsas har ID delen redan informerat om.

Ökad packningstäthet

Den ursprungliga metoden som IBM gav upphov till kallas *FM* och används för enkel packningstäthet. Klockfrekvensen är 250 kHz och vid inspelningen blandas klocka och data. Genom omkastning av skrivströmmen genom huvudet registreras klock- eller datapuls. När inspelningen består av enbart nollor saknas datapulser mellan klockpulserna. Med bara ett or i data fördubblas pulsfrekvensen.

När man ville öka informationsmängden på skivan kunde inte klockfrekvensen ökas eftersom magnetmaterialet inte tillät detta. I stället inriktade man sig på att ersätta klockpulserna med data. Utan klocka för synkroniseringen uppstår dock problem när data består av sammanhängande nollbitar. Flera metoder har presenterats för att klara detta. *MFM* (modifierad FM), *M²FM* (Modifierad modifierad FM) samt *GCR*. Den sistnämnda innebär överförande av halva bytes i en 5 bits kod. Koden gör att en etta kan sättas in efter två nollor. Synkroniseringen, vilken vid dubbel packningstäthet kräver minst en VCO, klarar sig över två sammanhängande nollor. De andra metoderna är likartade och bygger på att man stoppar in en extra klockpuls i intervallen mellan saknade klockpulser. Metoden som IBM slutligen valde för dubbel packningstäthet blev *MFM*.

Skilda format

Likheterna mellan formaten är många. Index-, adress och datamärken är lika. Sektorns ID har samma utseende. Likaså datadelen utom att antalet databytes är 256. Sektorantalet är samma. Skillnaden ligger främst i märkesidentifieringen. Eftersom klocka saknas består de i stället av data. Vidare skall spår noll, dvs skivans identifieringsspur, vara inspelad i enkel packningstäthet. Möjligheterna att öka lagringskapaciteten är inte uttömda efter införande av dubbel

packningstäthet. Först ökades antalet spår inne i centrum. Den lilla skivan fick ytterligare fem spår, totalt således 40 per sida. Nu har flera fabrikanter släppt ut drivenheter med dubbla huvuden: Ett på vardera skivsidan. En signal till mekanismen kopplar in endera huvudet. Tidigare har vi sett att sidnummer finns inspelat i ID delen på sektorn. På en miniflexskiva ligger då de första 10 sektorerna på sida ett. De övriga 10 på sida två. Men det är inte slut med detta. Drivenheter med dubbel spårstäthet har nu kommit fram. Skivor avsedda för dubbel packningstäthet fordrar en något högre kvalitet. Fantastiska en miljon bytes rymms på en sådan 5 1/4" skiva. I *fig 3* kan vi se specifikationer på fyra minidriverheter av märke **MPI**. Observera hur snabb mekaniken nu är. Tiden för stegning av huvudet spår till spår är endast 5ms mot de 40 ms som ursprungliga miniflex-specifikationerna krävde.

Handhavande

Det finns en del saker att tänka på då man arbetar med ett flexskivesystem. En viktig sak är att alltid se till att back-up finns, dvs en kopia av filerna som förändrats. Den bör göras tillräckligt ofta för att kopian skall kunna uppdateras någorlunda smärtfritt ifall olyckan råkar vara framme. Det här skall man inte slarva med. Smällen brukar komma när man minst anar det. För att förhindra oavsiktlig skrivning på skivan kan vi vänta med att "osäkra" den, dvs att stänga drivenhetens dörr. Öppen dörr hindrar att huvudet läggs an mot skivan. Först skall de andra delarna i datorsystemet startas upp. Alla nätspänningar skall vara tillslagna. På liknande sätt öppnas dörren innan man börjar stänga av datorn. När ett nytt assemblerprogram körs för första gången öppnas med fördel först luckan. Det är alldeles för vanligt att programmet "skenar" vid detta tillfälle. Även disketterna skall behandlas varsamt. Inte läggas löst på bordet, utan kvickt in i sitt fodral och ned i en box med lock. Akta den för starka

	<u>51</u>	<u>52</u>	<u>91</u>	<u>92</u>
Capacity Specifications (In 10³ bytes)				
Single Density (unformatted)				
Per Disk	125	250	250	500
Per Track.	3.13	3.13	3.13	3.13
Double Density (unformatted)				
Per Disk	250	500	500	1000
Per Track.	6.25	6.25	6.25	6.25
Functional Specifications				
Tracks	40	80	80	160
Track Density (TPI)	48	48	96	96
Recording Density (max. BPI)				
Single Density	2768	2938	2788	2961
Double Density	5536	5876	5576	5922
Rotational Speed (RPM)		300 ±1½%		
Average Latency		100ms		
Transfer Rate				
Single Density		125K bits/sec		
Double Density		250K bits/sec		
Encoding Method		FM, MFM, M ² FM		
Access Time				
Track to Track		5ms		
Average.	84ms (51/52)	150ms (91/92)		
Head Settling Time		15ms		
Head Load Time		35ms		
Power-Up Delay		0.5sec		
Interfacing.		Industry/ANSI compatible		

magnetfält. Se upp för nätstransformatorer och högtalarmagneter. Överdriven värme är inte heller bra. En flexskiva som låg i en väska i solen i en bil liknade mer en bubblig tunnpannkaka än ett data-medium. Med lite sunt förnuft i umgänget med flexskivan är den ett utmärkt lagringsmedia för hemdatoristen.

En eller flera drivenheter

Upp till fyra drivenheter brukar man kunna ansluta till samma anpassningskort. Vanligast är dock att man nöjer sig med två. Systemskivan kan då alltid sitta i drivenhet 0. I enhet 1 finns skivan man arbetar på. Det finns dock inga krav på att minst två enheter måste finnas. Det går alldeles utmärkt att leva med endast en. Vid kopieringar

mellan skivor används speciella program som mellanlagrar informationen i datorns primärminne. Är filerna små kan åtskilliga få plats i minnet samtidigt. Sedan skiftar man disketterna i den enda drivenheten och för upp filerna från minnet. Det är inte riktigt lika lättasamt som då två enheter finns, men knappast särskilt besvärligt. Till varje skiva som man skall arbeta med kopieras enbart de nödvändiga kommandona, exempelvis editor, assembler och printrutiner. Det sparar utrymme. Basic interpretatorn kan ligga på systemdisketten eftersom den kan läsas ned för sig. Operativsystemet är också resistent. Väl i minnet ligger det kvar tills strömmen slås av. Starta alltså med en drivenhet om ekonomin är skral. Komplettera senare om behovet uppstår. ■

Fig 7:3. Så här kan specifikationen för drivenheter för flexskivor se ut. De olika versionerna tillverkas av MPI.

RT-datorns uppbyggnad och möjligheter

AV ÅKE HOLM

Dubbla processorkretsar och kraftfulla expansionsmöjligheter kännetecknar RT-datorn.

■ RT-datorn är benämningen på en modulbyggd, mångsidig dator för självbyggare. Datorkonstruktionen är avpassad för både hobby och utvecklingsändamål och ger ett nästan obegränsat antal användningsområden tack vare möjligheten, att man själv kan bygga anpassningskort av skilda slag.

Som dator för nybörjare består RT-datorn av ett processorkort, ett litet terminalkort och ett mo-

derkort med tillhörande nätrel. En dator av detta slag kostar ca 2.000 kr att bygga och man får då en dator, som funktionsmässigt innehåller CPU med 6808, 1 k RAM och 1 k monitorprogram. Terminallen består av ett litet kretskort med sex lysdiodindikatorer och 20 tangenter för kommandon och hexadecimala tal. Det tillhörande monitorprogrammet (ECBUG) möjliggör inläggning av egna program i minnet samt felsökningsmöjligheter.

När man vill utöka datorns kapacitet, behöver man bara förse moderkortet med fler kortkontakter och i dessa plugga i minneskort, ACIA-kort, videoterminal, flexkontrollkort m m. En RT-dator utbyggd med 48 k RAM-minne, videoterminal och två flexskiveenheter kostar ca 13.000 kr exkl. moms. För denna summa får man en dator med egenskaper, som få av de färdiga datorerna för hobbybruk kan erbjuda.

Dubbla processorkretsar

Som mikroprocessor i RT-datorn kan man använda både den välkända MC6800 och den nya MC6809, av många ansedd som den kraftfullaste åttabitsars mikroprocessorn på världsmarknaden. Omkoppling mellan de båda sker med en vippomkopplare på CPU-kortet. Varför finns det nu två processorkretsar i RT-datorn? Svaret på det är att 6800 fortfarande är den vanligaste av de två och att den är relativt lätt att lära sig förstå och använda. 6809-processorn är kraftfullare och snabbare, men inte alls lika lätt att programmera "för hand", om man är nybörjare.

I fig 8:2 finns en minneskarta över RT-datorn. För att datorn skall vara kompatibel med andra hobbydatorer för 6800 är minnet disponerat på olika sätt för de båda processorerna. För 6800 finns möjlighet att ha 32 k kontinuerligt minne mellan 0000 och \$7FFF. Ovanför det finns ett område reserverat

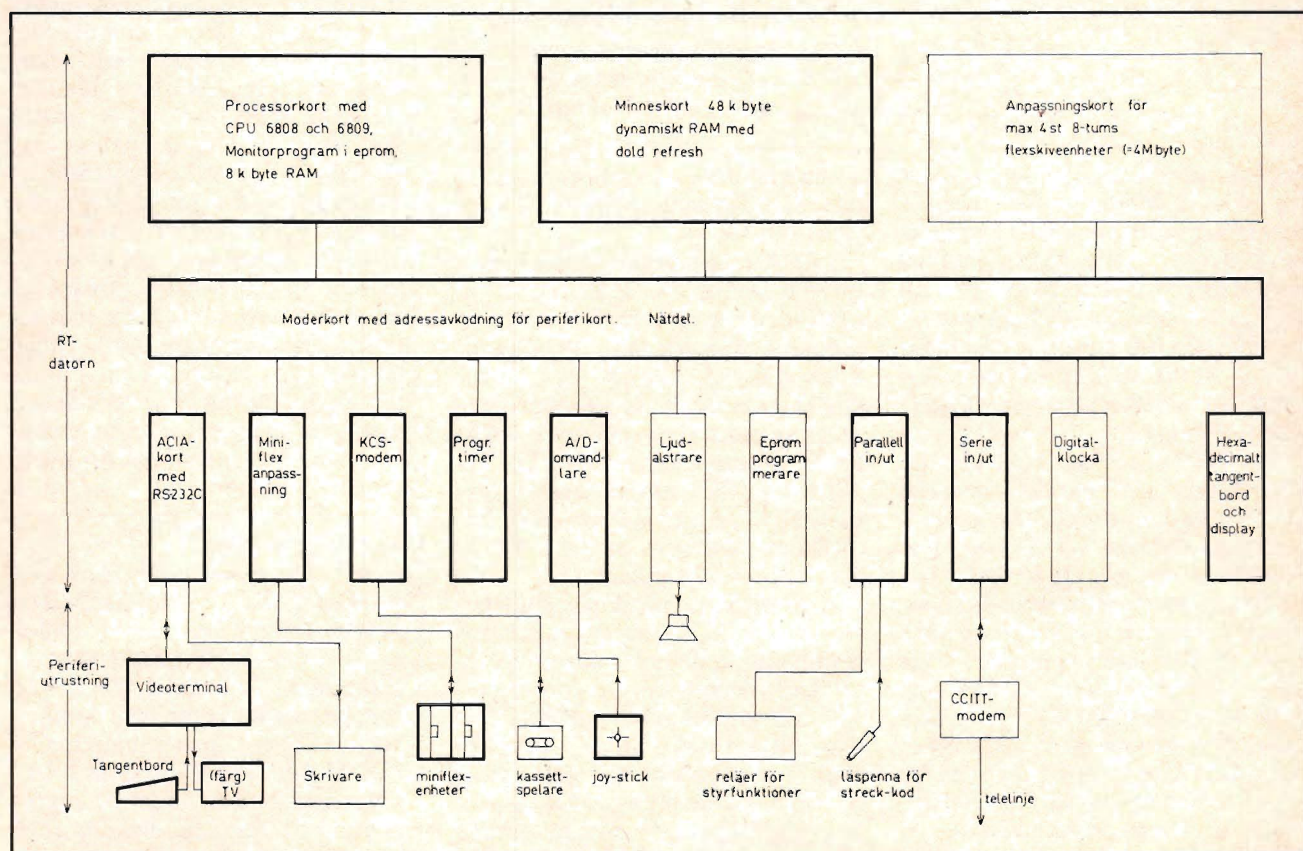
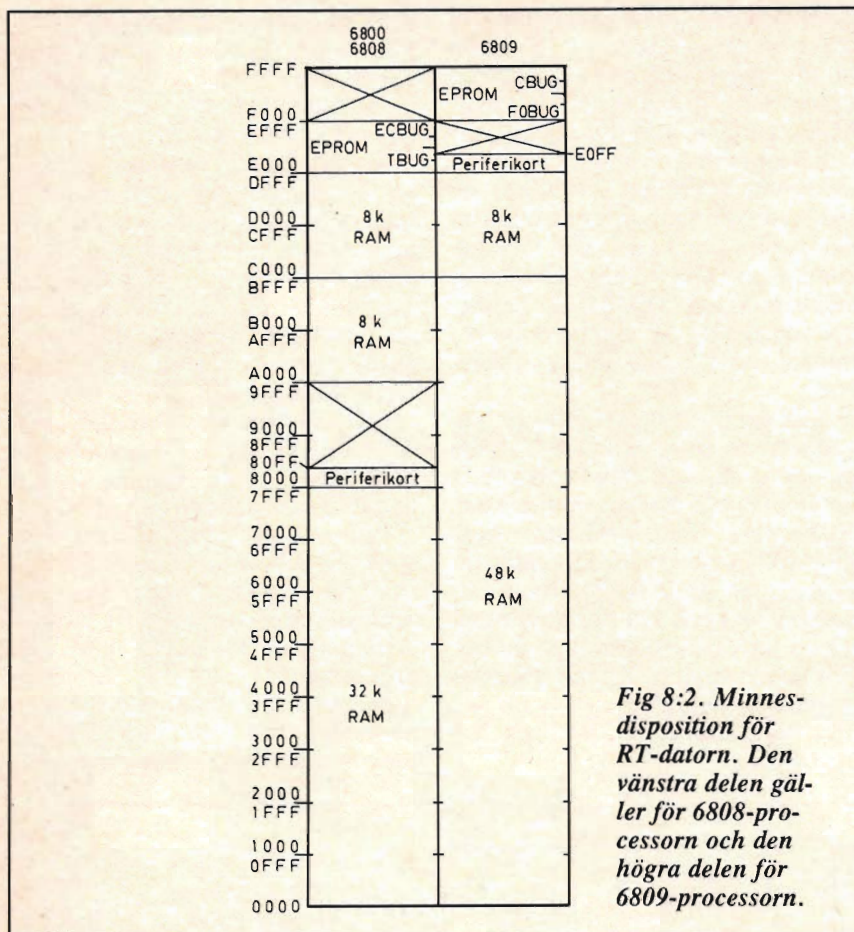


Fig 8:1. Blockschemat för RT-datorn. De med grova linjer inramade enheterna är beskrivna i byggboken.



för alla periferikretsar dvs kretsar, som förbinder datorn med yttvärlden. De korsade områdena används f.n. inte. Mellan \$A000 och BFFF finns 8 k minne och mellan \$C000 och DFFF går det att ha ytterligare 8 k minne. Området mellan \$E000 och EFFF är reserverat för monitorprogram (TBUG och EBUG).

För 6809-processorn ser det lite annorlunda ut. Periferikretsarna har flyttats upp till \$E000 och monitorprogrammen till \$F000. Det innebär att det går att ha 56 k kontinuerligt minne från 0000 och uppåt. Med större minne kan man köra mera omfattande program. Har man flexskivesystem, så används området \$C000 till DFFF av själva operativsystemet för 6809 eller området \$A000 till BFFF för 6800.

Att monitorprogram och periferikretsar flyttar sina adresser i minnesarean, behöver man inte tänka så mycket på, eftersom den-

na omkoppling sker automatiskt då man slår om omkopplaren på processorkortet.

I fig 8.1 återfinns blockschemat för RT-datorn. De delar som inramats med grova linjer är beskrivna i boken. Längst upp till vänster finns datorns hjärna, processorkortet med de båda mikroprocessorer samt vissa minneskretsar. I mitten finns minneskortet för data-lagring. Moderkort med nätdel kan sägas vara en del av själva datorlådan, eftersom dessa delar är fast monterade i lådan.

Åtta olika periferikort skall beskrivas. ACIA-kortet är ett kort för seriekommunikation som krävs då vi skall ansluta en videoterminal. Till videoterminalen ansluts ett tangentbord och en tv av lämpligt slag. Till ACIA-kortet kan också en skrivare anslutas. Miniflexanpassning kopplar samman datorn med en eller två flexskiveenheter vilka fungerar som snabba mass-

minnen för lagring av program och data. Till KCS-modemet kan vi ansluta en vanlig kassettspelare och spela in data på kassetter. KCS-modemet används ihop med kortet, vilket innehåller hexadecimalt tangentbord och display. Detta kort kan bl.a. användas som terminal till datorn. En programmerbar timer finner sin användning då man behöver räknare som kan styras för frekvensdivision m m. Till ett kort med parallella in/utgångar kopplar vi reläer eller kanske en läspenna för den allt vanligare streckkoden. Det går också att via ett modem för telefonledning ta kontakt med en annan dator. Modemet ansluts till ett periferikort med seriein/utgång. Analoga signaler kan anslutas via en A/D-omvandlare. Det kan till exempel vara signaler från en skjoy-stick, vilken ibland används i TV-spel för att styra tänkta rymdfarkoster m m.

Bland övriga tänkbara periferikort kan vi nämna D/A-omvandlare, digitalklocka, ljudalstring, epromprogrammerare, IEEE-busanslutning och labkort av olika typer. Har man behov av större massminne kan man ansluta upp till fyra 8-tums flexskiveenheter. Man har då tillgång till 4 Megabyte "on line". Med större skivminnen kan ca 16 Megabyte lagras och läsas med kort åtkomsttid. Dessa typer av minnen faller dock utanför hobbytillämpningarna (inte minst prismässigt), varför vi inte kommer att gå närmare in på dem i boken. De nämns dock som exempel på vad RT-datorn har för eventuella möjligheter.

Materialsatser

till RT-datorn har tagits fram av CÅ Elektronik. Försäljning sker genom Telko i Stockholm, Malmö och Göteborg samt av EH:s Elektronik, Västerås. Enstaka specialkomponenter som kretskort och programmerade PROM säljs även. För dem som vill göra sina kretskort på egen hand visas mönster i bokens appendix.

Några tips inför byggandet av RT-datorn

AV ÅKE HOLM

Datorbygge kräver en del eftertanke. Vi ger här några råd om montering och provning som du bör ta del av innan du påbörjar bygget.

■ De följande avsnitten innehåller byggbeskrivningar på olika delar av RT-datorn. Varje avsnitt innehåller principschema med funktionsbeskrivning, komponentförteckning, komponentplacering på kretskort och funktionskontroll.

För den som vill göra kretskortet själv, finns i bilagan i slutet av boken kretskortsbilder till alla datorskorten i skala 1:1 eller 1:2.

Det första goda rådet till den, som ivrigt kastar sig över en del av en datorbyggsats är: *ha inte för bråttom!* Det är så lätt att ett litet fel uppstår, ett fel som sedan kan ta lång tid att hitta.

Kontrollera mönsterkortet

När man skall montera ett kretskort, bör man först förvissa sig om det inte har några tillverkningsfel. Sådana är visserligen mycket sällsynta, men med en extra kontroll kanske felsökningen underlättas. Håll upp kretskortet mot en lampa och se om det är avbrott på någon ledare eller om det är överbyggnad mellan två närliggande ledare. Det sista bör man kontrollera extra noga på ledare, som passerar mellan stiften på IC-kretsarna.

En kortslutning mellan två ledare på kortets översida kan vara svår att lokalisera, om den är dold under en IC-hållare.

Gå igenom komponenterna

När man förvässat sig om att kortet är felfritt, bör man kontrollera de

till kortet hörande komponenterna med stycklistan. På grund av varierande tillgång på olika komponenter kan det ibland hända, att vissa komponenter ersatts med andra funktionsekvivalenta med annan beteckning. IC-kretsar, vilka i stycklistan angivits som LS-TTL, kan vara ersatta med standard-TTL och saknar då bokstäverna LS i typbeteckningen. Olika tillverkare har dessutom olika prefix till dessa typnummer, beroende på vilken kåpa och vilket temperaturområde IC:n är avsedd för.

Val av kondensatorer

På kondensatorer, speciellt de som används för *avkoppling av matningsspänningen*, kan vissa avvikelser förekomma från det nominella värdet. I stället för 4,7 uF kan man använda 2,2 uF, 6,8 uF eller 10 uF. Värdena är i dessa fall inte särskilt kritiska.

Kondensatorer för *tidsbestämmande kretsar* bör vara av styrol- eller polyelektrolyttyp, med låg tolerans.

När man har bekantat sig med alla komponenterna är det dags att börja löda in dessa på kortet. De kretskort som ingår i byggsatser är försedda med komponenttryck för att underlätta placeringen av de olika komponenterna.

Börja med byglar

Man bör alltid börja med de minsta komponenterna. På vissa kort skall en eller flera byglar inlödås. Dessa är på komponenttrycket markerade med heldragna linjer mellan två hål. Alla sådana byglar skall lödas in allra först. Därefter löder man in samtliga dioder, motstånd och kondensatorer. Det är viktigt att alla polariserade kondensatorer tantalelektrolyter m fl) vänds åt rätt håll. En felvänd tantalkondensator fungerar som en kortslutning med onormalt hög strömförbrukning som följd. Om polaritetsmärkningen saknas eller är otydlig kan ledningsmönstret följas upp och polariteten listas ut. IC-hållare bör användas till alla kretsar. De har en markering i det hörn där stift 1 skall placeras. Denna markering

skall vändas åt samma håll som markeringen på kretskortet.

Sist monteras transistorer, kontakter och omkopplare.

Kontrollera lödningarna

När kretskortet är färdiglött bör man se efter, att ingen lödning är glömd eller att inget extra lödtenn förorsakar någon oönskad förbindelse.

När allt verkar vara i sin ordning kan IC-kretsarna sättas in. Vissa kretsar måste först benen bockas på så att båda stiftraderna är parallella. Det kan vara svårt att få i kretsen i hållaren annars och risken för att ett stift skall vika sig minskas också. Kretsar med 24 eller fler stift kan ibland vara svåra att få ner i sina hållare. Lagg då kretskortet på ett plant underlag och tryck med jämn kraft över hela kretsen, så att inga brytningskrafter uppstår.

Prov i datorn

Då kortet skall provas placeras det i datorn på angiven plats. Det finns för de flesta beskrivningarna en funktionskontroll, som bör genomföras. Skulle något inte fungera enligt beskrivningen bör man kontrollera om någon komponent är felvänd eller om någon omkopplare står i fel läge. Det kanske är testprogrammet som är fel inslaget eller något annat yttre fel som gör att det inte fungerar.

Komponentfel är sällsynta. Det vanligaste felet är monteringsfel eller dåliga lödningar. Skulle ändå inte felet hittas, kanske det finns någon annan, som kan titta på bygget och kanske hjälpa till att få igång det.

Stift för spänningsmatning till kretsarna är ofta utlämnade, men de framgår ur datablad.

Till sist vill vi bara påminna om att hembyggda nätanslutna apparater av detta slag ej är provningspliktiga för S-märkning. Man får dock inte sälja eller ge bort bygget till annan person, utan den som byggt ihop sin dator måste behålla den för gott.

Lycka till med byggandet! ■

Datorlådan, moderkort och nätdel

AV ÅKE HOLM

Det kan vara lämpligt att som första etapp i datorbygget starta med själva lådan och tillhörande nätdel och moderkort.

■ Det är lämpligt och praktiskt att börja med nätdel och moderkort till datorn innan de övriga korten monteras. Man har då någonstans att stoppa i korten, vartefter datorbygget framskrider. RT-datorns låda innehåller en nätdel som kan lämna +5 V/10A för IC-kretsarna, +12 V/4 A för flexskiveenheter och vissa IC-kretsar samt -12V/1A till EPROM-kretsar och dynamis-

ka minnen. I lådan finns ett moderkort med upp till fyra 86-poliga kortkontakter avsedda för CPU-kort, minneskort och terminalkort. På moderkortet finns också plats för maximalt femton 22-poliga kortkontakter för olika periferikort. Hur många av dessa kontakter man bör löda in, beror på vilka planer man har för datorns funktion.

Om man vill börja med det enklaste systemet räcker det med en kontakt av varje sort. Vill man däremot bygga för full minneskapacitet och flexskivesystem behövs tre 86-poliga och minst tre 22-poliga kontakter.

Lådans frontpanel

På lådans front finns längst ner till vänster nätströmbrytare med kontrolllysdiod. Bredvid denna finns två tryckknappar, en för restart (återstart) och en för NMI-avbrott (programavbrott). Ovanför dessa kan två flexskiveenheter monteras. Börjar man med det enkla sys-

temet kan hål tagas upp för indikatorkortets tangenter och lysdiodindikatorer till höger om flexskiveenheterna (se omslagsbilden).

På lådans baksida finns nätkabelintag, säkringshållare, kontakt för anslutning av skrivare samt en slits för kabeln till tangentbord och TV-monitor. Om datorn skall användas för programutveckling och provning av olika periferikort behövs ingen fläkt eftersom lådans lock mest är avtaget. Skall locket sitta på för jämnare ventilationshål tas upp i lock och bakstycke för att inte arbetstemperaturen skall bli för hög.

Lådan är av enkel typ och den består av en U-böckad aluminiumplåt, vilken tjänstgör som baksida, botten och front. Lådan fungerar även som kylfläns för likriktare och stabilisatorkretsar.

I fig. 10:1 finns måttskiss för håltagning i fronten på lådan. Hålen för flexenheterna är noggrannt avpassade för att utnyttja lådan på bästa sätt. Om man börjar med det

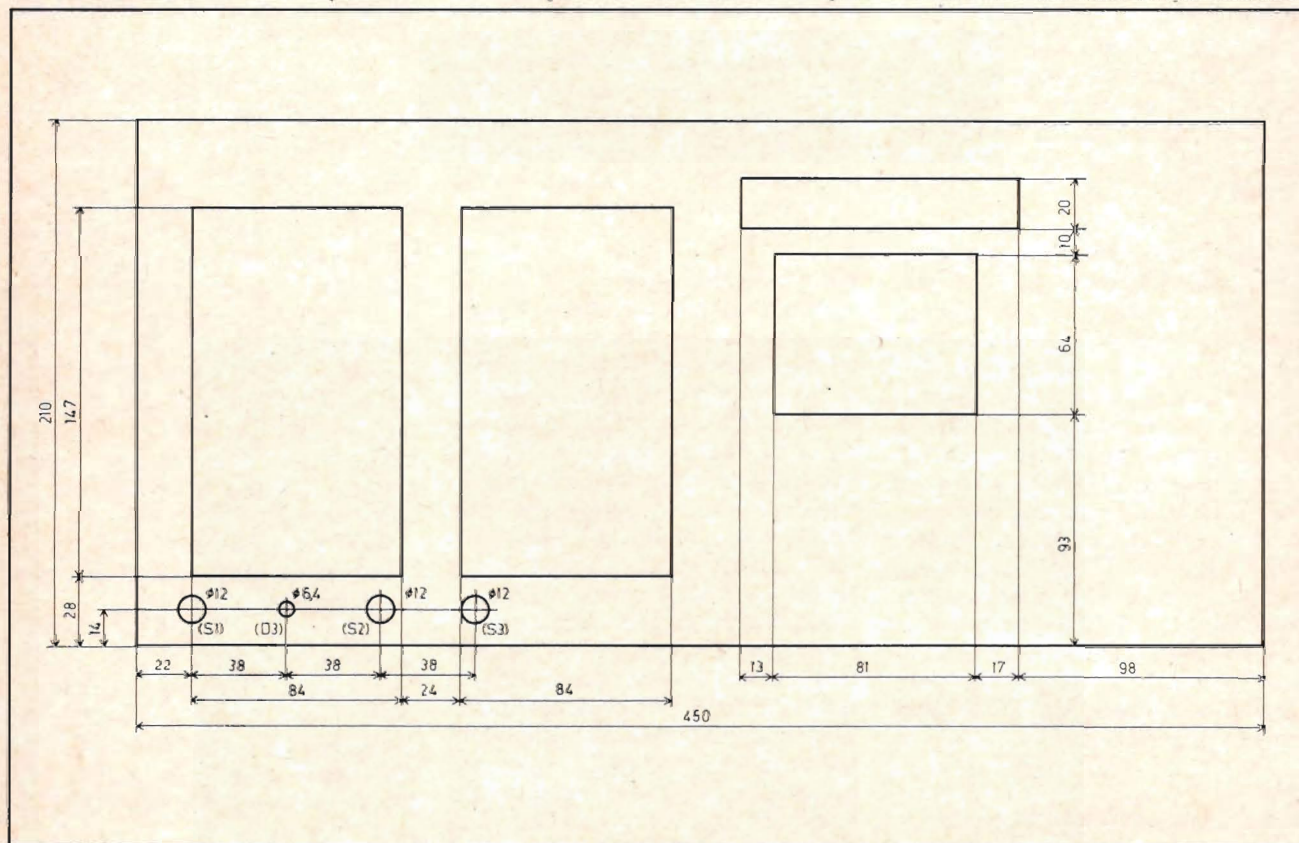


Fig 10:1. Måttskiss för hålen i datorlådans front.

minsta systemet, men inom kort tänker utöka med en eller två flexskiveenheter, kan det vara bekvämt att ta upp hålen för dessa på en gång. Måtten på hålen i botten (se fig 10:2) är avsedda för fastsättning av en monteringsplåt för flexskiveenheterna och för moderkortet. För att förenkla uppmärksningen av hålen för moderkortet kan man förfara på följande sätt. Märk upp de två hålen i lådans högra kant. Lägg därefter det omonterade kretskortet med komponentsidan mot lådans botten och märk sedan de övriga fästhålerna genom kretskortets hål.

Innan man sedan börjar borra kan det vara bra att en extra gång förvissa sig om att alla hål kommer på rätt plats.

På lådans baksida skall bland annat en stabilisatorkrets monteras. För att förbättra kylningen av denna skall färgen under den skrapas bort, innan den fastskruvas. I de olika måttskisserna finns inom parentes olika komponentnummer

angivna för att förtydliga var de olika delarna skall monteras.

I fig 10:4 finns måttuppgifter för den monteringsplåt, på vilken de båda flexskiveenheterna skall monteras. Denna plåt är även kylplåt för IC11 och IC12 samt fäste för C6 och C8.

Montering av lådan

När alla erforderliga hål är gjorda i lådan monteras S1, S2, S3 samt D3 i fronten. Skruva fast dragavlastningen för nätkabeln samt säkringshållaren i bakstycket. Koppla in nätkabeln till S1 och koppla den gulgröna skyddsledaren till ett lödöra, som fästs med skruv vid det främre vänstra fästhålet i botten på lådan. Montera därefter moderkortet och löd (på kortets baksida) in de två grova ledningarna (röd och svart) för +5V och jord. Förbind även på kortets undersida de två punkterna för +5V. Ledningarna skall vara så långa att de räcker fram till IC10. Löd in en gul ledning till +12V-punkterna och en

blå ledning till -12V-punkterna. Anslut fyra ledningar (röd, svart, gul och grön) till kretskortets hål för S2, S3 och D3.

Montera erforderligt antal kortkontakter med tillhörande styrgejdrar. Dessa fästs med speciella skruvar (UNF 4-40×3/16). På grund av att dessa gejdrar är gjutna i plast kan det hända att spåren på vissa är i smalaste laget för de dubbelzijdiga kretskorten. Man bör därför fila upp dessa spår med en tunn fil innan gejdrarna skruvas fast. Detta gäller framförallt gejdrarna till kontakterna J16–19 samt J2. När moderkortet är monterat kan det skruvas fast i lådans botten. Det monteras med 8 mm distanser. Koppla därefter in S2, S3 och D3 till de fyra ledningarna. R16 löds in mellan D3 och jordstiftet på S2. Lägg in den röda och den svarta ledningen till IC10. Den svarta ledningen skall lödas till ett lödöra på IC10:s ena fästskruv. Koppla samman IC10 och D10. Löd in säkringshållaren och skruva

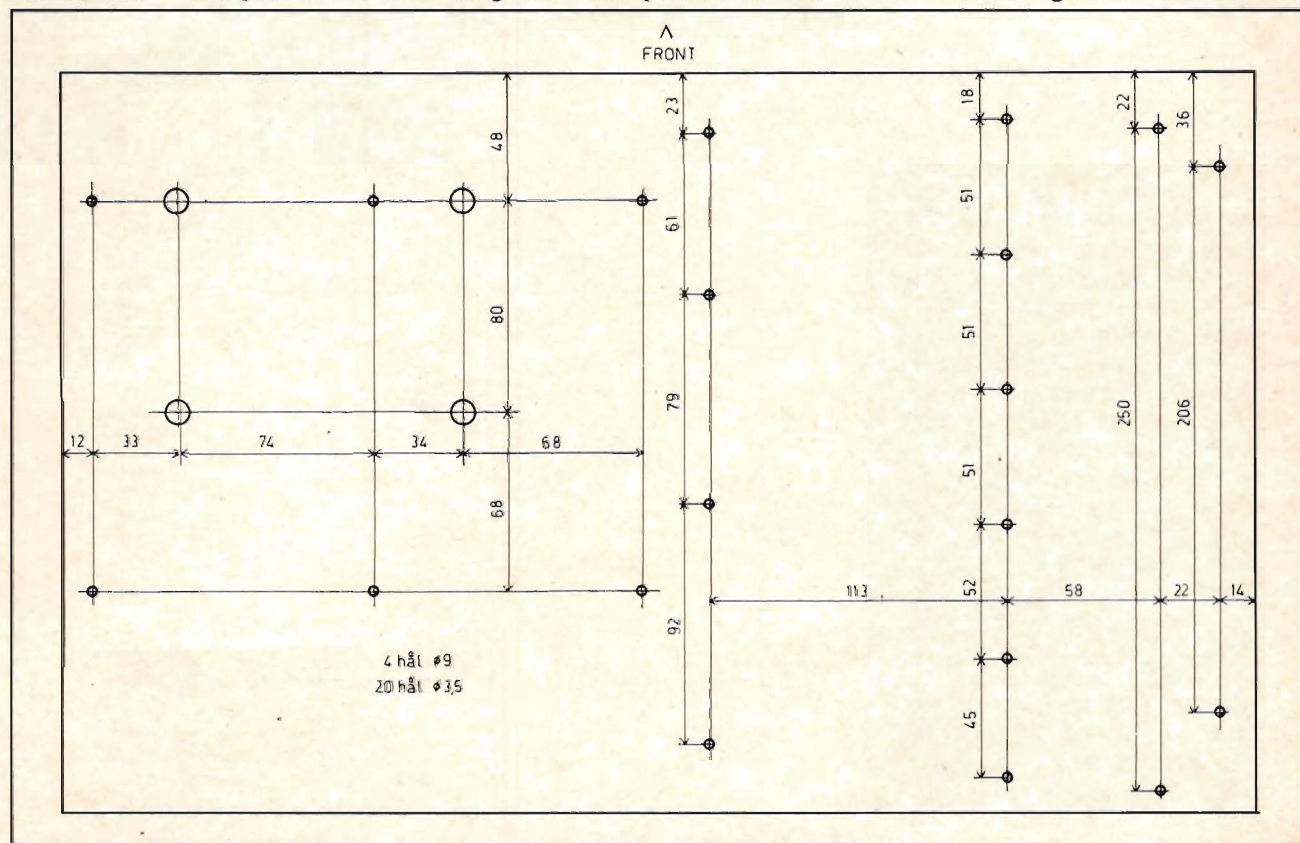


Fig 10:2. Måttskiss för hålen i datorlådans botten.

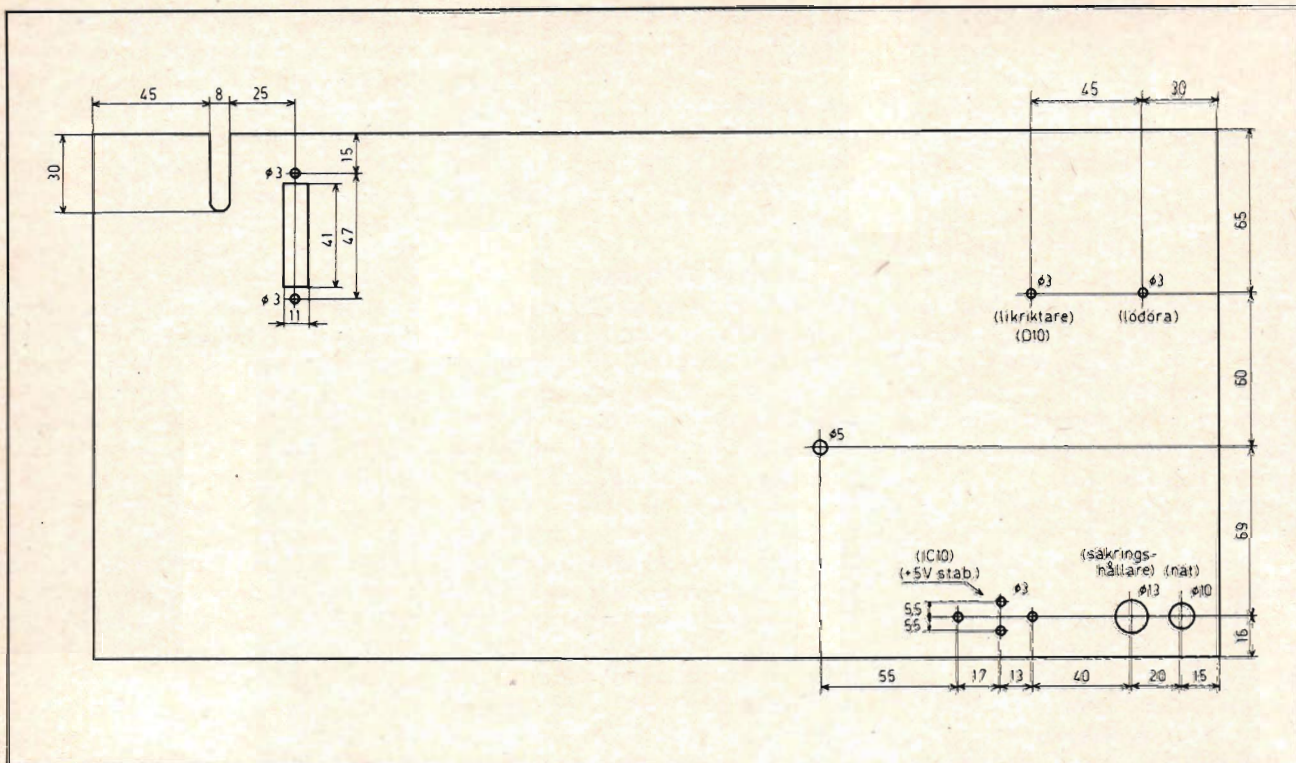


Fig 10:3. Måttskiss för hålen i datorlådans baksida.

fast transformatorn i bakstycket med M5-skruven och brickorna.

På monteringsplåten skruvas IC11 och IC12 fast. IC11 skruvas fast från undersidan, så att anslutningarna blir åtkomliga när plåten är monterad i lådan. IC12 skall förses med isoleringsbricka och isolernippel. Mellan IC12 och lödörat kopplas C10 samt dioderna D4-7 in. Montera C6 och C8 i de stora hålen. Plåten fästs i lådans botten med sex 30 mm skruv. För att justera höjdläget på plåten används dubbla muttrar, en på varje sida om plåten. Höjden regleras så att översidan på plåten kommer i samma höjd som öppningen i fronten för flexskiveenheter.

Koppla samman D10 med C6. Löd in transformatorns båda 8V lindningar till D10. Se till att de läggs parallellt, i annat fall blir det en rejäl kortslutning. De båda ledningarna från transformatorns 14V/1A-lindning löds in till D4-7. Fäst ett lödöra i hålet bredvid D10 och koppla in de fyra dioderna D8, 9, 11 och 12 som *fig 10:5* visar. Koppla därefter ihop C8 med IC11

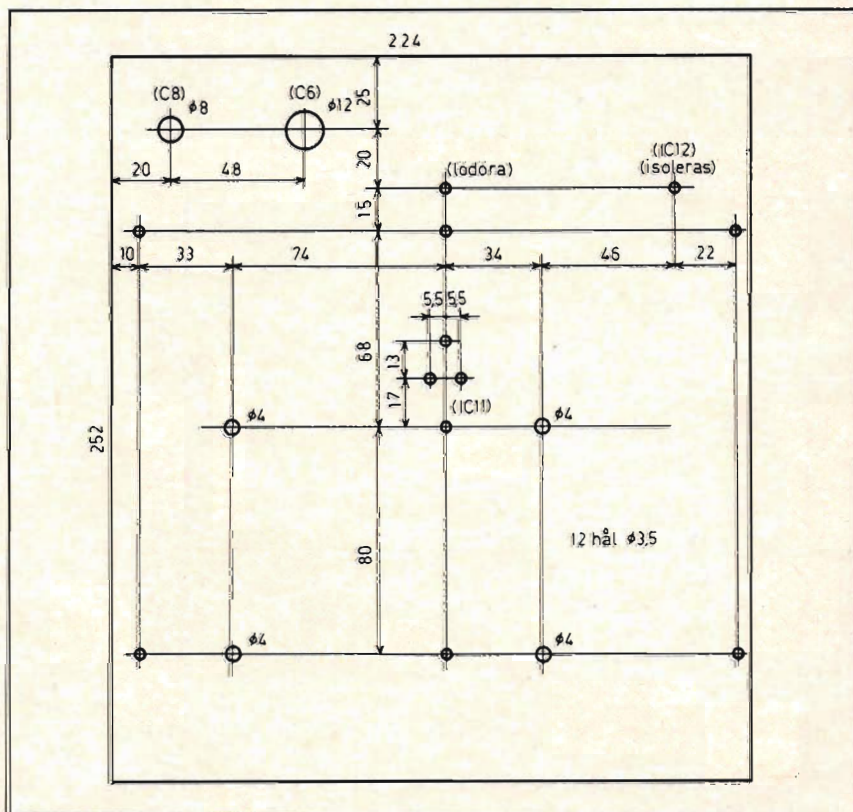


Fig 10:4. Måttskiss för hålen i monteringsplåten.

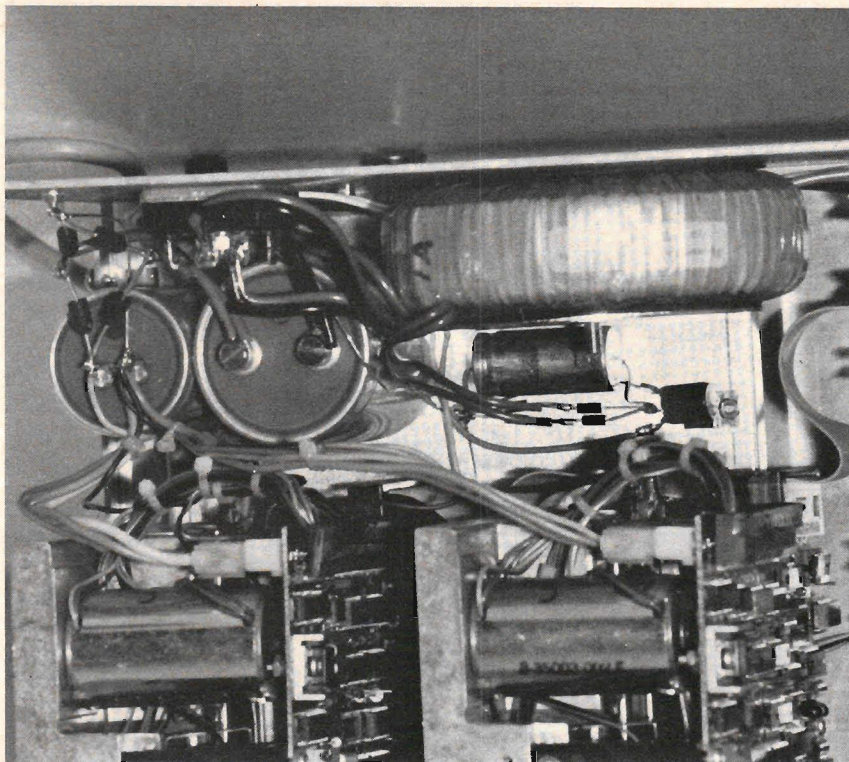


Fig 10:5. Bilden visar hur nätdelen är monterad i lådan.

och anslut den gula och den blå ledningen från moderkortet till IC11 resp. IC12. Kondensatorerna C7, C9 och C11 skall monteras direkt på resp. IC. Om flexskiveenheter skall ingå i datorbygget, skall kontakterna för spänningsmatningen till dessa anslutas enligt fig 10:6. Dessa kontakter är av klämtyp och ledningarna behöver därför inte lödas fast.

Nu när allt är sammankopplat kontrollerar man alla förbindelser och sätter in en säkring i säkringshållaren. Med en voltmeter kontrolleras att de stabiliserade spänningarna håller de angivna värdena.

Moderkortets funktion

Principschema för moderkortet visas i fig 10:7. De fyra 86-poliga kontakterna J16–19 är parallellkopplade och återfinns till vänster i schemat. De femton 22-poliga kontakterna för periferikort är parallellkopplade så när som på ett av stiften och återfinns till höger på

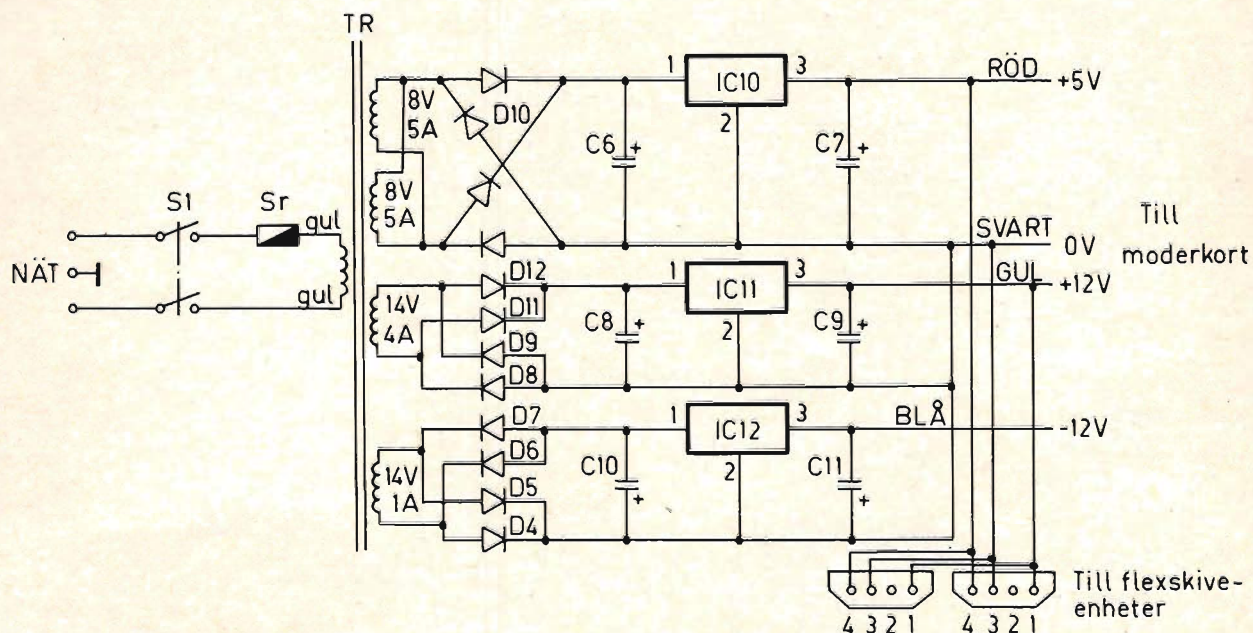


Fig 10:6. Principschema för nätdelen i RT-datorn.



44

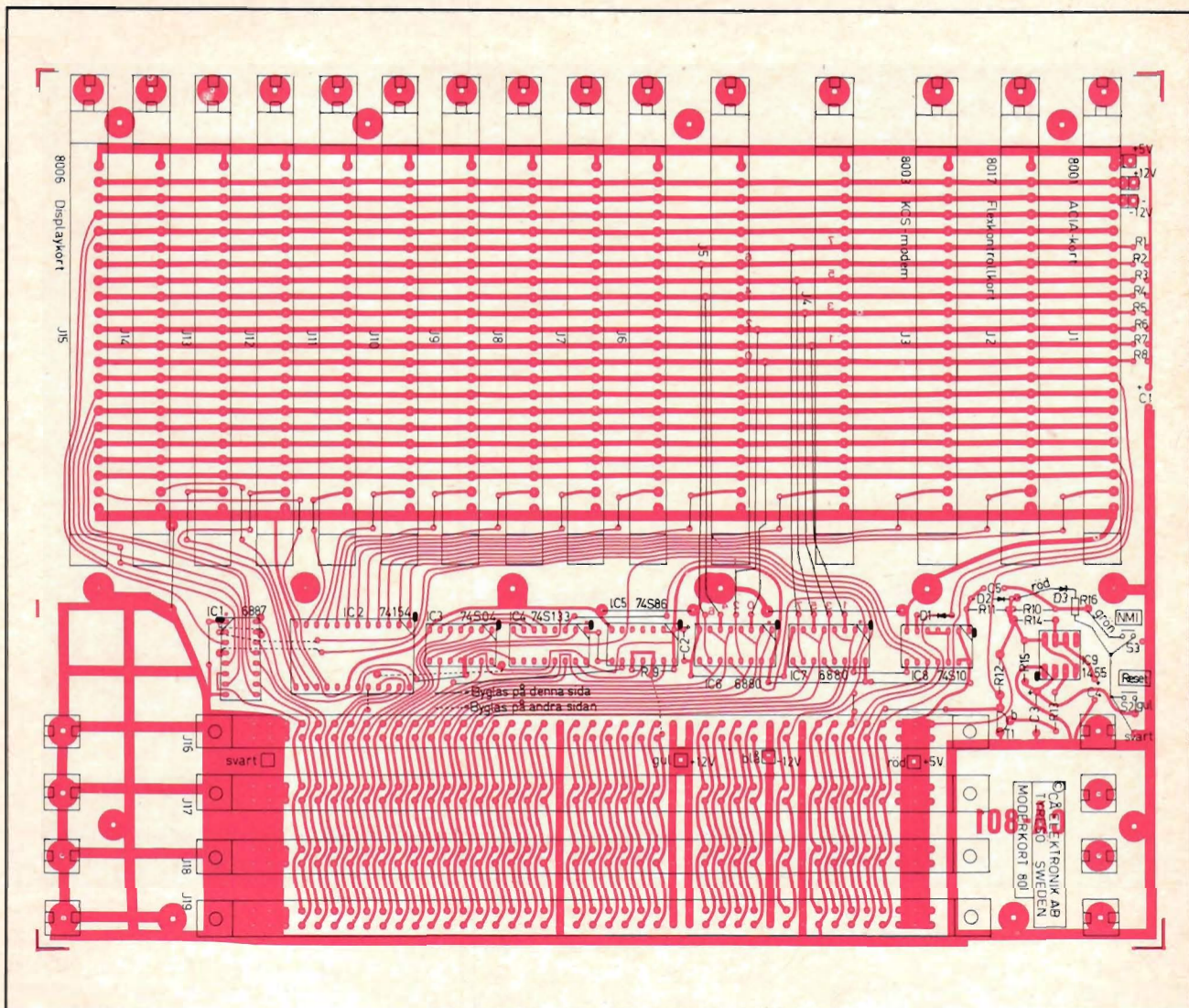
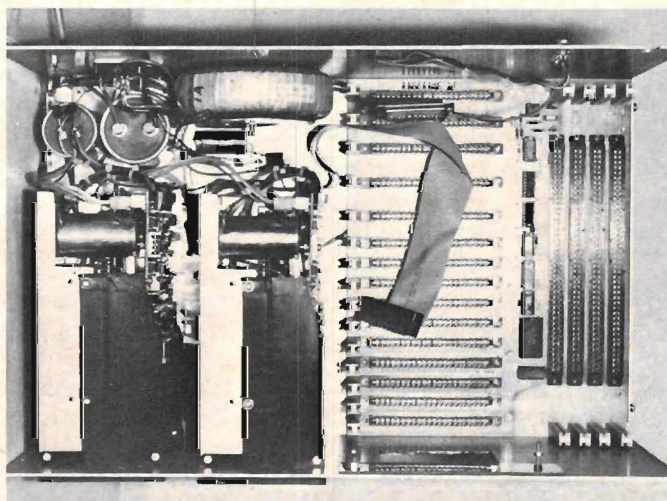


Fig 10:8. Komponentplaceringen på moderkortet. Ledningsmönstret visas i färg och komponenttrycket i svart.

Fig 10:9. Bilden visar datorlådan med moderkort, nätdel och två flexskivenheter.



schemat. Kretsarna på moderkortet har följande funktioner. IC3–5 avkodar de åtta högsta adresslederna (A8–A15) och kopplar in dekodern IC2 vid rätt adresser. IC5 är en exklusiv-eller grind och styrs med en signal från CPU-kortet via stift 18 på den 86-poliga busskontakten. När detta stift har hög nivå kommer utgången på IC4 att ha låg nivå på alla adresser mellan \$8000 och \$80FF. Avkodning sker sedan i IC2, så att det första periferikortet (anslutet till J1) får de sexton adresserna \$800–800F, det andra kortet \$8010–801F, o s v.

Denna avkodning används när

Komponentförteckning till datorlåda med nätrel och moderkort

C1-2	10 µF 16 V tantal
C3	0,22 µF tantal
C4	0,47 µF tantal
C5	0,1 µF pol
C6	47000 µF 16 V
C7	100 µF 6 V
C8	22000 µF 25 V
C9, 11	47 µF 16 V
C10	2200 µF 25 V
D1-2	1N4148
D3	gul lysdiod
D4-7	1N4002 el likn.
D8-9,	
11-12	1N5401 el likn.
D10	BYW21 el BYW61
IC1	6887 eller 6885
IC2	74154
IC3	74S04 el 7404
IC4	74S133
IC5	74S86 el 7486
IC6-7	MC 6880, 8T26 el

14	skruvar ECS 3×15
1	skruv ECS 3×20
6	skruvar ECS 3×30
1	skruv ECS 5×50
40	muttrar M3
1	mutter M5
14	distansrör 8 mm
23	kortgejdrar
23	skruv för d:o
1	monteringsplåt
1	säkringshållare
1	dragavlastning
1	nätkabel
1	nätkontakt med
	jorddon
1	låda
1	fästnippel för D3
4	lödöron
	kopplingstråd

eventuellt erfordras även:

1	rött plastfilter
1	plastskiva
2	skruvar ECS 3×15
6	muttrar M3
1 el. 2	DC-kontakter för
	flexenheter

	75136
IC8	74S10 el 7410
IC9	555
IC10	78P05
IC11	78H12
IC12	7912
J1-15	22-pol. kortkontakt
	0,156-deln.
J16-19	86-pol. kortkontakt
	0,156-deln.
R1-12	4,7 k 1/8W 5 %
R13	2,2 k
R14-15	1 M
S1	nätströmbrytare
	(Telko J152)
S2-3	1-pol. tryckknapp
	(Telko J195)
Sr	säkring 1A trög
T1	BC 548 el likn.
Tr	transformator
	CÄ-9288,
1	kretskort CÄ-801
1	IC-hållare 8-pol.
3	IC-hållare 14-pol.
4	IC-hållare 16-pol.
1	IC-hållare 24-pol.
7	skruvar ECS 3×8

man arbetar med 6808-processorn. Där är det vedertagen standard att in/utportar skall ligga på dessa adresser. En ACIA för terminalanslutning skall exempelvis ha adress \$8008.

Vid upprättande av monitorprogram för 6809-processorn har det blivit en annan planering av det tillgängliga adressområdet. Detta innebär, att området \$0000-DFFF reserverats för arbetsminne (RAM), området \$E000-E0FF för periferikretsar samt området \$F000-FFFF för monitorprogram i ROM eller EPROM. När CPU-kortet kopplas om för 6809-processorn får stift 18 på busskontakten låg nivå, varvid adressledarna A13 och A14 inverteras före avkodning. Utgången på IC4 kommer därvid att bli låg för alla adresser mellan \$E000 och \$E0FF i stället. Den ACIA som i 6808-fallet hade adress \$8008 har nu automatiskt fått adress \$E008. Genom denna

kretslösning har alla de femton kortkontaktarna J1-15 fått ändrade adresser.

Eftersom det i 6808-fallet inte får finnas RAM på adress \$8000, används stift 18 på den 86-poliga busskontakten till att även koppla in och ur minne på dessa adresser.

IC6-7 är databuffertar som styr dataflödet till och från de 22-poliga kontaktarna med hjälp av IC8. IC1 är en buffertkrets för de fyra lägsta adressledarna samt för skrivpuls och klockfrekvens till samtliga periferikort. Den ordning som signalerna har på den 22-poliga bussen har omsorgsfullt valts ut för att ge en enkel kretskortlayout mellan kontakten och periferikretsar av typ PIA, ACIA och PTM.

Avbrottsfunktion

Längst ner på schemat finns en vippa uppbyggd med IC8. S3 är en återfjädrande tryckknapp. Då stift 6 går till låg nivå, åstadkommes ett

NMI, d v s ett icke maskerbart avbrott, till mikroprocessorn. Monitorprogrammen TBUG och CBUG är utförda så att ett NMI avbryter det pågående programmet, skriver ut registerinnehållen samt återgår till monitorprogrammets kontrollslinga. Denna möjlighet till programavbrott kan vara praktisk exempelvis om ett program skenar iväg eller fastnar i en slinga. Med ett tryck på S3 avbryts programmet och man kan med ledning av registerinnehållet få tips om vad som blivit fel. Skulle ingenting hända vid tryck på S3, har programmet skenat iväg och förstört (=ändrat innehåll) i periferikretsarnas kontrollregister. Det enda sättet att komma tillbaka är då att trycka på resettangenten S2. Ett tryck på denna knapp ger med IC9 en startningspuls på ca 0,5 sekund. C4 har till uppgift att ge en puls vid nättillslag (automatisk start).

När datorlådan nu är monterad och provad är det bara att fortsätta med nästa del av datorbygget. ■

Processorkortet- datorns hjärna

AV ÅKE HOLM

RT-datorns processorkort har plats för två mikroprocessorer 8 k RAM samt upp till 4 k monitorprogram för varje processor.

■ Hjärnan i datorn är naturligtvis själva mikroprocessorn. I detta datorbygge har vi två mikroprocessorer, vilka tillsammans med ett antal minneskretsar är monterade på ett kretskort kallat CPU-kort. CPU står för *Central Processing Unit*. Dessa två processorkretsar är en MC6809 och en MC6808. Den se-

nare är helt enkelt en 6800 vilken har försetts med inbyggd klockfrekvensoscillator. Att kortet försetts med båda dessa kretsar har flera orsaker:

En är att det ännu så länge inte finns lika stort utbud av maskinkodsprogram för 6809 som det finns för 6800. En annan orsak är att de som är programutvecklare troligtvis vill kunna utveckla program för båda dessa processorer. Med den lösning av dubbla processorkretsar som bygget innebär får man maximal flexibilitet för alla tänkbara applikationer.

RAM och ROM finns på kortet

Principskemat för CPU-kortet visas i fig 11:1. Förutom de båda processorkretsarna finns det plats för upp till 8 k byte RAM-minnen

av typen 2114. Dessa minnen kan adresseras till \$A000-BFFF eller till \$C000-DFFF genom enkel bygling på kortet. Har man bara 6808-processorn, behöver man RAM från \$A000 och uppåt. Alla 8 k behövs om flexkivesystem med FLEX 2.0 skall användas. Har man endast 6809-processorn behövs minst 1 k RAM mellan \$DC00 och DFFF och 8 k mellan \$C000 och \$DFFF, om flexkivesystem med FLEX 9.0 skall användas. Vill man kunna använda både FLEX 2.0 och 9.0, krävs 8 k RAM på ett separat minneskort. Totalt 16 k erfordras då mellan \$A000 och \$DFFF. Vidare måste man naturligtvis ha RAM-minne från adress 0 och uppåt. Hur mycket man behöver beror på programstorlek, men 16 k är nog ett minimum.

För monitorprogram finns 8

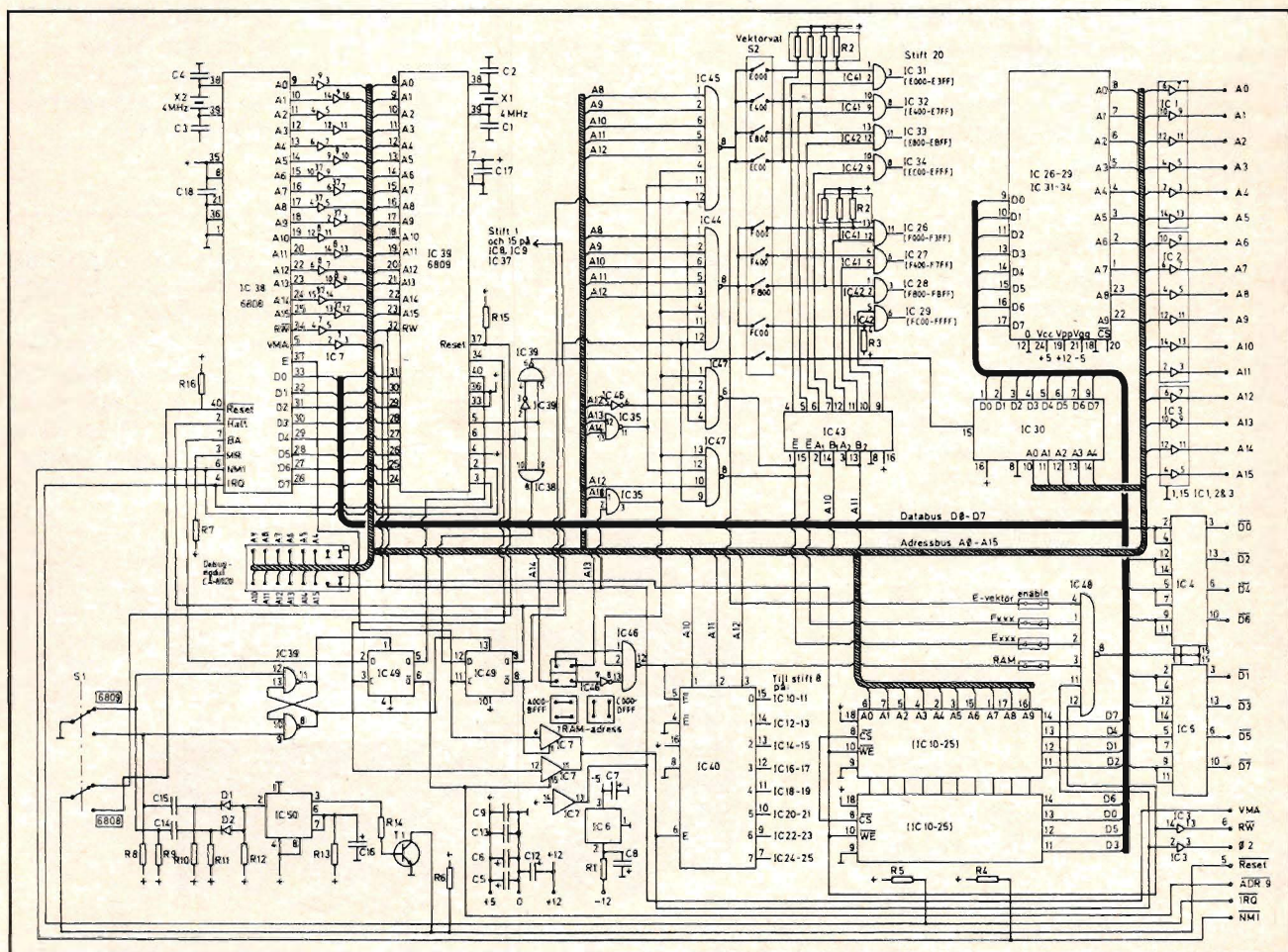


Fig 11:1. Principskemat för CPU-kortet till RT-datorn.

socklar avsedda för eprom av typ 2708. Fyra av dessa täcker området SE000-EFFF och kopplas in i läge 6808. De övriga fyra täcker området SF000-FFFF och kopplas in i läge 6809. Genom den automatiska omkopplingen av monitorprogram och periferiadresser uppstår det ingen risk för kollision mellan program för 6808 och 6809. Ledningen ADR 9 på busskontakten kopplar om avkodningen för periferikorten så att dessa ligger på adresserna S8000-80EF för 6808 och på SE000-E0EF för 6809. Med omkopplaren S2 kan man välja från vilken eprom-krets respektive processor skall hämta sina restart- och interruptvektorer. De övriga kretsarna har hand om avkodningen av adresserna på kortet. Det finns fyra byglar för urkoppling av minnena på kortet. Dessa byglar kopplar om så att de adresser som minneskretsarna täcker adresseras inom kortet. Detta innebär exempelvis att om man inte vill ha minneskretsarna SA000-BFFF på CPU-kortet, så skall bygeln RAM och de 16 kretsarna IC10-25 tagas bort.

Automatisk återstart då processorerna växlas

Omkopplaren S1 är kopplad så, att varje gång man byter processor alstras automatiskt en restart-puls. På så sätt undviker man att processorn skenar iväg och kanske förstör det som står i dataminnenet. IC50 är en monostabil vipa som alstrar restartpuls. IC39 är kopplad som en RS-vipa vilken eliminerar kontaktstuds från S1. D-vippan IC49 styr omkopplingen mellan de båda processorkretsarna på följande sätt:

När omkopplaren S1 ställs i läge 6808, går utgången 5 på IC49 låg och HALT-ingången på IC39 stoppar processorn. En låg nivå på halt-ingången innebär att processorn slutför pågående instruktion innan den signalerar Bus Available. BA-signalen går till ingång 12 på IC49, och vid nästa positiva flank på stift 11 går utgång 9 hög och släpper halt-funktionen hos IC38. Det omvända förhållandet

gäller vid skifte mellan 6808 och 6809. Genom detta förfarande kopplas bara en processor i taget in på adress- och databussen. Signalen VMA saknas från 6809 eftersom denna, tack vare annan intern funktion, lämnar SFFFF ut på adressbussen då ingen access till

minne sker. VMA kan därför läggas på konstant hög nivå vid 6809-drift. Detta sker via IC7 stift 14.

IC-kretsarna IC44-46 avkodar adresserna till minneskretsarna på kortet. IC40 kodar ut de åtta paren RAM-minnen, och IC43 kodar ut de två 4-grupperna eprom-kretsar.

Komponentförteckning

C1-4	22 pF keram skiv	IC50	MC 1455
C5-6	47 µF 25 V el lyt	R1	12 ohm 1W
C7-12	10 µF 16 V tantal	R2	7×4,7 k motst nät
C13-15,		R3-9	4,7 k
17-18	0,1 µF pol	R10-11,	
C16	0,47 µF tantal	15-16	22 k
D1-2	IN4148	R12	100 k
IC1-3	MC 6887 el 6885	R13	1 M
IC4-5	MC 6880 el 8T26	R14	2,2 k
IC6	7905	S1	2-pol omkopplare för kort
IC7	MC6887 el 74367	S2	8-pol DIP-omkopplare
IC8-9, 37	74LS367		
IC10-25	2114	T1	BC 548
IC35,		X1-2	kristall 4 MHz
41-42	74LS08	1	kretskort CÅ-6819
IC 36	MC 6809	1	skruv ECS 3×5
IC 38	MC 6808	1	mutter M3
IC39	74LS00	1	IC-hållare 8-pol
IC40	74LS138	10	IC-hållare 14-pol
IC43	74LS139	11	IC-hållare 16-pol
IC44-45,		16	IC-hållare 18-pol
48	74LS30	8	IC-hållare 24-pol
IC46	74LS10	2	IC-hållare 40-pol
IC47	74LS20	6	kortslutningsbyglar
IC49	74LS74	6	stiftpar till dito

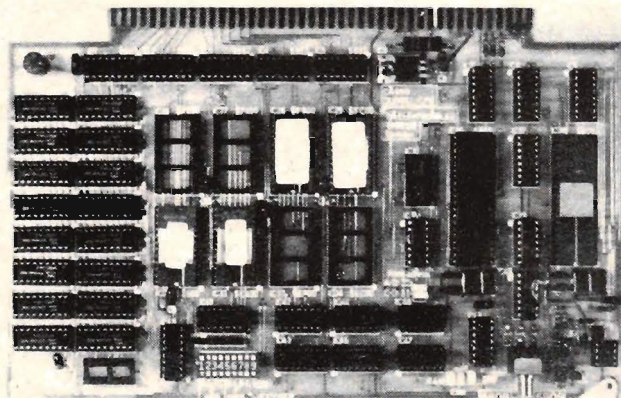


Fig 11:2. Bilden visar det monterade CPU-kortet bestyckat med båda processorkretsarna, 8 k RAM och monitorprogram i EPROM.

Enkel terminal för maskinkod- programmering

AV ÅKE HOLM

Den här terminalen arbetar med hex-kod. Kombinerad med CPU-kortet ger den ett billigt alternativ att starta med.

Programmeringen sker i maskinkod från ett tangentbord och lysdiodindikatorer visar adress och data.

■ Indikatorkortet skall placeras i kontakten J15 på moderkortet. I sockeln IC34 på processorkortet

placeras Eprom-kretsen med monitorprogrammet ECBUG och datorn är klar att användas. Med monitorprogrammet kan man ändra i minnesceller, beräkna offset för relativ adressering, lägga in brytpunkter, starta program och med ett anpassningskort för kassettspelare spela in och av data på en vanlig audiokassett.

Det finns även en variant av monitorprogrammet för den som direkt vill arbeta med 6809-processorn. Monitorn kallas FOBUG och har samma kommandon som ECBUG. För att bruka datorn behövs till att börja med endast 1 k RAM på adress \$A000-A3FF.

Periferikortets funktion

Principsschemat för det kombinerade indikator- och tangentbordkortet återfinns i *fig. 12:1*. För att ge en enkel programhantering och

ökad användbarhet i andra tillämpningar än just terminalfunktionen används två specialkretsar från **National Semiconductor**. Den ena kretsen har beteckningen *MM74C917* och är en anpassningskrets för lysdiodindikatorer. På ena sidan av kretsen ansluts data och adress-signaler från mikroprocessorn och på den andra sidan sex lysdiodindikatorer av 7-segment-typ. Kretsen innehåller minnen för de olika indikatorerna samt multiplexfunktion för att minska antalet förbindelser mellan indikatorerna och kretsen. Från mikroprocessorns synvinkel adresseras varje lysdiodindikator som en minnescell av "write only"-typ. I kretsen sker hexadecimal avkodning av de fyra databitarna D0-D3 till de sju segmenten på indikatorerna. Databit 7 används för att tända decimalpunkten på varje indikator. I *fig 12:2* finns blockschema för

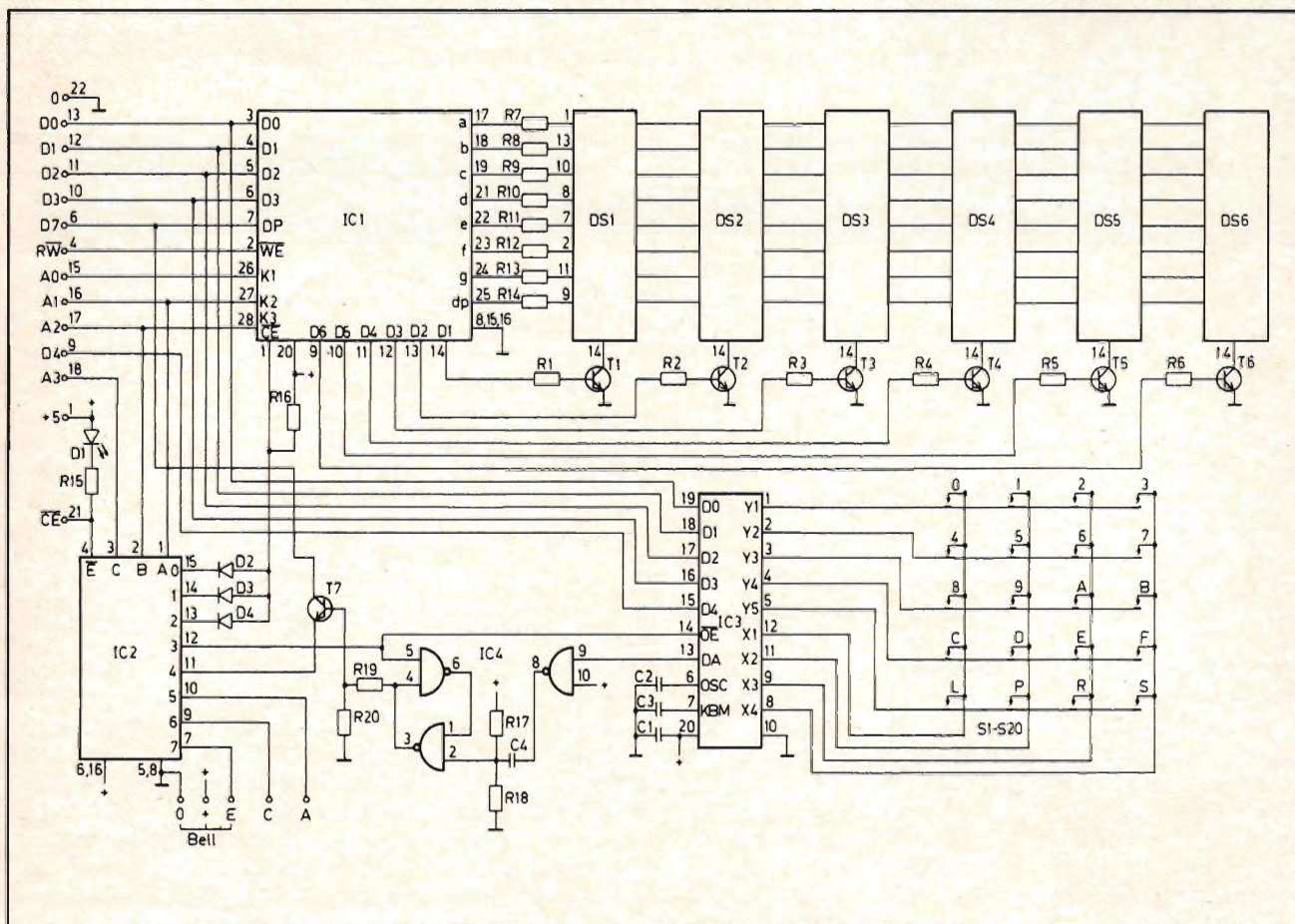


Fig 12:1. Principschema för det lilla terminalkortet.

MM74C917 återgivet.

Den andra specialkretsen har beteckningen MM74C923 och dess blockschema återfinns i fig 12:3. Till kretsen ansluts de 20 tangenterna i matrisform. IC2 är en avkodare, som ger varje funktion på kortet sin unika adress. De sex lysdiodindikatorerna har adresserna \$80E0-\$80E5, data från tangentbordet hämtas på adress \$80E6 och indikering på att någon tangent är nedtryckt finns på adress \$80E8. De tre högsta adresserna \$80EA, \$80EC samt \$80EE är lediga. I programmet ECBUG används dock utgången \$80EE för att ge en puls till en "BELL"-krets av något slag. Tonsignaleringen används för att indikera fel i exempelvis "memory change"-rutinen.

När någon tangent har tryckts ned, kommer utgången DA på IC3 att gå till högre nivå och ställa RS-vippan IC4 så att utgången på stift 3 blir hög. När processorn läser på adress \$80E8, blir databit 7 låg, eftersom T7 fungerar som en grind. När basen är hög och emittorn låg, blir kollektorn också låg. Adressen används vid test för att undersöka om någon tangent har tryckts ner. När en data sedan hämtas kommer en läspuls på utgången stift 12 på IC2. Denna puls återställer RS-vippan IC4 och lägger ut värdet från IC3 över dess ingång OE (output enable). På databussen får man då ett hexadecimalt tal mellan \$E0 och F3 eftersom de tre högsta databitarna alltid är "1".

Komponentplaceringen återfinns i fig 12:4. Om man vill placera indikatorerna och tangenterna i en låda för sig, kapas kortet vid den heldragna linjen och de båda kortdelarna förbinds med flatkabel. På grund av att indikatorerna är multiplexade bör flatkabeln inte vara längre än 50-60 cm, eftersom störningar annars kan komma in på tangentbordskretsen IC3. Om kortet skall monteras odelat kan man ta upp ett hål i fronten och placera ett rött filter framför lysdiodindikatorerna.

Av fig 12:4 framgår även var de olika uppmärskade tangenterna skall lötas in. Indikatorerna löts direkt

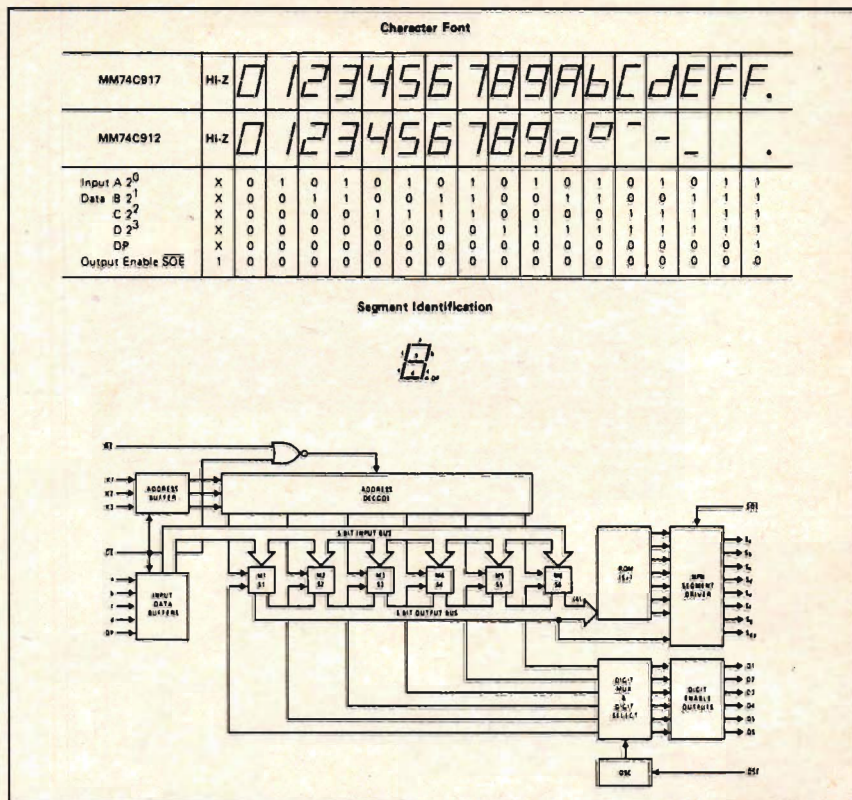


Fig 12:2. Blockschema för displaydrivarkretsen MM74C917.

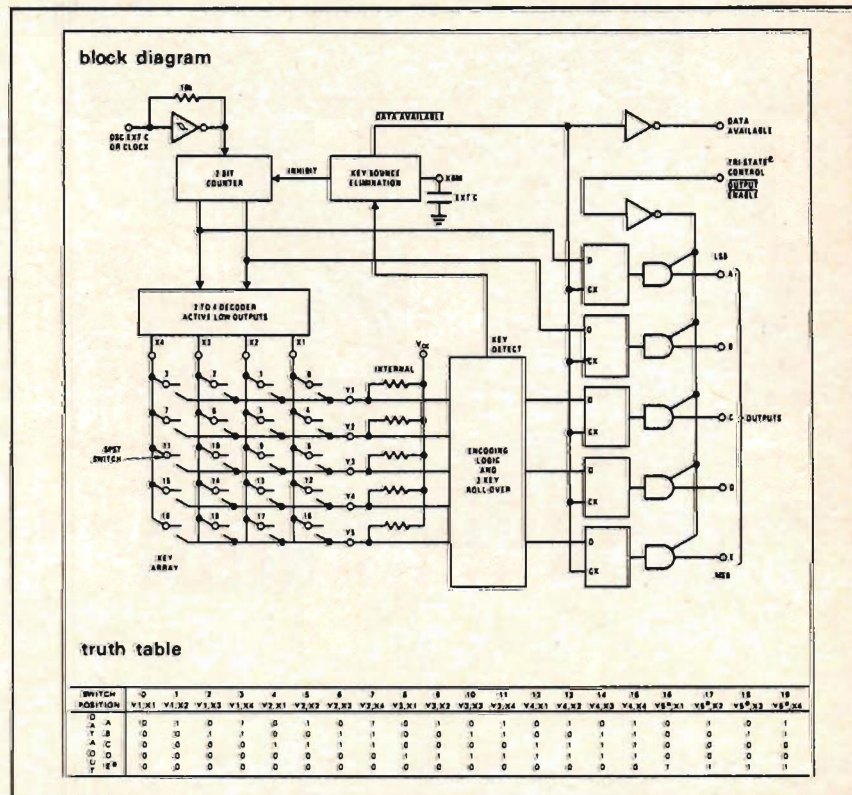


Fig 12:3. Blockschema för tangentbordsavkodaren MM74C923.

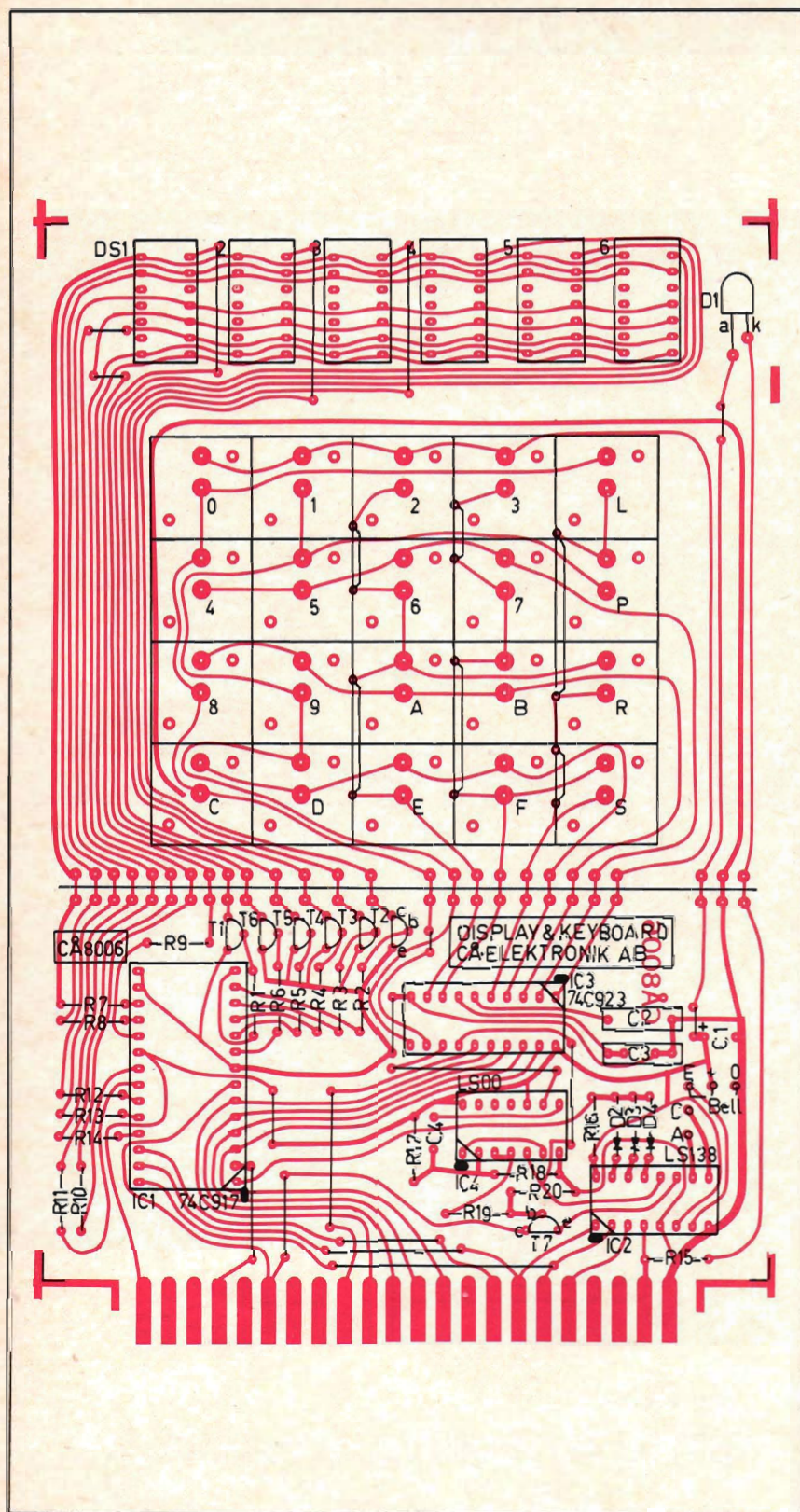


Fig 12:4. Kretskortet sett från komponentsidan. Mönstret är tryckt i färg och komponentplaceringen i svart.

på kretskortet. IC1 och IC3 är av MOS-typ och skall ej sättas i för-rän alla andra komponenter är in-lödda.

När kortet är färdigmonterat kan man prova datorn. På moderkortet skall indikatorkortet placeras i en kontakt på position J15. Kortet vänds så att lödsidan är vänd åt kontaktposition J14. I någon av kontaktarna J16-19 placeras ett processorkort med MC6808. I sockel IC34 placeras monitorprogrammet ECBUG och i socklarna IC10 och IC11 skall två 2114-kretsar sitta. Byglingarna "RAM-adress" på processorkortet kopplas för A-B, och alla fyra bygglarna vid omkopplaren S1 skall vara på plats. Omkopplaren S2 för vektorval kopplas för EC, dvs den fjärde sektionen från vänster skall slås till. För den som skall använda MC 6809 gäller, att sektionen fem (FO) också skall slås till. Monitorprogrammet för 6809 placeras i sockel IC26 och de båda kondensatorerna C1-2 samt kristallen X1 löds in innan IC36 sätts i hållaren.

När spänningarna ansluts skall de sex indikatorerna visa 0.00000 och decimalpunkten blinka långsamt på den första indikatorn.

Monitorprogrammets funktioner

De 20 tangenterna märkta 0-9 samt A, B, C, D, E, F, L, P, R och S används för att mata in hexadecimala tal för data och adresser samt för olika kommandon. För att ge en viss likhet med de monitorprogram vilka har en bildskärm som presentationsenhet har vissa funktioner införts för att förenkla handhavandet för användaren. Detta har även den fördelen, att det blir relativt lätt att vänja om sig vid senare övergång till en bildskärms-terminal.

De fyra första indikatorerna används av monitorprogrammet för att visa adresser och andra 16-bitars tal. De två övriga används för att visa data och 8-bitars tal. Båda monitorprogrammen har samma kommandon och identiska egenskaper. Till varje monitorprogram medföljer en programlistning med

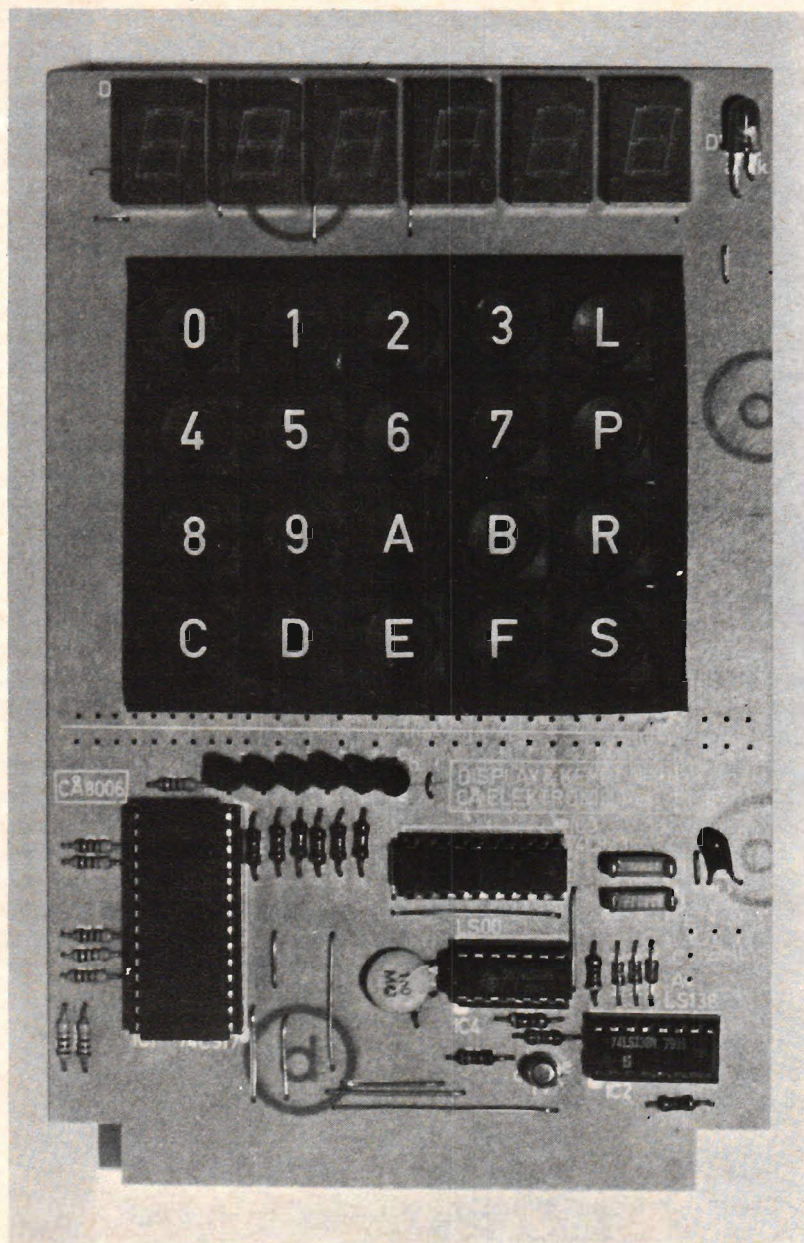


Fig 12:5. Bilden visar det monterade terminalkortet.

kommentarer.

När punkten på första indikatorn blinkar långsamt innebär det att programmet är i kontrollslungan och väntar på ett kommando. Det finns då åtta kommandon att välja mellan:

- A, Minnesmanipulering, används för att undersöka och ev. ändra innehåll i minnesceller.
- B, Brytpunkt, används för att sätta in brytpunkter i programmet.

- C, Continue.
- D, Delete brytpunkt.
- E, Execute program.
- L, Ladda minnet från kassett.
- P, Dumpa minnesinnehåll till kassett.
- R, Registerpresentation.

Kommandona L och P förutsätter att ett kassettmodem (se nästa kap) finns anslutet. För en detaljerad beskrivning av de olika kommandona hänvisas till kapitlet som handlar om monitorprogram. ■

Komponentförteckning

C1	2,2 μ F tantal
C2	22 nF polyester
C3	0,22 μ F polyester
C4	1 nF keram.
D1	lysdiod
D2-4	1N4148
DS1-6	indikator sju-segments lysdiodtyp MM74C917 (NS)
IC1	74LS138
IC2	74LS138
IC3	MM74C923 (NS)
IC4	74LS00
R1-6	150 ohm 1/8 W 5 %
R7-14	100 ohm
R15	220 ohm
R16-18	4,7 k
R19-20	3,3 k
T1-6	BC 337
T7	2N914 el likn.
20	tangenter med tecken
1	IC-hållare 14 pol
1	IC-hållare 16 pol
1	IC-hållare 20 pol
1	IC-hållare 28 pol
1	kretskort CÅ-8006
1	Eprom-krets med monitorprogram ECBUG (6808) eller FOBUG (6809) och lista.

Kassetthanpassning för den enkla terminalen

AV ÅKE HOLM

Nu har turen kommit till ett kassetmodem för lagring av data på vanliga audiokassetter. Modemet arbetar på såväl 300 som 1200 bauds hastighet och är avsett för det i förra avsnittet beskrivna terminalkortet.

■ Till det i förra avsnittet beskrivna lilla terminalkortet fordras ett kassetmodem för att man skall kunna spela in och lagra data på en vanlig kassettspelare.

Kassetmodemet har en egen ACIA-krets för att kommunicera med mikroprocessorn och modemet kan överföra data med både 300 och 1200 baud. Det senare är

värdefullt med tanke på den tid det tar att med 300 bauds hastighet ladda in ett program på 8 k ord i minnet. Med 1200 bauds hastighet minskas den tiden till 25 % och denna tidsvinst kan vara värdefull, speciellt om man vid första försöket att ladda in programmet misslyckas och måste börja om från början igen.

Principschemat visas i fig 13:1. IC7 är oscillatorn för både sändnings- och mottagningsklocka. Den lämnar 4800 Hz (justeras med R30) vid 300 baud och 19200 Hz vid 1200 baud (justeras med R33). Datasignalen, som spelas in på band, består av tonfrekvens, som skiftar mellan 1200 och 2400 Hz, beroende på om det är en nolla eller en etta. För att ge samma tonfrekvenser vid 1200 baud som vid 300 baud delas klockfrekvensen från IC7 i IC8, då S1 står i läge 1200 baud.

Vid avspelning förstärks signalen från bandet i T1 och T2 och på stift 11 på IC1 har det åter blivit en kantvåg med frekvensen 1200/2400 Hz. Dioderna D4 och D5 med tillhörande RC-nät ser till att det på stift 3 på IC1 blir en kort puls för varje flank på insignalen. Genom

att dessa pulser kommer tätare vid 2400 Hz än vid 1200, kan man återvinna ettor och nollor i de båda halvorna av IC4. Till hjälp för detta krävs en puls från stift 10 på IC3. Den skall ha en bredd av 280 μ S för att datasepareringen skall fungera klanderfritt. Första halvan av IC3 är en återtriggande monovippa, som över stift 6 och 7 meddelar ACIA-kretsen att bärvåg finns (DCD=data carrier detect). Lysdioden D7 indikerar bärvågens närvaro och stift 8 på IC4 öppnar datautgången, så att inga störpulser skall komma ut på dataledningen RXD, om bärvågsindikeringen faller ur. Lysdioden D6 blinkar när data kommer in från bandet, den är släckt när data är "1" och tänd när data är "0".

Klockpulser för mottagna data kan man få från signalen som kommer från bandet, men för att öka säkerheten används IC7 även för mottagningsklockan.

Normalt skall KCS-modemet placeras i kontakt J2 på moderkortet i datorn och får då adressen \$8020 för 6800-processorn. IC9 ingår inte i funktionen utan har medtagits därför att KCS-modemet skall kunna få en fullständig av-

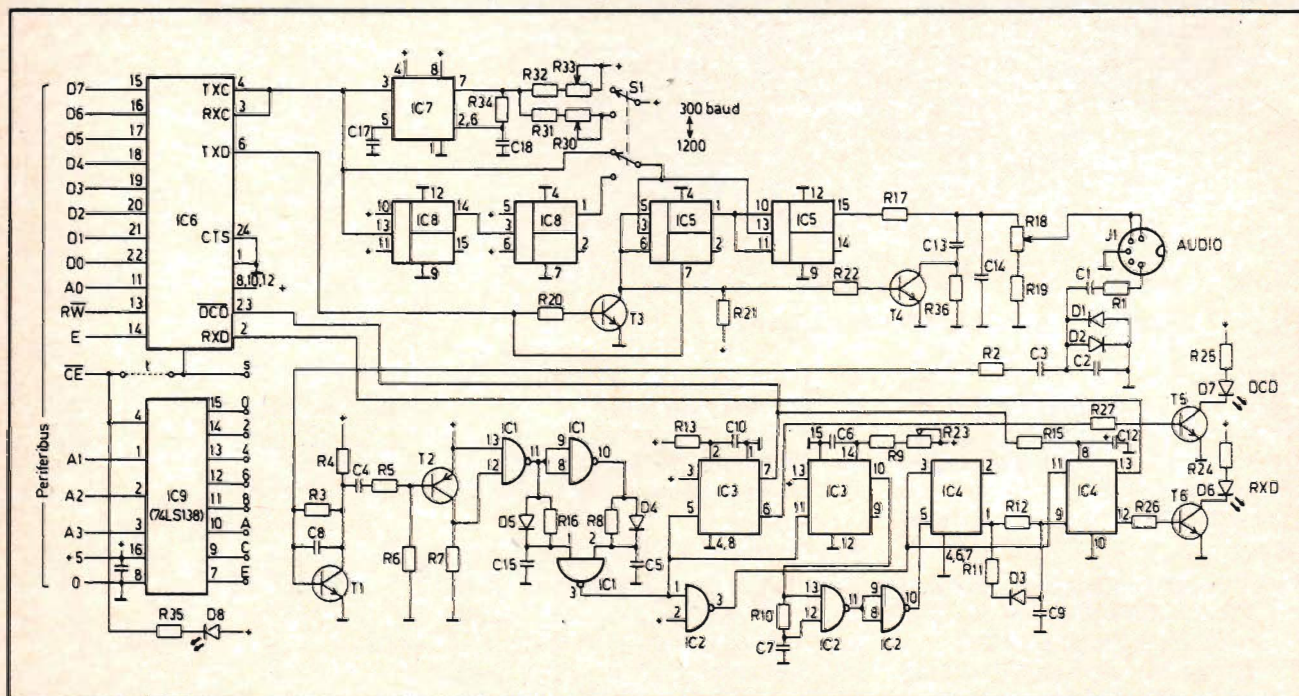


Fig 13:1. Principschema för kassetmodemet.

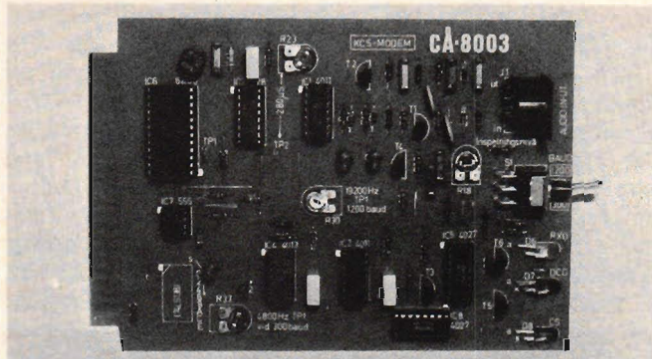


Fig 13:3. Bilden visar det monterade KCS-modemet.

kodning, om det exempelvis skall kopplas parallellt med ACIA-kortet i kontakt J1.

Med KCS-modemet kan man nu spela in data på band och spela av data från band med de båda kommandona P och L i monitorprogrammen ECBUG och FOBUG.

Provning av modemet

Kretskortet för modemet återges i fig 13:2 sett från komponentsidan med komponentplacering. Om IC9 inte skall användas (vilket är det normala), skall inte heller någon sockel inlåsas. Däremot skall bygeln T inlåsas.

När kortet är monterat och kon-

trollerat är det dags att prova det. Den intrimning som behövs består i att justera de tre trimpotentiometrarna enligt beskrivningen. R18 används för att justera in lagom nivå till bandspelarens ingång. Modemet kan sedan provas enligt den beskrivning som gäller kommandona P och L för ECBUG-monitor.

Den som använder FLEX 2.0 operativsystem och vill kunna lägga ut data och text på kassett, kan göra ett program TAPE.SYS för det. Kommandot är i princip uppbyggt som PRINT.SYS med en annan adress för ACIA:n. Istället för att man skickar data till en skrivare matas data till kassettmodemet. ■

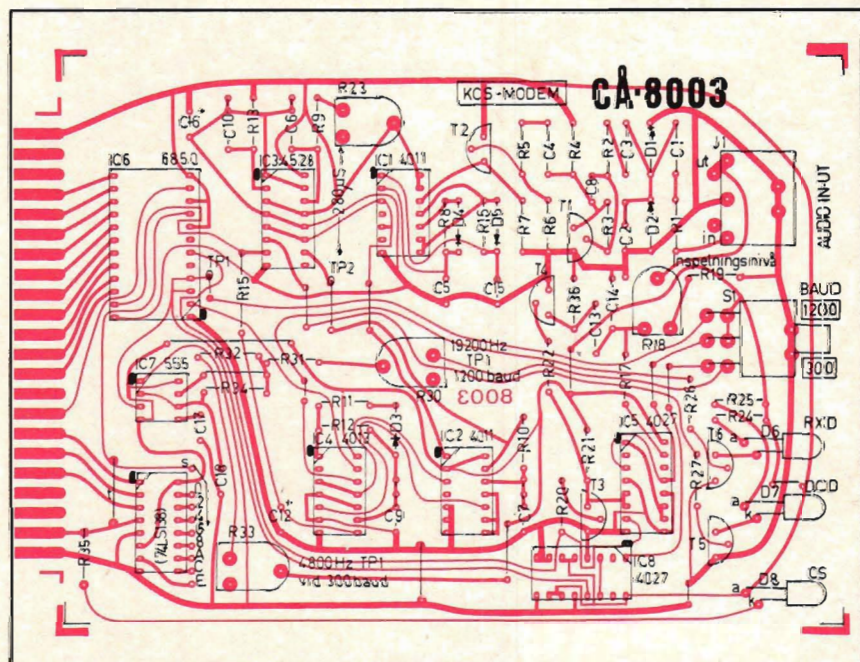


Fig 13:2. Kretskortets ledningsmönster sett från komponentsidan med komponenttrycket i svart.

Komponentförteckning

C1,3-4,	0,22 µF polyester
10	
C2,8	1 nF keram. skiv
C5, 15	2200 pF pol.
C6	8,2 nF pol.
C7,9	6,8 nF pol.
C12	0,68 µF el
	1µF Tantal
C13-14	22 nF pol.
C16	47 µF 16 V
C17	0,1 µF pol.
C18	1 nF pol.
D1-5	1N4148
D6	gul lysdiod
D7	grön lysdiod
D8	röd lysdiod
IC1,2	4011
IC3	4528
IC4	4013
IC5, 8	4027
IC6	6850
IC7	555
(IC9	74LS138)
J1	5-pol. DIN-kontakt
R1	3,3 k 5 % 1/8 W
R2, 19	1 k
R3	680 k
R4-5, 7,17,	
26-27	10 k
R6, 15, 36	100 k
R8, 16	1,5 k
R9	68 k
R10, 21	4,7 k
R11	82 k
R12	33 k
R13	220 k
R18, 23,	
33	100 k trimpot
R20, 22,	
34	22 k
R24, 25,	
35	220 ohm
R30	10 k trimpot
R31	18 k
R32	180 k
S1	2-pol omkopplare för kort
T1, 3-6	BC 548 el likn.
T2	BC 558 el likn.
1	kretskort CA-8003
1	IC-hållare 8 pol.
3	IC-hållare 14-pol.
3	IC-hållare 16-pol.
1	IC-hållare 24-pol.

Minneskort för stora data-mängder

AV ÅKE HOLM

Ett minneskort med dynamiska minneskretsar ingår i RT-datorn. Minnet rymmer 48 k ord och har mycket låg effektförbrukning.

■ I datorn erfordras minne, i vilket det går att både *skriva* in data i och att *läsa* data från. Denna typ av minne kallas program- eller dataminne. En mer allmän benämning är dock RAM – förkortning för Random Access Memory. På processorkortet finns plats för 8 k ord programminne, vilket räcker långt för en nybörjare, som bara använder det lilla terminalkortet. Om man använder videoterminal och skall köra basic i datorn krävs emellertid mera programminne. Man använder då ett särskilt minneskort, vilket med full bestyck-

ning kan lagra 48 k ord. Minneskortet är organiserat som tre 16 k block, vardera fast adresserat till varsin del av minnesarean. Vill man använda en kassett med en basictolk, kan man klara sig med det första 16 k blocket på minneskortet. Basictolken tar 8 k i anspråk och man har sedan 8 k kvar att lagra sitt program i. För den, som använder flexskivesystem med 6808-processorn, erfordras minst 16 k, om den vanliga basictolken skall användas. Om den utökade versionen av basictolken skall användas, erfordras 32 k min-

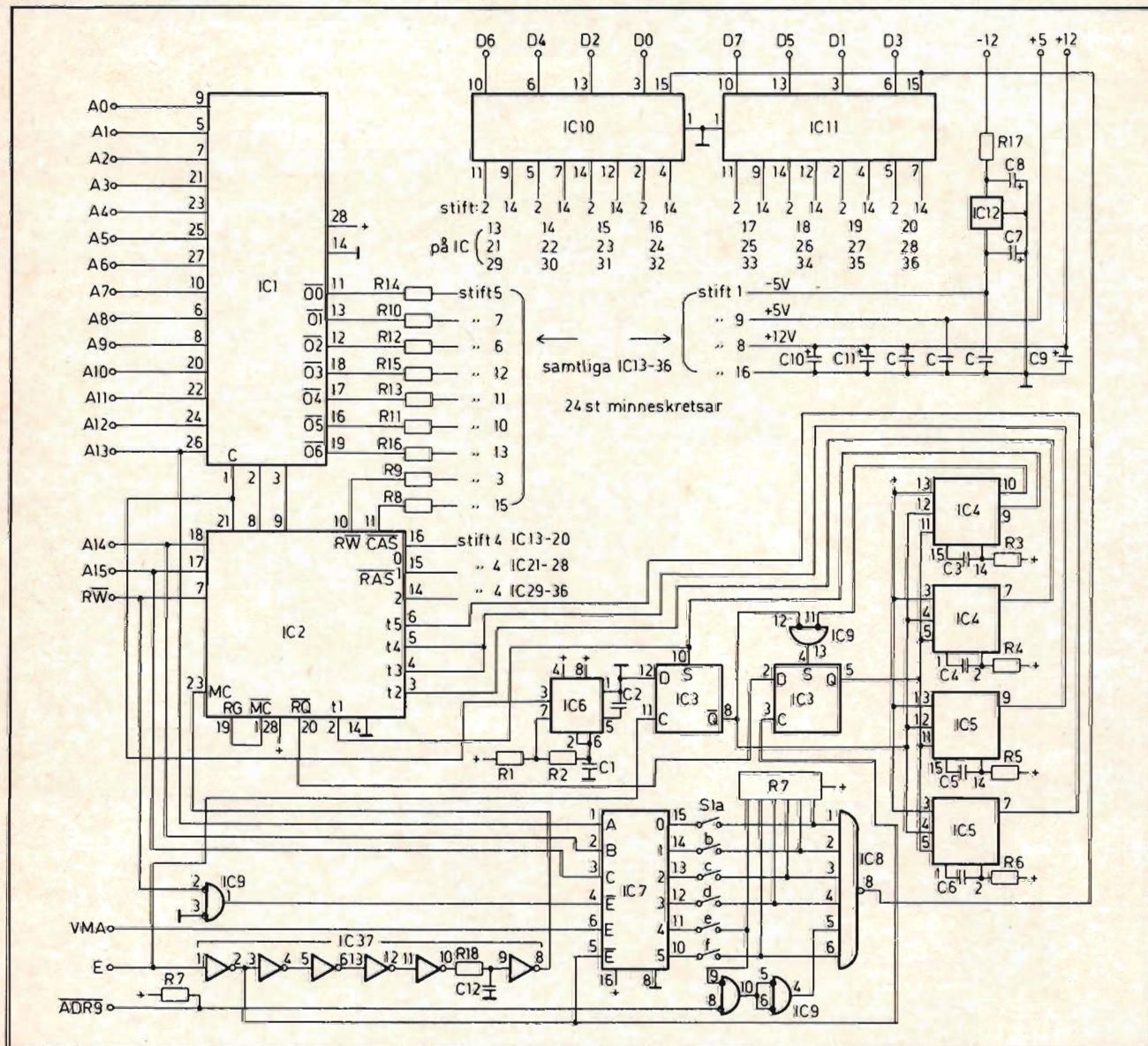


Fig 14:1. Principschema för RT-datorns minneskort på 48 k ord.

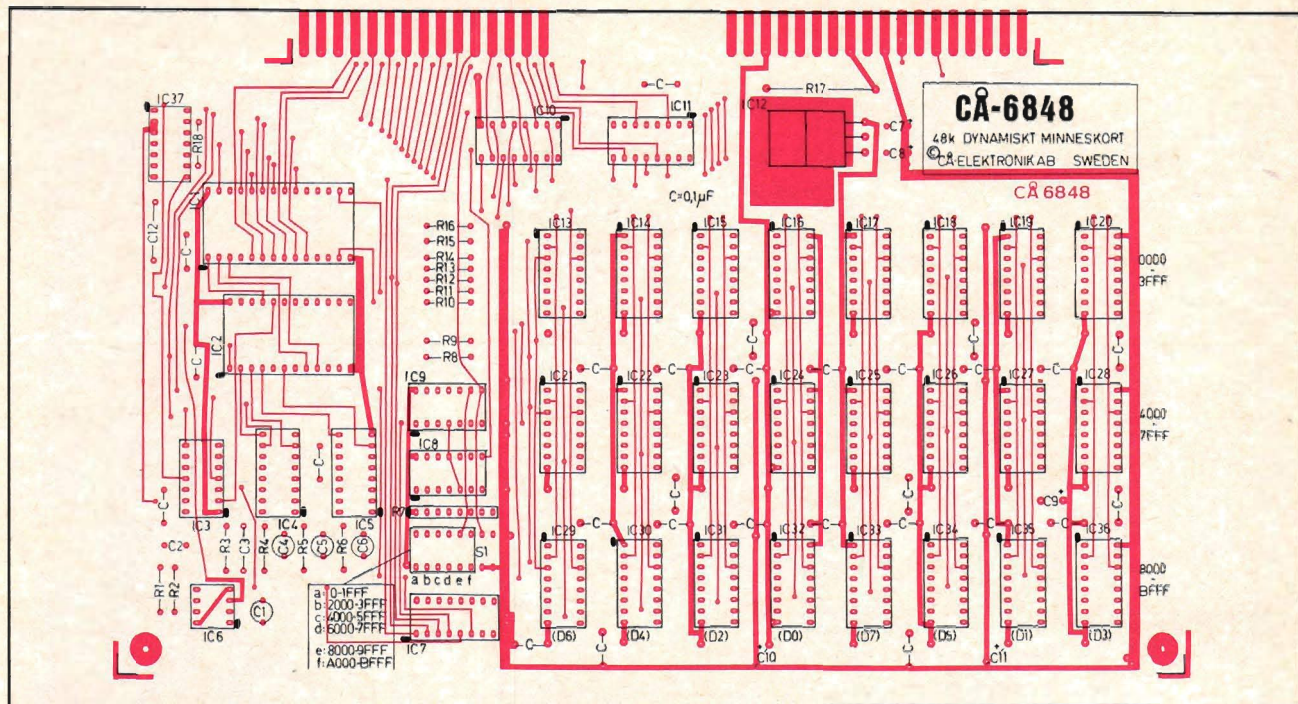


Fig 14:2. Kretskortet sett från komponentsidan. Mönstret visas i färg och komponenttrycket i svart.

ne, eftersom denna basictolk själv behöver 19 k minne. I samtliga fall där flexskivesystem används, fordras 8 k minne mellan adresserna \$A000 och BFFF om enbart 6808-processorn används. Används 6809-processorn med tillhörande flexskivesystem, krävs 8 k minne mellan \$C000 och DFFF. Vill man kunna köra båda processorerna kan det därför vara lämpligt att bestycka minneskortet med alla 48 k och placera de "översta" 8 k (C000–DFFF) på processorkortet. På så sätt har man alltid maximalt minne inkopplat till rätt adresser.

Elektrisk funktion

Principschema för minneskortet återfinns i fig 14:1. De använda minneskretsarna är av dynamisk typ med multiplexad adressbus. Varje krets rymmer 16 384 bitar och det fordras således åtta kretsar till varje minnesblock på 16 k ord. Dynamiska minnen fordrar kretsar för refresh, dvs ständig uppdatering av minnesinnehållet, för att inget data skall gå förlorat. IC2 är en specialkrets för refreshfunktionen och IC1 har hand om adressom-

koppling och stegning av adresserna vid refresh. Minneskortet har dold refresh, vilket innebär att mikroprocessorn inte "märker" att minneskortet behöver refresh med jämna mellanrum. Det senare innebär att en intern refreshfunktion sker var 16:e μ S. IC6 är en oscillator, som lämnar 64 kHz och begär en refreshperiod. Att refreshen är dold innebär att den sker under den tid då E-signalen är låg, dvs då processorn inte läser eller skriver i minnet. Kretsarna IC4–5 är monostabila vippor, vilka genererar pulser som i IC2 styr omkopplingen för adressmultiplexning och skrivpulser. IC37 fördröjer E-signalen så att funktionen hos IC2 inte kommer i konflikt med pulserna från IC4 och 5.

Adressavkodningen sker med IC7. För varje utgång finns en omkopplarsektion för att kunna koppla bort olika delar av minneskortet. Det kan vara aktuellt vid exempelvis felsökning. Varje omkopplarsektion täcker 8 k. För att få automatisk bortkoppling av området \$8000–9FFF då 6808-processorn används, har IC9 längst ner i schemat lagts in. IC10 och IC11 är da-

tabuffrar för in- och utmatning av data från kortet.

Om kortet till en början endast skall bestyckas med 16 k minne, erfordras av minneskretsarna endast IC13–20. På ena sidan av kretskortet, se fig 14:2, finns angivet vilka adresser de olika minnesblocken har. Alla kondensatorer märkta C är avsedda för avkoppling av matningsspänningarna och är på 0,1 μ F. Alla dessa bör inlödats även om bara en del av minneskretsarna monteras.

Provning av minnet

När kortet är monterat och kontrollerat kan det provas. Sätt i alla kretsar och placera kortet i en av de stora kortkontakterna på moderkortet. I TBUG2-monitorn finns ett minnestestprogram, som anropas med kommandot W. Med detta kan hela kortet eller något av de tre minnesblocken provas. Ange start- och slutadress för minnet ex.vis 0000–7FFF. Observera att det inte går att testa minnesområdet \$8000–9FFF med 6808-processorn, eftersom området är bortkopplat. Det går ej heller att testa området \$A000–A07F, eftersom

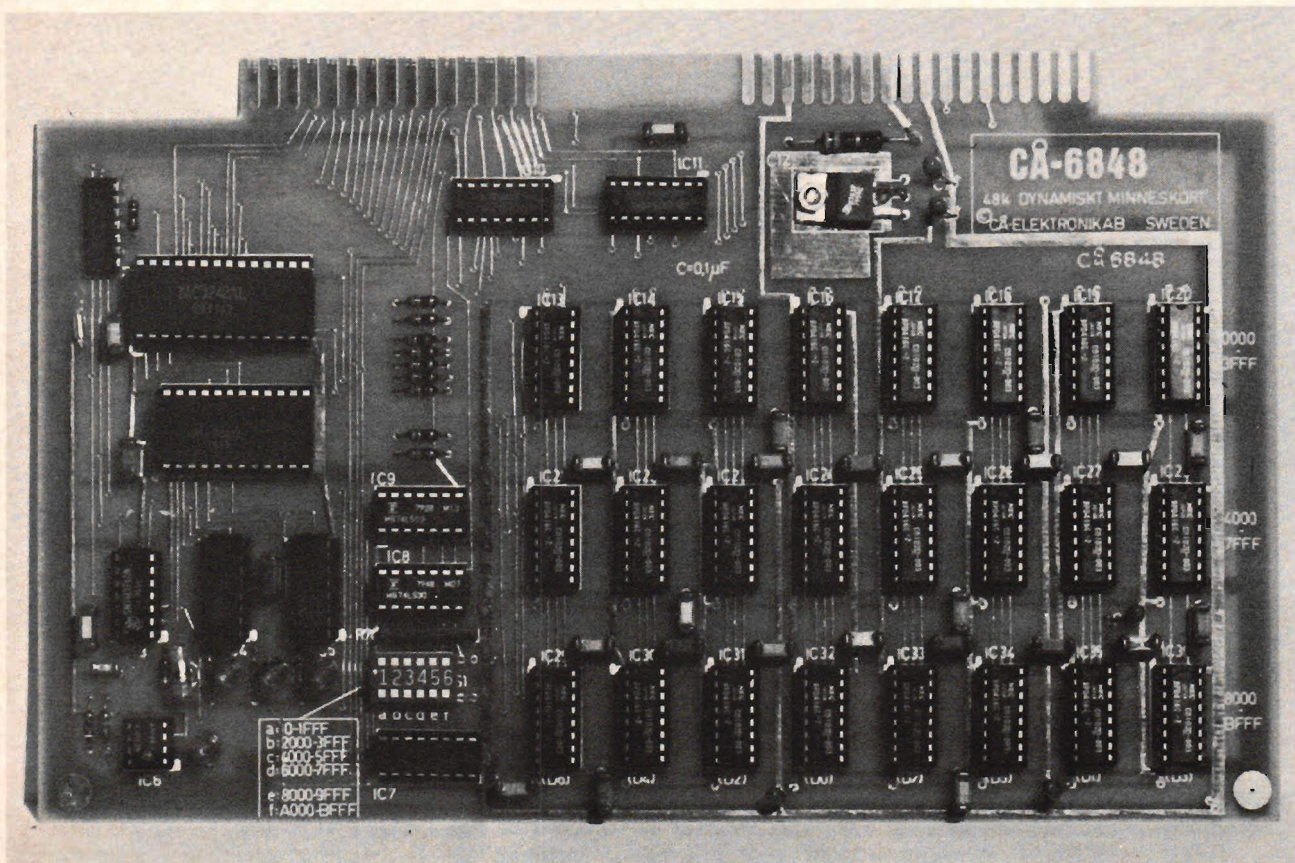


Fig 14:3. Bilden visar det monterade minneskortet.

minnestestprogrammet använder detta område för egna parametrar. Om minnestestet inte hittar något fel, skrivs bokstäverna ABCDEW ut på videoterminalens bildskärm. Om inget ytterligare skrivs ut under den närmaste halvtimmen, kan man vara förvissad om att minneskortet är felfritt. Skulle något vara galet skrivs adresser och data ut på de minnespositioner, där något är fel. Det troligaste felet i sådant fall är att finna kring de kretsar, vilka styr refreshfunktionen. Minneskretsarna bör ha en accesstid av 300 ns eller mindre för att få god marginal.

Ett fullt bestyckat minneskort drar cirka 350 mA från +5 V och cirka 35 mA från +12 V och -12 V. Effektförbrukningen blir då ungefär 2,7 W, vilket är betydligt mindre än vad motsvarande minneskretsar av statisk typ förbrukar vid samma storlek av minne. ■

Komponentförteckning till minneskortet:

C (28 st)	0,1 μ F pol.	IC37	74LS04
C1	470 pF styrol	R1	4,7 k 1/8 W 5 %
C2	0,1 μ F pol.	R2	18 k
C3	47 pF styrol	R3	2,4 k 1 %
C4-6	100 pF styrol	R4	3,6 k 1 %
C7-11	10 μ F 16 V tantal	R5	4,42 k 1 %
C12	220 pF styrol	R6	8,06 k 1 %
IC1	MC 3242 (Motorola)	R7	7 $\times$ 4,7 k
IC2	MC 3480 (Motorola)		SIL-motståndsnät
IC3	74LS74	R8-16	22 ohm
IC4-5	DM 8602	R17	47 ohm 1/2 W
IC6	555	R18	22 ohm
IC7	74LS138	S1	DIP-omkopplare
IC8	74LS30		6-pol.
IC9	74LS02		kretskort CA-6848
IC10-11	MC 6880, 8T26 el		IC-hållare 8-pol.
	75136		IC-hållare 14-pol.
IC12	7905		IC-hållare 16-pol.
IC13-36	4116		IC-hållare 24-pol.
			IC-hållare 28-pol.
			skruv ECS 3 $\times$ 5
			mutter M3

Periferikort för seriekommunikation förbinder videoterminal med dator

AV ÅKE HOLM

Kommunikationen mellan datorn och en videoterminal sker seriellt, och datorn måste därför ha en serieport. Den görs lämpligen med en ACIA-krets. Här beskriver vi ett kort med den funktionen.

■ När vi skall ansluta en videoterminal till datorn, måste vi bygga ett kort för seriekommunikation. I fig 15:1 visas hur detta kort ser ut och i fig 15:2 visas principalschemat. Kortet är avsett att placeras i kontakt J1. ACIA-kretsarna kan då adresseras inom området \$8000–800F resp. \$E000–E00F. Kortet har försetts med två ACIA-kret-

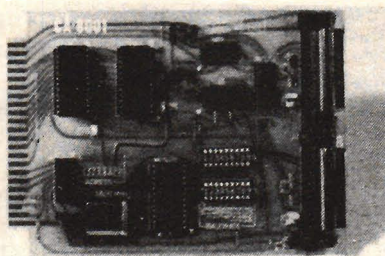


Fig 15:1. Bilden visar det monterade ACIA-kortet.

sar, en för terminalanslutning och en för anslutning av en separat printer. Anslutning av terminal och printer kan ske med de båda 25-pol. kontakterna J1 och J2 eller över kabel och propp till den 16-pol. sockeln till höger på schemat.

Den senare anslutningen är tänkt för vårt fall då man har 25-poliga kontakter, vilka med kabel ansluts till kortet. Den 16-poliga kontakten inkopplas till videoterminal och skriverutgång enligt fig 15:3.

I sitt grundutförande är detta kort bestyckat med endast en ACIA, markerad med A. Till den matas sändnings- och mottagningsklocka från baudrategeneratoren IC4 med tillhörande kristall. Val av baudflöde kan ske med en DIP-omkopplare eller helt enkelt genom att man löder in en tråd på önskat tal.

Kretsarna MC 1488 och MC1489 är anpassningskretsar för RS-232C (V24-snitt). Vissa terminaler fordrar dessutom handskakning till datorn via ledaren CTS, detta för att

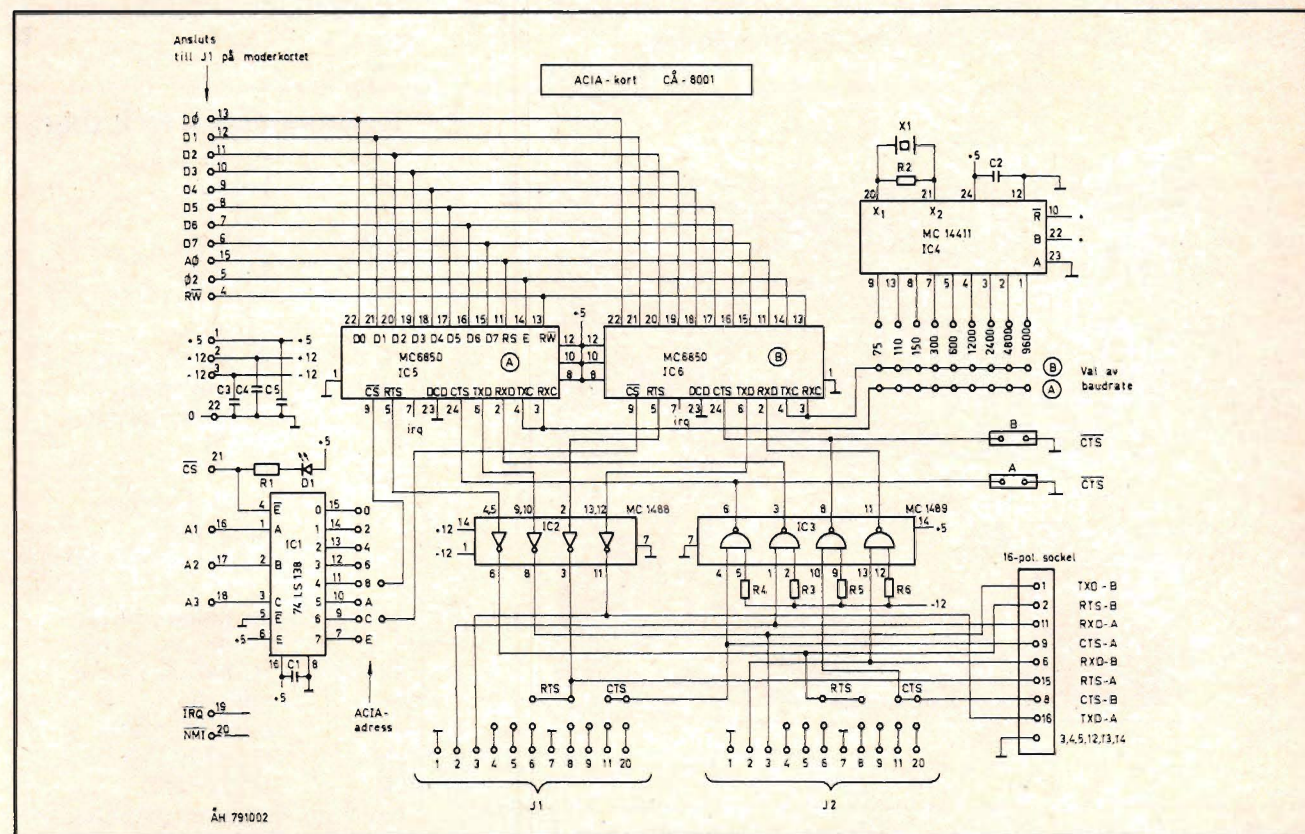


Fig 15:2. Principschema för ACIA-kortet.

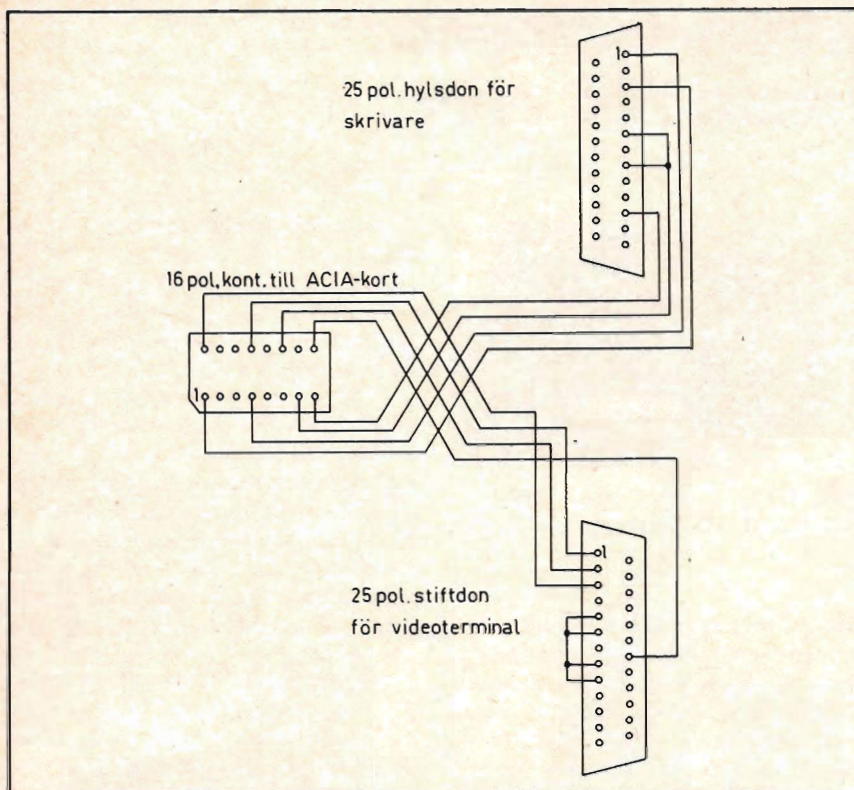


Fig 15:3. Inkoppling av den 16-poliga kontakten till videoterminalens 25-poli-ga kontakt och till kontakten i datorlådan för skrivarutgång.

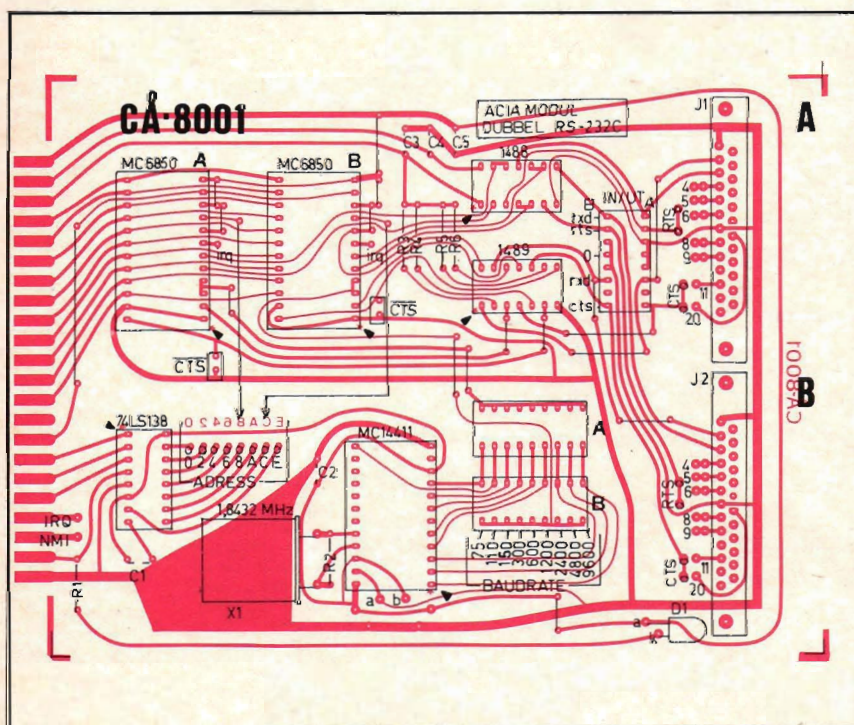


Fig 15:4. Kretskortet sett från komponent sidan. Ledningsmönstret visas i färg och komponenttrycket i svart.

inte tappa tecken vid hög överföringshastighet. För att kunna anpassa till olika terminaler ges man möjlighet till att koppla CTS-ledaren till stift 11 eller 20.

Avkodaren IC1 möjliggör att varje ACIA får sin egen unika adress. ACIA A skall adresseras till hålet märkt 8 och ACIA B till hålet märkt C om TBUG eller CBUG monitorprogram skall användas. Lysdioden D1 indikerar när processorn adresserar kortet. Denna indikation är mest till hjälp vid programutveckling och felsökning.

Observera, att CTS-ledaren till ACIA A måste jordas om den inte ansluts till terminalen. I annat fall kan inga tecken sändas från datorn. RTS-utgången används för närvarande inte, men den har tagits med för att finnas till hands om någon tillämpning så kräver.

Kortet monteras och kan sedan provas tillsammans med processorkortet och TBUG eller CBUG-monitorn. Skrivarutgången används huvudsakligen tillsammans med flexskivesystemets printrutiner.

Komponentförteckning:

C1-5	0,1 μ F pol.
D1	lysdiod
IC1	74LS138
IC2	MC1488
IC3	MC1489
IC4	MC14411
IC5 (IC6)	MC6850
(J1-2)	25-pol. D-kontakt för kortmont.
R1	220 ohm 1/8 W
R2	4,7 M
R3-6	27 k
(S1-2)	9-pol. DIP-omkopplare
X1	kristall 1,8432 MHz
1	kretskort CA-8001
2	IC-hållare 14-pin
2	IC-hållare 16-pin
2 (+1)	IC-hållare 24-pin
6	stiftpar
4	byglar för dito
(4)	skruv ECS 3x8
(4)	mutter M3
Komponenter inom parentes ingår ej i grundsats med en ACIA.	

Intelligent videoterminal till RT-datorn

AV ÅKE HOLM

Videoterminalen till RT-datorn innehåller en egen mikroprocessor och ger möjlighet till färgåtergivning och grafik enligt datavisionsformatet (view-data).

■ Med en videoterminal till datorn kan vi nu börja med högnivåspråk. Videoterminalen arbetar enligt teletextformatet och består av ett kretskort med terminalens elektronik, tangentbord och en videomonitor.

Videomonitorn ingår inte som byggobjekt. Till detta använder man istället en TV av lämpligt slag.

Intelligent terminal med mikrodator

En terminal brukar kallas intelligent när den är bestyckad med en mikroprocessor och då det går att redigera text direkt på bildskärmen.

Med denna terminal, som har full styrning av markören, kan man sätta in eller ta bort tecken eller hela rader med text eller grafik. Det går även att sända en hel rad eller hela sidan till datorn. Alla systemparametrar styrs direkt från tangentbordet. Det går att välja läshastighet i baud och paritet samt halv eller full duplex. Upp till 39 horisontella tabulatorstoppar kan ställas in eller tagas bort efter behag. Markören kan styras från såväl datorn som tangentbordet och tändas eller släckas. Eftersom den är av transparent typ, dvs den inverterar videon på den plats den står, är alla tecken på skärmen synliga.

Kommunikationen med datorn sker via det standardiserade seriensnittet RS-232-C (V24).

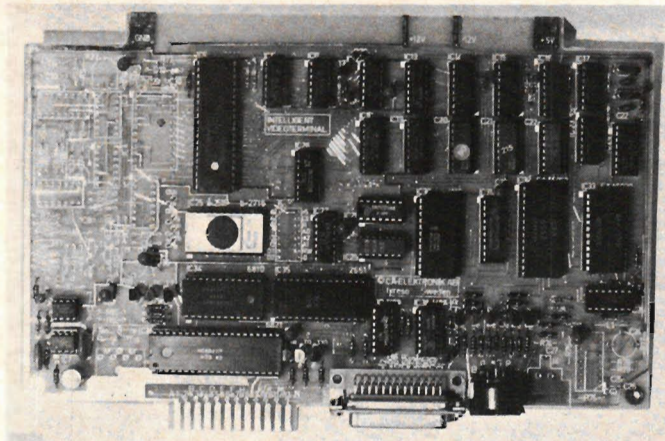


Fig 16:1. Bilden visar ett monterat terminalkort.

Col Row	0	1	2	2a	3	3a	4	5	6	6a	7	7a
0	—	—	□	□	0	□	É	P	é	□	p	□
1	Röd text	Röd grafik	!	□	1	□	A	Q	a	□	q	□
2	Grön text	Grön grafik	"	□	2	□	B	R	b	□	r	□
3	Gul text	Gul grafik	#	□	3	□	C	S	c	□	s	□
4	Blå text	Blå grafik	□	□	4	□	D	T	d	□	t	□
5	Magenta text	Magenta grafik	%	□	5	□	E	U	e	□	u	□
6	Cyan text	Cyan grafik	&	□	6	□	F	V	f	□	v	□
7	Vit text	vit grafik	'	□	7	□	G	W	g	□	w	□
8	Blink start	—	(	□	8	□	H	X	h	□	x	□
9	Blink slut	Anslutande grafik	)	□	9	□	I	Y	i	□	y	□
A	—	Separe-rad grafik	*	□	:	□	J	Z	j	□	z	□
B	—	ESC	+	□	;	□	K	Ä	k	□	ä	□
C	Normal höjd	Svart bakgrund	,	□	<	□	L	Ö	l	□	ö	□
D	Dubbel höjd	Ny bakgrund	-	□	=	□	M	Å	m	□	å	□
E	gå till ASCII mod	Grafik över styrt.	.	□	>	□	N	Ü	n	□	ü	□
F	—	Styrt blankt	/	□	?	□	O	—	o	□	—	□

Fig 16:2. Teckenuppsättningen hos terminalen samt kontrollkoder för färg och grafik i den grafiska moden.

Fyra möjligheter för textuppläggning

Det finns fyra olika möjligheter för textuppläggnen på bildskärmen.

- Det normala är "scroll-mode", då texten börjar i övre vänstra hörnet. När hela sidan är fullskriven, matas all text upp en rad i taget, då den understa raden är fullskriven eller då tecknet för radmatning mottages.

- Det finns även "pagemode", då utskriften börjar om på nytt i övre vänstra hörnet, när sista raden är fullskriven.

- De andra två möjligheterna har vi kallat halvsidesscroll och två-radsscroll. I det första av dessa fall är det bara den undre halvan med 12 rader som radmatas när sista raden är fullskriven. I det andra fallet är det bara de två understa raderna som stegas.

Dessa funktioner är behandliga,

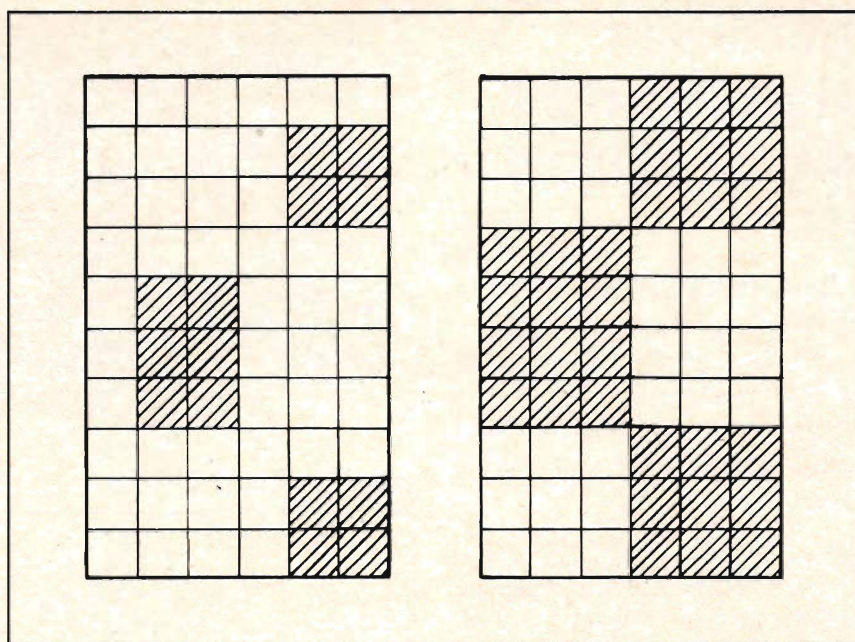


Fig 16:3. Kontinuerlig (B) och separerad (A) grafik.

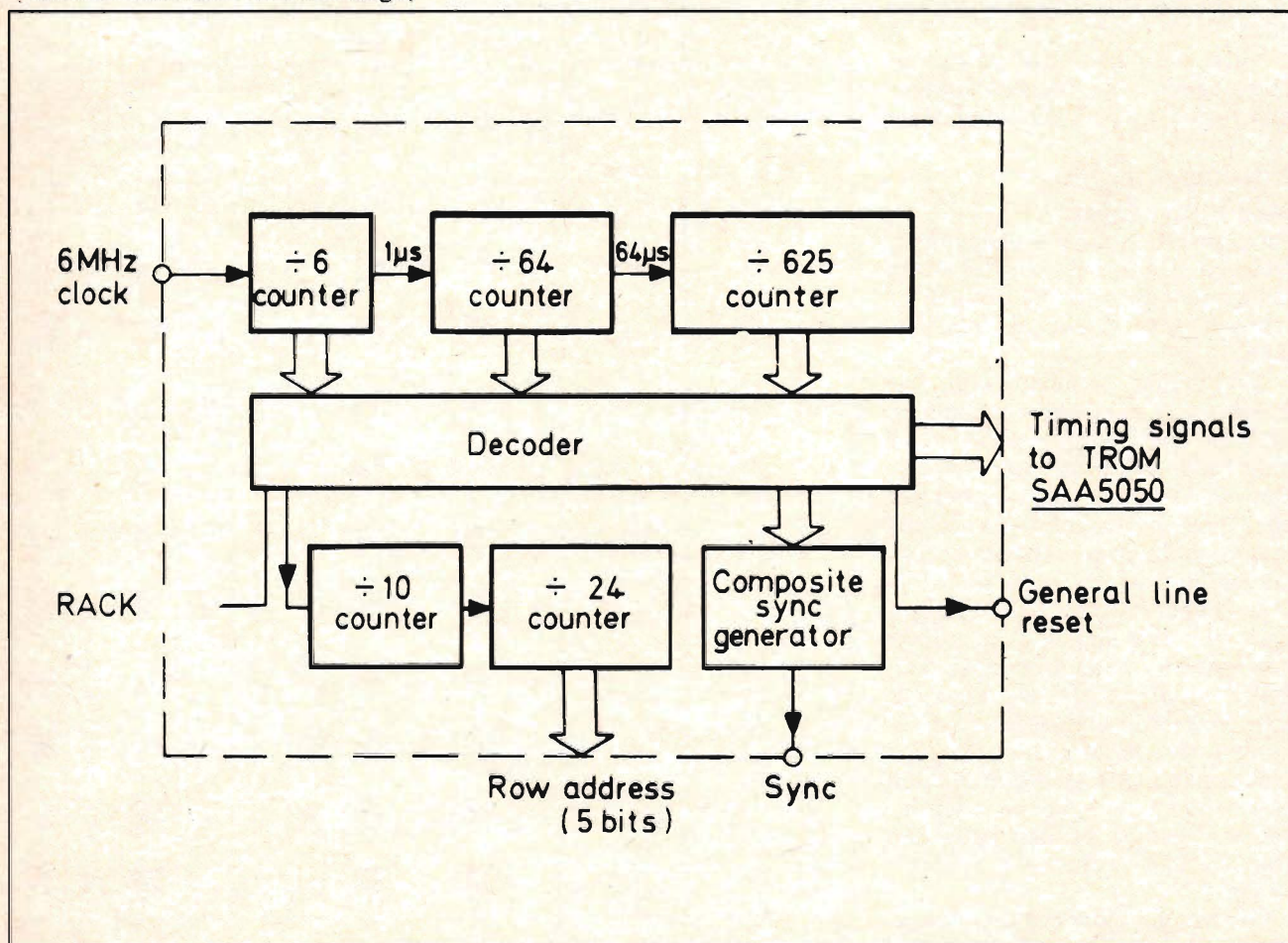


Fig 16:4. Blockschema för Mullard-kretsen SAA 5020.

då man låter datorn skriva ut ett antal rader från exempelvis en texteditor. Därefter kopplar man om till halvsidesscroll, och kan skriva om och rätta de rader som är fel, utan att de tidigare raderna försvinner.

Från datorn kan man även sända kontrollkoder, som möjliggör valfritt antal rader vid scrollning.

Teletextformatet med text och grafik

Teletextformatet innebär att bildskärmen kan innehålla 24 rader med 40 teckenpositioner per rad. Såväl stora som små bokstäver kan återges och naturligtvis finns både Å, Ä och Ö med. Hela teckenuppsättningen återfinns i *fig 16:2*. Det finns 126 grafiska symboler som är uppbyggda av 2×3 punkter. Hälften av dessa är s k separerad grafik, se *fig 16:3*. Både alfanumeriska

tecken och grafiska symboler kan återges i sju olika färger samt blinkande. Varje rad börjar dock för enkelhetens skull med vit alfanumerisk text utan blink.

Den, som har kännedom om ASCII-koden, ser snart i *fig 16:2* att de flesta kontrollkoderna har annan funktion än vad som är brukligt. Detta beror på att terminalen kan arbeta i två olika moder, ASCII-mod (=normal terminal-mod) och grafisk mod. De i *fig 16:2* angivna funktionerna i kolumn 1 och 0 (=koderna \$00–1F) hänför sig till grafikmoden. Omkopplingen mellan dessa båda moder sker med olika kontrolltecken och kan således styras under programkontroll från datorn. En lysdiod på tangentbordet indikerar, när grafikmoden är aktiverad. I båda fallen kan man direkt styra markören till valfritt ställe på bildskärmen.

Trots de många möjligheterna och funktionerna hos denna terminal har dimensionerna kunnat hållas på en rimlig nivå. Detta har varit möjligt främst genom att i konstruktionen använda två LSI-kretsar från Mullard. Dessa båda kretsar är egentligen framtagna för text-TV-dekodrar, men de går utmärkt att använda även till videoterminaler. Den ena kretsen, SAA 5020, innehåller hela synkdelen för videoanpassningen, se *fig 16:4*. Den andra kretsen, SAA 5052, se *fig. 16:5*, innehåller en teckengenerator enligt svensk standard, avkodningskretsar för färg och grafiska symboler samt en av Mullard patenterad koppling för teckenrundning. Tecknen är normalt uppbyggda i matris 5×7, men med denna koppling blir den verkliga upplösningen 10×14 punkter vid 625 linjers system. Man slipper

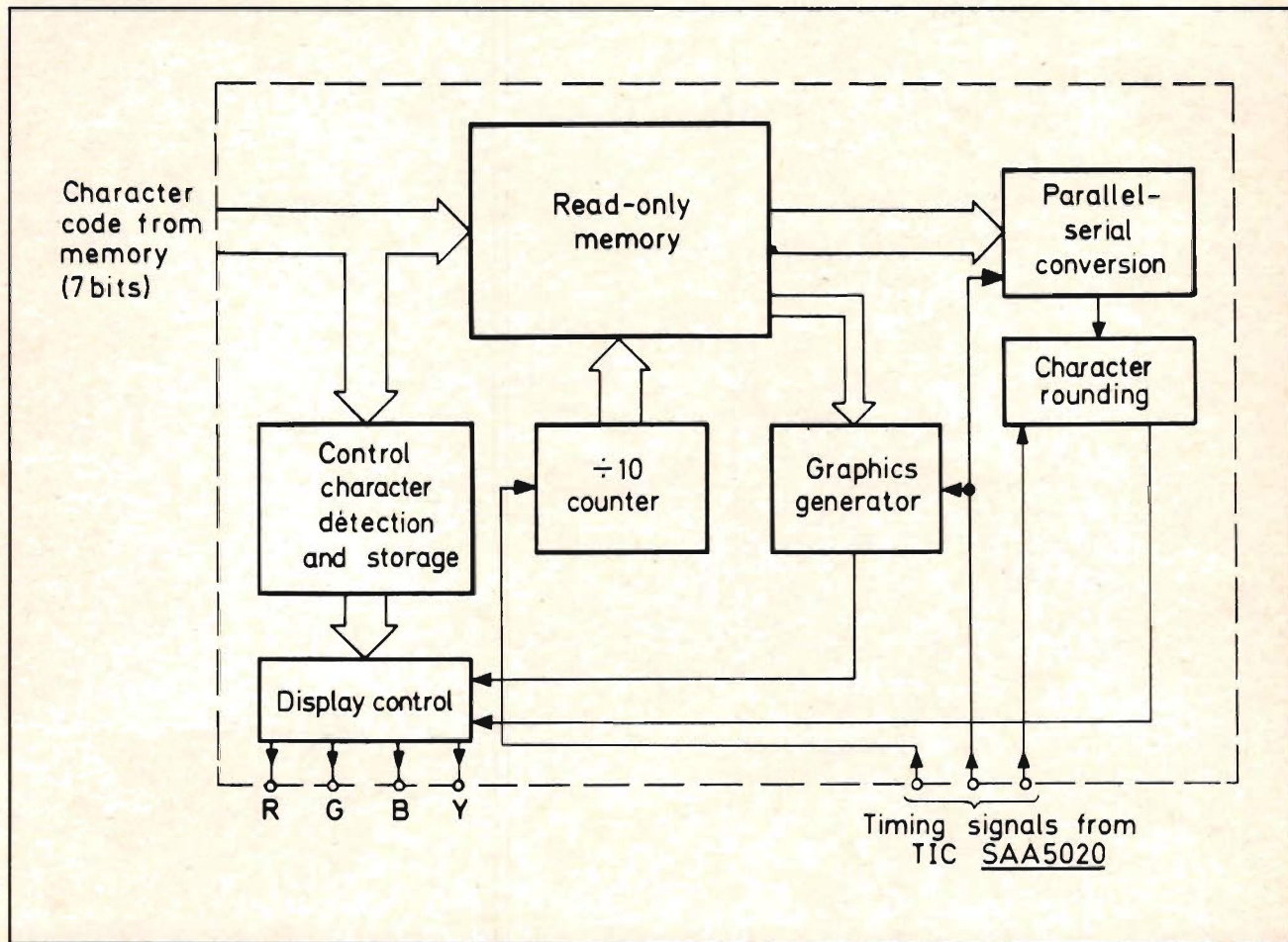


Fig 16:5. Blockschema för Mullard-kretsen SAA 5052.

kantigheten, som annars är vanlig på enklare videoterminaler.

I övrigt är terminalen uppbyggd med en 6800-processor med därtill hörande kretsar för datalagring, programminne, parallellanpassning till tangentbord och serieanpassning till dator och audiokassettspelare.

Elektrisk funktion

I terminalen ingår två kristalloscillatorer. Den ena med kristallen X1 lämnar 12 MHz, vilket blir 6 MHz efter delning i IC23. Dessa 6 MHz är styrfrekvens till IC33 (SAA5020) och för hela videodelen. Den andra kristallen X2 lämnar 10,1376 MHz, vilket är utgångsfrekvensen för baudrategeneratoren i IC35. I IC16 och IC23 delas denna frekvens ner och som resultat erhåller man bland annat 4800 Hz till KCS-modem. IC33 lämnar en uppsättning

styrsignaler till IC32 (SAA5052) och klockpulser för kolumnräknarna IC14 och IC15. Radräknaren ingår i IC33 och utgångarna från den buffras i IC22, innan signalerna kommer till de båda kretsarna IC20 och IC21.

I dessa kretsar, som är förprogrammerade bipolära läsminnen, omvandlas kolumner och rader till en linjär adress hos bildskärmsminnet IC30. Utgången F1 på IC33 är en 1 MHz signal, som används till klocksignal för den i terminalen ingående processorn IC9 (6800). IC10 genererar de båda klockfaserna $\Phi 1$ och $\Phi 2$ till IC9.

Synkroniseringen av videodelen med processorklockan är så utförd, att dessa båda turas om att använda bildskärmsminnet IC30. Då $\Phi 2$ är hög är det processorn, som kan nå IC30, och då $\Phi 1$ är hög är det videodelen, som hämtar

tecken från IC30 och skickar ut det till videomonitorn. På detta sätt får man aldrig några störningar i bilden, när utskrift sker. IC31 är en åttabits hållkrets för teckengeneratoren i IC32. IC18 och IC19 kopplar in adressbussen från IC9 till bildskärmsminnet, då $\Phi 2$ är hög.

Den högsta biten på varje teckenposition används för att indikera markörens position. Denna bit kommer från stift 19 på IC31 och fördröjs i IC29 och IC28 så att den hamnar på rätt teckenplats. Markören adderas till videosignalen i IC39, vilken är en exklusiv-ellergrind. Utsignalen från IC39 matas till fyra emitterföljare T13–T16, tre för färgsignalerna röd, grön och blå, samt en för den svartvita utsignalen. Den senare förses även med synkpulser via D6 innan den matas till utgångskontakten och VHF-modularen IC40. På J4 finns

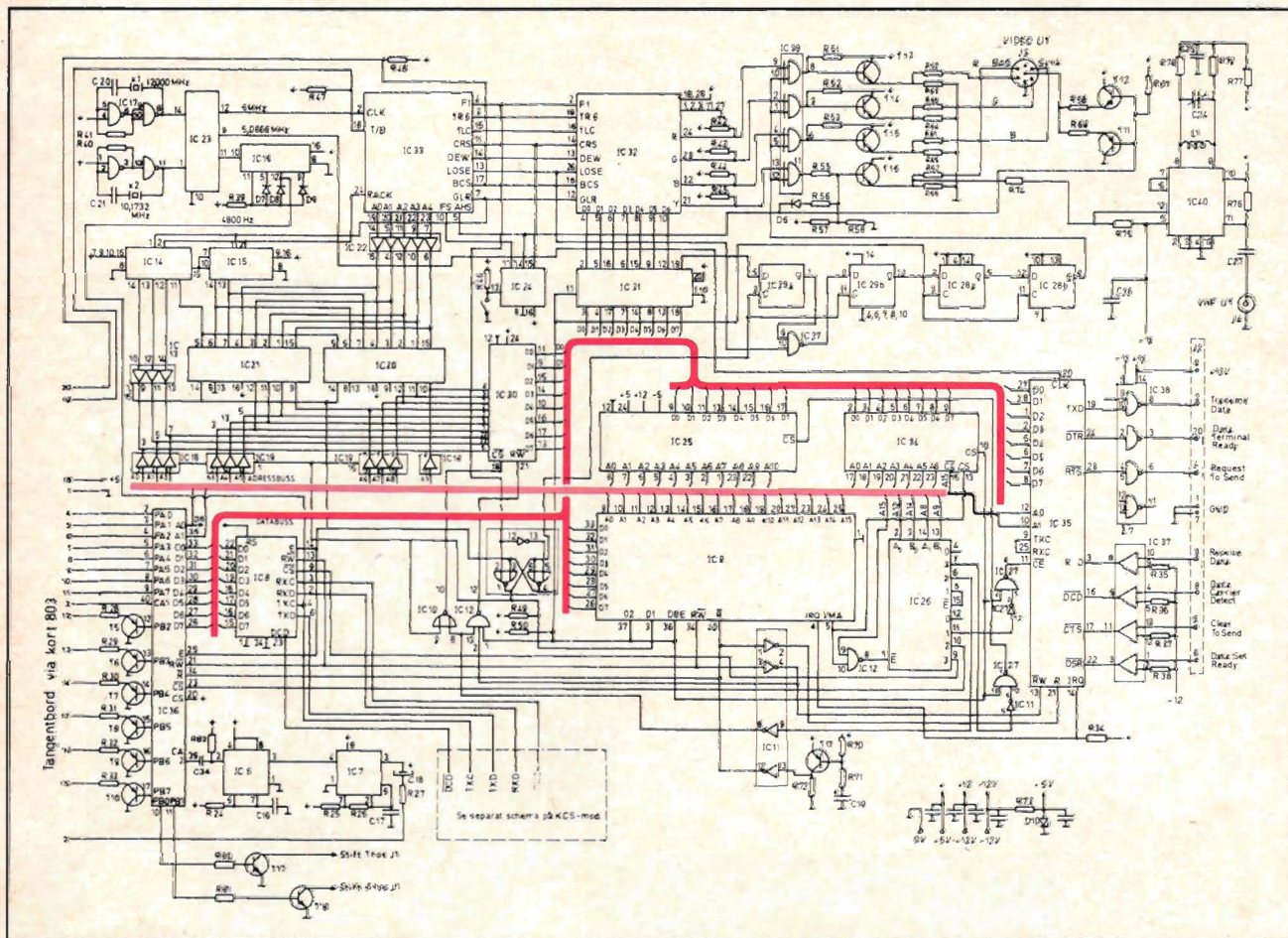


Fig 16:6. Principskemat över den intelligenta videoterminalen.

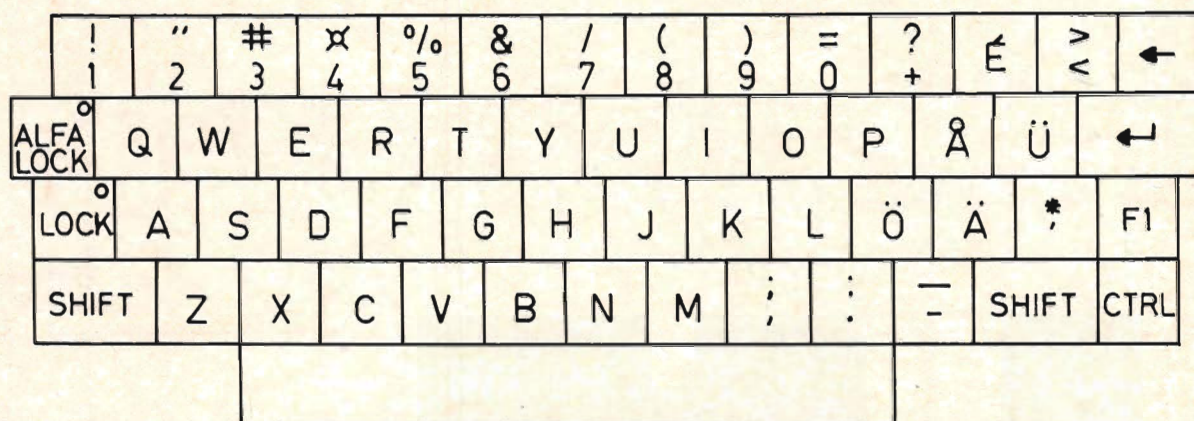


Fig 16:7. Tangenternas placering på tangentbordet L1980 efter modifiering.

vhf-bärvåg som kan inställas mellan kanal 3 och 4. De tre färgsignalerna har 0,7 volts amplitud vid 75 ohms belastning. Den sammansatta videosignalen har 0,3 volt synk och 0,7 volt video. För att driva en färgmonitor fordras även en separat utgång med sammansatt synk, och det finns på stift 6 på J3. Amplituden är ca 4 volt vid 75 ohms belastning.

Mikrodatordelen i terminalen består av processorkretsen IC9, ett dataminne IC34 (6810), ett programminne i form av ett eprom IC25 (2716), en PIA-krets IC36 (6821) för tangentbordet och lysdioderna samt en seriekommunikationskrets IC26 (2651) för datoranslutningen. IC8 är en extra seriekommunikationskrets som används för KCS-modemet. De övriga kretsarna är IC26 vilket är en adressavkodare samt IC6 och IC7 vilka alstrar bell-signalen. T17 ger terminalen en resetpuls vid tillslag av matningsspänningen. De två kretsarna längst till höger, IC37 och IC38 nivåanpassar in- och utgångarna till datorn.

Teletext utan radsprång

Teletextkretsarna arbetar normalt med 625 linjers system och den

bild som texten presenteras på är tänkt att betraktas på normalt tv-tittaravstånd. I datorsammanhang är det ibland irriterande att betrakta en textskärm som arbetar med radsprång (vilket det alltid blir vid udda linjetal). En textskärm betraktas som regel på kort avstånd, och radsprång är därför inte önskvärt. För att ordna detta har vi klämt in IC24 som räknar linjepulser och "lurar" IC33 att skicka iväg några linjepulser för mycket. Resultatet är att videosignalen har 628 linjer (eller rättare 314 linjer per delbild) utan radsprång då stift 13 på IC24 är jordat. Avlägsnas jordningen får vi en vanlig 625-linjers bild.

Det går att ansluta en skrivare för utskrift av data. Inkopplingen av denna kräver en yttre ACIA-krets och skall ske över en 16-polig ic-sockel märkt J5 på kretskortet. På denna sockel finns anslutet databuss och nödvändiga styrsignaler. Vilka stift som har vad framgår av komponenttrycket på kretskortet.

Tangentbordet

Som tangentbord kan man i princip använda vilket avkodat tangentbord som helst. Med tanke på att

teckenuppsättningen är svensk används emellertid **Keytronics** s k Sverige-bord.

För att tangentbordet skall fungera ihop med videoterminalen måste vissa modifieringar först utföras. Man börjar med att flytta tre tangenter. Tangenten *REP* toges bort och istället placeras *F1*-tangenter på dess plats. *CTRL*-tangenter flyttas till *F1*-tangents förra plats, och en *ALPHA LOCK*-tangent placeras överst till vänster där *CTRL* satt. Tangenterna skall nu sitta enligt fig 16:7.

På tangentbordets kretskorts översida finns en IC med beteckningen 7410. Den sitter i mitten av den bakre raden IC. Klipp av stift 10 på denna IC. Det räcker ej med att kapa ledaren på kortets baksida! På kortets undersida finns två punkter märkta 4 och 5 (bredvid den stora IC:n) som är förbundna med varandra. Kapa förbindelsen mellan dessa! På kortets baksida i det hörn, där *F1*-tangenter sitter, finns tre punkter märkta 18-19-20. Kapa förbindelsen mellan 18 och 19. Förbind sedan 18 och 20 istället. Bakom *ALPHA LOCK*-tangenter finns också tre punkter märkta 12-13-14. Kapa förbindelsen mellan 12 och 14 och förbind

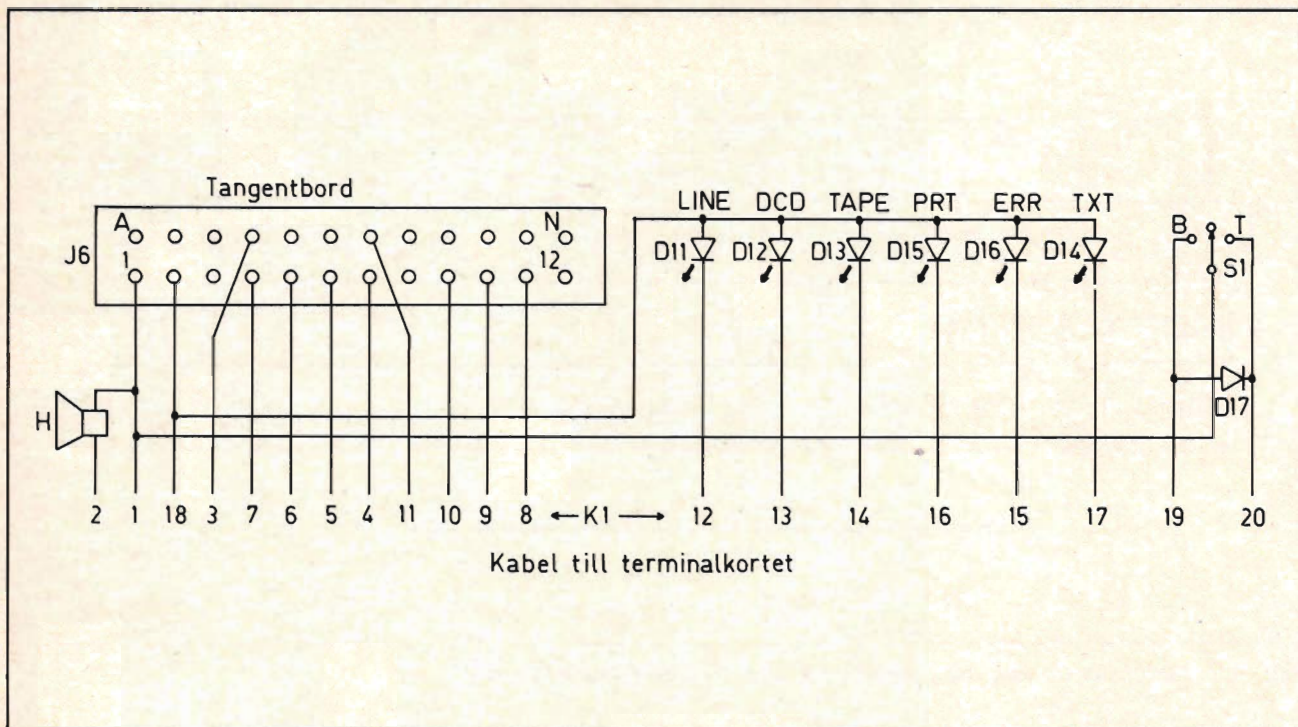


Fig 16:8. Principischemat över kretskortet för tangentbordsinkopplingen.

istället 14 med 13. Glöm sedan inte att avlägsna den lilla tejpremsan med aluminiumfolie, innan kortkontakten ansluts.

Med dessa ändringar har tangentbordet nu en alpha lock-funktion. Denna innebär att då lysdioden i ALPHA LOCK-tangenten är släckt går det att skriva både små (oskiftat) och stora (skiftat) bokstäver. Siffror och andra tecken fungerar som vanligt. När lysdioden är tänd, kan man bara skriva stora bokstäver oavsett om skift-tangenten är nertryckt eller ej. Detta är det normala vid kommunikation med datorn. Små bokstäver används huvudsakligen vid skrift av brev och vid uppbyggnaden av grafiska symboler.

Repetition erhålls automatiskt när en tangent hålls nertryckt länge. Tangenten F1 är kopplad så att den högsta databiten får hög nivå då tangenten trycks ner. Terminalen informeras om att det gäller styrkod och kopplar om terminalen på önskat sätt. På tangentbordet finns även sex lysdioder och en omkopplare. I botten på tangentbordslådan är högtalaren för "bell"-tonen monterad.

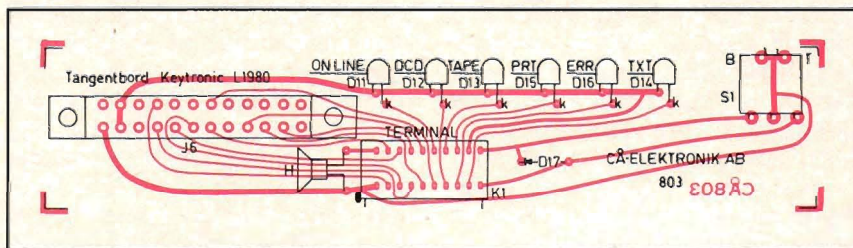


Fig 16:9. Komponentplaceringen på det lilla kretskortet i tangentbordslådan.

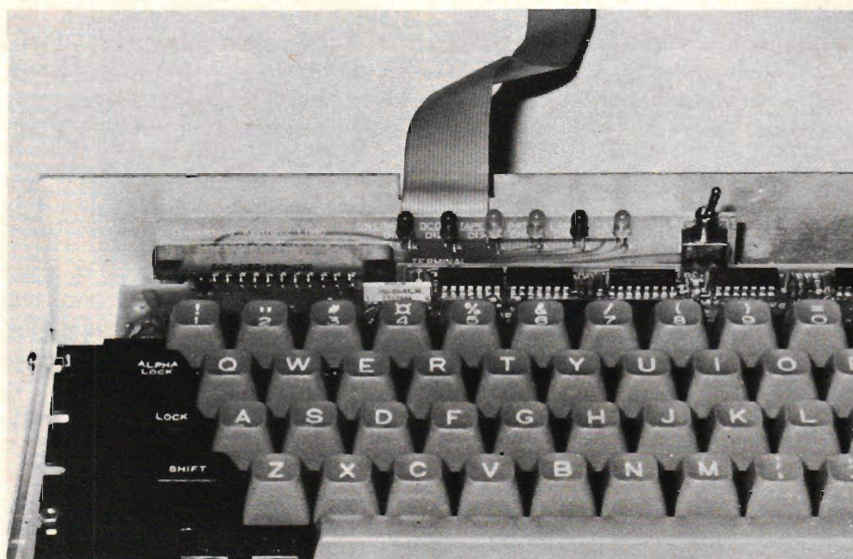


Fig 16:10. Bilden visar hur kretskortet i tangentbordslådan monteras.

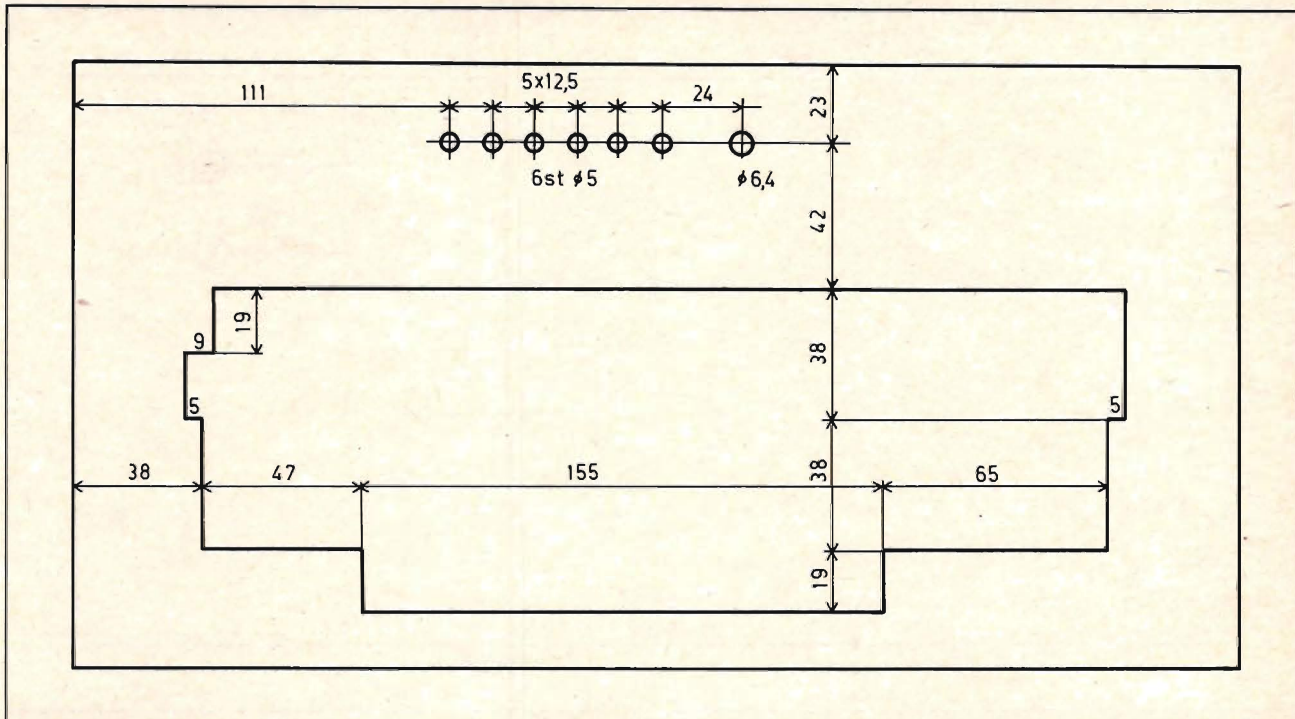


Fig 16:11. Måttskiss för håltagningen i tangentbordslådans lock.

De sex lysdioderna på tangentbordet indikerar olika driftstillstånd hos terminalen. *On Line* lyser när terminalen är kopplad för full eller halv duplex till datorn, *DCD* indikerar Data Carrier Detect från datorn. *TAPE* är tänd, då terminalen har KCS-modem monterat och då automatisk in- och urkoppling av detta kan ske med speciella kontrolltecken. Då terminalen är i grafikmod lyser *TXT*, och när printrutinen är inkopplad lyser *PRT*. Lysdioden *ERROR* indikerar mottagningsfel i form av paritetsfel eller annan felrapportering från IC35. Med omkopplaren i mittläge återges alla 24 raderna och med omkopplaren i något av sidolägena återges de övre resp undre 12 raderna med dubbel teckenhöjd. I fig 16:8 visas principschemat för det lilla kretskortet i tangentbordslådans och i fig 16:9 kretskortet med komponentplacering. Kretskortet monteras så att flatkabeln kan vikas runt kortet och ledas ut genom en slits i bakkanten på lådan, se fig 16:10. I fig 16:11 och 16:12 finns måttuppgifter på hålen i lådan. I byggsatserna levereras lådan med det stora hålet färdigt. Hålen i bot-

ten på lådan är avsedda för fastsättning av tangentbordet och för högtalaren. De korta skruvarna monteras i framkanten och de långa skruvarna längst in. Med dubbla muttrar justeras sedan tangentbordets höjd så att underkanten på tangenterna kommer i nivå med lådans lock och så att omkopplaren på det lilla kortet kan fästas i locket.

Hålen vid lysdioderna märks upp med motsvarande beteckningar. Man kan lämpligtvis använda vita gnuggbokstäver till detta. Högtalaren går bra att limma fast i botten på lådan ovanför de fyra hålen. För att få en förbindning som är enkel att utföra, används flatkabel med monterade klämdon. Den med rött markerade ledaren i flatkabeln skall vändas åt kontakten J1 på terminalkortet.

Montering av terminalkortet

I fig 16:13 återfinns kretskortet med komponentplacering.

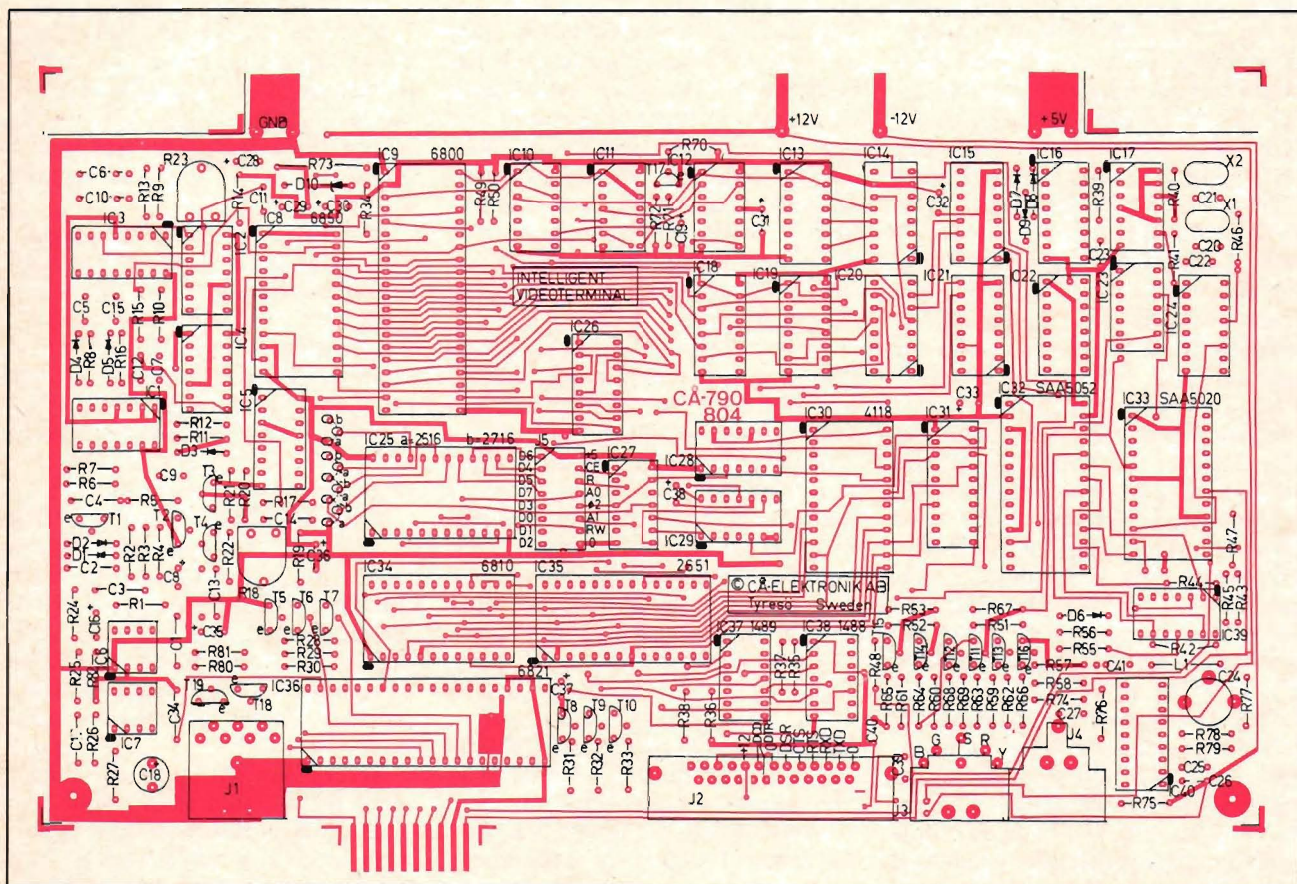
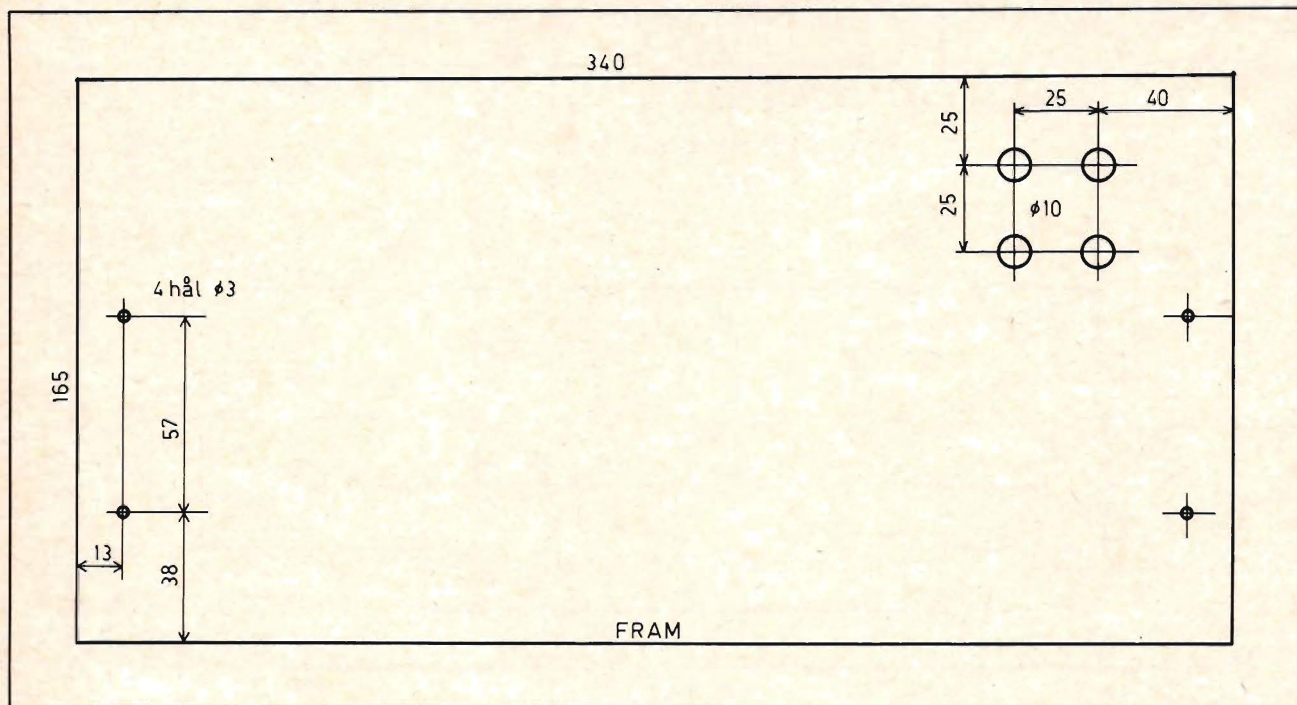
Till IC25 kan det finnas två typer av eprom-kretsar, en som matas med enbart +5 V, och en som dessutom fordrar +12 och -5 V.

För att kunna använda båda dessa typer av kretsar har kortet försetts med ett bygglingsfält bredvid IC25. Hur detta skall bygglas framgår av komponenttrycket. Om IC25 är av +5 V-typen behövs inte D10 och R73.

Observera att vissa tillverkare av eprom-kretsar kallar enspänningsvarianten av IC25 för 2716, 8516 eller 2516. Eftersom trespänningsvarianten nu är den minst vanliga, kan man förutsätta att den levererade kretsen är av enspänningsvarianten om ej särskilt angivits. D10, R73, C29 och C30 ingår ej i byggsatser med enspännings-IC25.

Vid monteringen av J2 får man se upp med de monteringsvinklar, som skall skruvas fast i kretskortet. På vissa utföranden av J2 är dessa vinklar av metall och man måste då lägga isolering mellan vinkeln och kretskortets ledningsmönster för att undvika kortslutning.

Stycklistan är uppdelad i flera delar för att man skall kunna särskilja de komponenter som ingår i KCS-modemet och hf-delen. Terminalen kan byggas utan dessa delar om man inte har behov av dem.



Dags för provning

När kortet är monterat och kontrollerat kan det provas etappvis. För provningen fordras videomonitor och tangentbord.

Börja med att sätta i IC10, 16, 17, 23, 24, 27, 28, 29, 31, 32, 33 och 39. Anslut videomonitorn och spänningarna. På bildskärmen skall nu finnas ett ruttmönster. Om det är fallet kan man fortsätta, i annat fall måste man först kontrollera att kristalloscillatorerna svänger och att det finns 1 MHz-signal på ledningarna från stift 1 och 4 från IC10. När allt är OK stänger man av spänningarna och sätter i IC11, 12, 13, 14, 15, 18, 19, 20, 21, 22 och 30. När spänningarna ansluts skall man på bildskärmen få ett virrvarr av tecken (i olika färger). Skulle så inte vara fallet, får man kontrollera signalerna till och från IC20 och IC21 samt ingångarna på IC31. På IC20 och 21 skall det finnas pulser synkroniserade med bild och linjefrekvens. Är allt i sin ordning kan resterande kretsar sättas på plats. Var noga med att alla kretsar vänds åt rätt håll. En vit markering vid varje sockel visar var stift 1 skall vara. Alla ic kunde vid kretskortlayouten tyvärr ej vändas åt samma håll, och en extra koll kan därför spara mycket felsökning.

Anslut även tangentbordet.

När spänningen ansluts denna gång skall skärmen raderas och skrivmarkören bli synlig i övre vänstra hörnet. Efter markören står några tecken som indikerar vilken (eventuell) variant av terminalprogram som ingår. Dessa tecken syns bara vid uppstartning eller då vissa kontrollfunktioner ändras.

Vid start är terminalen kopplad för 9600 baud, full duplex, on line och ASCII-mod. För att kunna kommunicera med datorn via J2 fordras viss information på de olika stiften för att sändning och mottagning skall kunna äga rum.

I tabell 10:1 återfinns en sammanfattning av vad de olika stiften på J2 har för signaler.

I fig 15:3 visas hur J2 skall kopplas in till kontakten på ACIA-kortet. Ledningen markerad DTR in-

dikerar när terminalen är upptagen. Det tar längre tid för terminalen att ta hand om det sista tecknet på nedre raden, eftersom hela bilden då skall matas upp en rad. Under denna tid måste därför teckenutmatningen från datorn stoppas för att inte tappa några tecken.

Om anslutningen till datorn nu är rätt utförd, är det bara att börja använda terminalen.

Det är att rekommendera att man först i lokal mod provar alla funktioner så att man får ett grepp om terminalens egenskaper. Det sker genom att man använder de kontrollkoder som finns i slutet av detta kapitel.

Lagring på band med ett KCS-modem

På terminalkortet finns även plats för ett KCS-modem för anslutning av en audiokassettspelare. En sådan är ett enkelt hjälpmedel för lagring av data till små mikrodatorsystem. Data spelas in med frekvensskift och med en hastighet av 300 baud.

Se principschemat i fig 16:14.

KCS-modemets trimning består i att R23 justeras för 280 μ S pulsbredd på IC3 stift 10 vid 1200 Hz infrekvens på J1.

Förutsättningen för att terminalen skall koppla in eller ur modem vid angivna kontrollkoder är att TAPE-dioden är tänd. Den tänds genom att trycka på F1 och Å samt släcks genom att trycka på F1 och Ö.

Om terminalen har försetts med ett modem, ansluts en kassettspelare till J1. Till denna kontakt kan dessutom två reläer eller transistorer för automatisk start och inspelningsomkoppling anslutas. På stift 7 finns i terminalen en transistor som leder då bandspelaren skall startas. Det sker då kontrollkoden \$11 eller \$12 mottages i ASCII-mod. På stift 6 finns en annan transistor som leder då bandspelaren skall stå i inspelningsläge. Detta är fallet då koden \$12 (Punch on) mottages. Båda dessa utgångar stängs av då koden \$13 eller \$14 mottages. Programmet är utfört så, att terminalen efter mottagandet av

tecknen \$11 eller \$12 väntar ca två sekunder innan nästa tecken tas om hand. Denna fördröjning är till för att bandspelaren skall komma igång ordentligt. Funktionen förutsätter att DTR-ledaren är inkopplad. Data till bandspelaren matas ut med 300 baud oavsett vilken hastighet som datorn sänder med. Dataflödet från datorn kommer därför att bromsas så att teckenutmatningen till bandspelaren blir i rätt hastighet.

Programstyrning

De sex lysdioderna på tangentbordet indikerar olika drifttillstånd hos terminalen. Av dessa kan bland annat TXT-funktionen styras av tecken från datorn (shift in/shift out). En ytterligare funktion som kan styras från datorn är positioneringen av markören. För att placera markören på godtycklig plats på skärmen fordras tre tecken från datorn. Det första är \$1B (escape-koden) som instruerar terminalen att det är markörpositionering på gång. Det andra är ett tecken med värden mellan \$40 och \$57 som talar om vilken rad markören skall placeras på. Radnummer anges som ett tal mellan 0 och 23 och skall vara i hexadecimal form. Det tredje tecknet är ett tal som anger på vilken position inom raden som markören skall ställas på. Talet \$40 anger vänstra kanten, och \$67 anger den högra kanten. Om felaktiga värden inmatas, påverkas inte markören.

Eftersom alla väsentliga funktioner kan programstyras, går det utmärkt att göra program som ritar upp diagram, staplar, kurvor och text i olika färger.

Kontrollkoder

Nedan följer en beskrivning av terminalens olika kontroll- och styrkoder. De funktioner som styrs direkt från tangentbordet anges som exempelvis F1+E, vilket betyder att tangent F1 och tangent E skall tryckas ner samtidigt. De funktioner, som även kan styras från datorn (via J2), anges med motsvarande hexadecimalkod inom parentes, exempelvis (1B+05). De

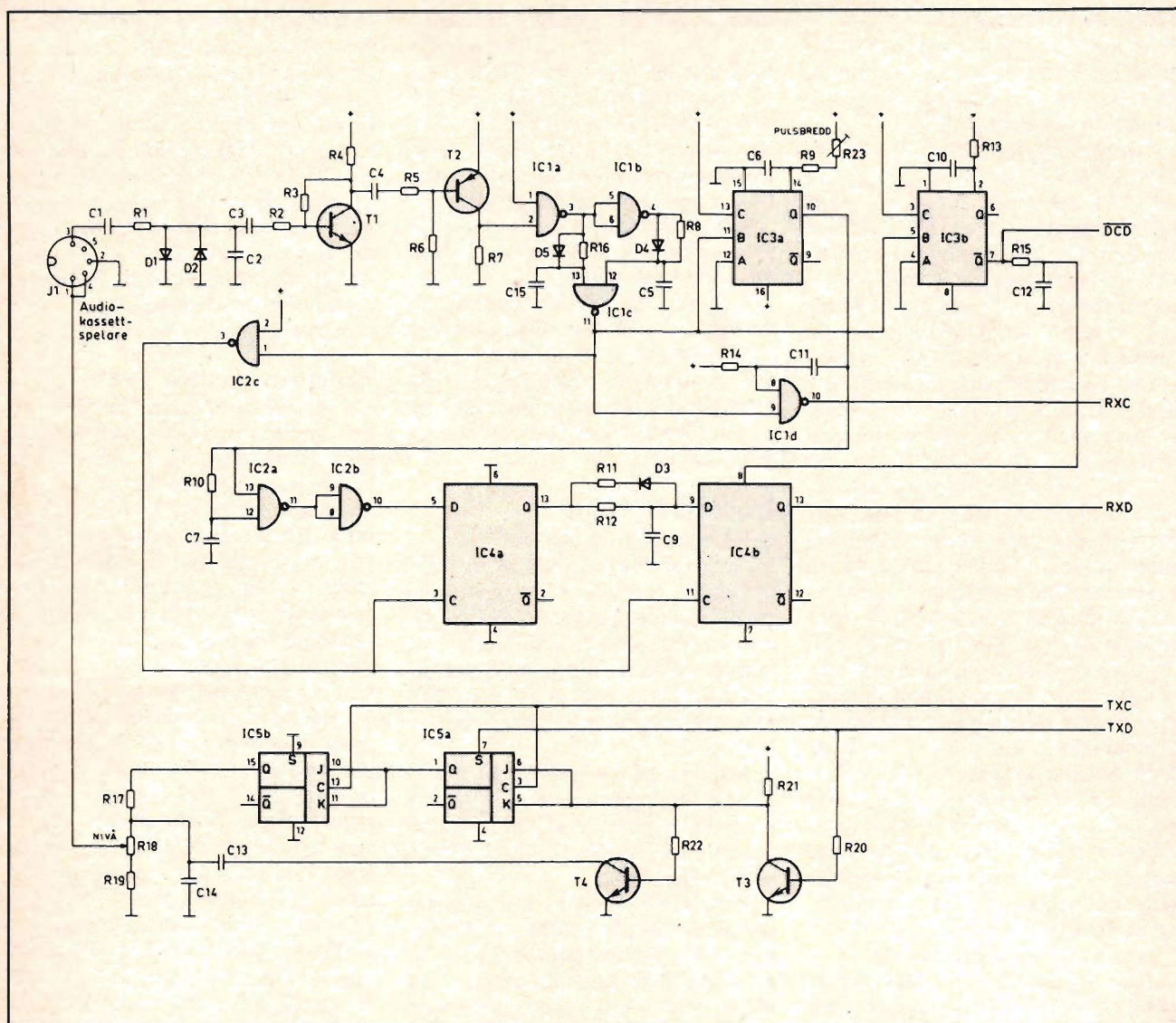


Fig 16:14. Principschema för KCS-modemet till videoterminalen.

flesta funktioner fordrar två tecken från datorn, vilka måste komma i en följd utan tecken emellan.

F1+A, on line med full duplex.

F1+CTRL+A, on line med halv duplex, dvs alla tecken kommer upp direkt på skärmen samtidigt som de sänds iväg.

F1+B, lokal utskrift, lysdioden on line släcks.

F1+C, ($IB+03$), radera från markörens position till slutet på samma rad.

F1+D, ta bort tabulatorstopp på markörens position.

F1+E, ($IB+05$), radera hela bilden

och placera markören i övre vänstra hörnet.

F1+F, flytta markören till nästa tabulatorstopp.

F1+G, sänd den rad på vilken markören står, från vänsterkanten till markören.

F1+H, ($IB+08$), flytta markören ett steg åt vänster.

F1+CTRL+H, (08), flytta markören ett steg åt vänster (backspace).

F1+I, ($IB+09$), flytta markören ett steg åt höger.

F1+J, sätt in ett tecken på markörens plats. Alla tecken till höger om markören flyttas ett steg åt höger. Raderna under påverkas ej.

F1+K, ($IB+0B$), flytta markören upp en rad.

F1+L, ($IB+0C$), flytta markören till övre vänstra hörnet (cursor home)

F1+M, ta bort den rad som markören står på. Alla rader under flyttas upp ett steg.

F1+N, ($IB+0E$), printer fränkoppad.

F1+O, ta bort det tecken, som markören står på. Alla tecken till höger om markören flyttas ett steg till vänster.

F1+P, ($IB+10$), printer till. Lysdioden PRT tänds. Fordrar att ACIA för printer finns inkopplad

Komponentförteckning

C17, 34	18 nF pol.	IC29	MC 14013	R68, 69	10 ohm
C16	0, 47 μ F tantal	IC30	MK 4118	R70-71	10 k
C18	22 μ F 40 V	IC31	74LS377	(R73	100 ohm 1 W)
C19	47 μ F tantal	IC32	SAA 5052	T5-11, 17	BC 558 el motsv.
C20-		IC33	SAA 5020	T12-16,	BC 548 el motsv.
21	22 pF keram	IC34	MCM 6810	18	
C28-33,		IC35	2651	X1	kristall 12,000 MHz
35-41	10 μ F tantal	IC36	MC 6821	X2	kristall 10,1376 MHz
D6-9	1N4148	IC37	MC 1489	2	IC-hållare 8 pin
(D10	5,1 V 1W zener-diod)	IC38	MC 1488	11	IC-hållare 14 pin
IC6-7	MC1455 el 555	IC39	74LS86	11	IC-hållare 16-pin
IC9	MC 6800	J2	25-pol D-kontakt	1	IC-hållare 20-pin
IC10	74LS02	J3	6-pol. DIN-kontakt	4	IC-hållare 24-pin
IC11	74LS14	R24	1M	2	IC-hållare 28-pin
IC12	74LS10	R25, 82	47 k	2	IC-hållare 40-pin
IC13,		R26	22 k	1	kretskort CÅ-790
18-19	74LS367	R27	33 ohm	2	skruvar ECS 3x8
IC14-15	74LS161	R28-33,		2	muttrar M3
IC16	MC 14040BCP	63-66	220 ohm		
IC17	74S00	R34, 39,			
IC20	Prom CÅ-6302 (röd)	46-48	6,8 k		
IC21	Prom CÅ-6301 (svart)	R35-38	27 k		
IC22	MC 14050	R40, 41, 56	100 ohm		
IC23	74LS93	R42-45	1 k		
IC24	MC 14022	R49-50,			
IC25	Eprom CÅ-2804	72	680 ohm		
IC26	74LS139	R51-55,	2,2 k		
IC27	74LS00	57-58,			
IC28	74LS74	67,			
		R59-61	330 ohm		
		R62	75 ohm		

VHF-delen:

C24	10-40 pF ker. trim
C25-27	4,7 nF ker.
IC40	LM 1889
J4	2-pol. antennkont.
L1	0,47 μ H drossel
R74	2,2 k
R75	10 k
R76	75 ohm
R77	100 ohm
R78-79	220 ohm
1	IC-hållare 18-pin

till sockeln J5.

(IB+OF), koppla om terminalen till grafikmod, lysdioden TXT tänds.

F1+Q, sänd hela sidan till datorn. Om terminalen är ASCII-mod, sänds efter varje teckenrad automatiskt en CR (\$0D) och en kort paus. Om terminalen är i grafikmod sänds alla teckenpositioner på varje rad utan paus eller CR.

F1+CTRL+Q, sänd hela sidan till det inbyggda KCS-modemet. Terminalen skall vara i grafikmod, och TAPE-dioden skall vara tänd.

F1+R, (IB+01), page mode, utskriften börjar om i övre vänstra hörnet, när en sida är fullskriven.

F1+CTRL+R, fyll sidan med data från KCS-modemet, terminalen skall vara i grafikmode och TAPE-dioden tänd. Om någon tangent på bordet nertrycks under det att bandet går och bildskärmen fylls så avbryts förloppet.

F1+S, sätt in en tabstopp.

F1+T, (IB+36), tvåradsscroll

F1+U, (IB+20), fullsidesscroll

F1+V, sätt in en ny rad där markören står. Alla rader under flyttas ett steg nedåt och understa raden försvinner.

F1+W, (IB+17), flytta markören nedåt ett steg.

F1+%, (IB+15), släck markören

F1+\$, (IB+02), tänd markören
 F1+Y, (IB+2C), halvssidesscroll
 F1+Å, KCS-modem inkopplat
 F1+Ö, KCS-modemet urkopplat
 F1+Ä, ta bort alla tabstoppar
 F1+X, återställ normalparametrar (9600 baud, ingen paritet, ASCII-mod, printer från, KCS-modem från, markören tänd m m)
 (07), bell-ton, (IB+07 i grafisk mod)
 (0A), line feed, radmatning, (IB+0A i grafisk mod)
 (0D), return, vagnretur, markören återgår till vänsterkanten (IB+0D i grafisk mod)
 (11), start av KCS-modem avspel-

ning (TAPE måste vara tänd) (1B+11 i grafisk mod) (12), start av KCS-modem inspelning (TAPE måste vara tänd) (1B+12 i grafisk mod) (13) eller (14), stopp av KCS-modem. (1B+13 resp. 1B+14 i grafisk mod).

F1+!, ingen paritet
F1+”, jämn paritet
F1+#, udda paritet

F1+2, 110 baud
F1+3, 135 baud
F1+4, 150 baud
F1+5, 300 baud
F1+6, 600 baud
F1+7, 1200 baud
F1+8, 1800 baud
F1+9, 2000 baud
F1+., 2400 baud
F1+;, 3600 baud
F1+<, 4800 baud
F1+=, 7200 baud
F1+>, 9600 baud

F1+É, sänd från bildens början till markörens position.

(1B+19), spara markörens position. Denna kod avbryter pågående utskrift och sparar markörens position. Man kan därefter med andra kontrollkoder flytta markören till valfritt ställe på bildskärmen, ex.vis översta raden och där skriva något till exempel klockslag med sekundangivelse. Med koden (1B+16) fortsätter sedan den tidigare avbrutna utskriften som om ingenting hade hänt. (1B+16), återtag markörens position.

(1B+xx+yy), flytta markören till valfri plats. xx skall ha ett värde mellan \$40 och \$57 och anger radnummer (00–23) fast i hexadecimal form. yy skall ha ett värde mellan \$40 och \$67 och anger position på raden.

(1B+ss), kod för scrollning. ss skall ha ett värde mellan \$20 och \$37. \$20 motsvarar fullsidesscroll och \$36 motsvarar tvåradsscroll.

Terminalen har som synes många funktioner som kan styras från datorn. Med möjligheten att återge bilder i färg och med grafiska tecken av olika slag har RT-datorn nu fått en mycket kraftfull terminal.

Till tangentbordet krävs:

D11–12	gul lysdiod	C11	1 nF keram
D13–15	grön lysdiod	C12	1 µF tantal
D16	röd lysdiod	D1–5	1N4148
D17	1N4148	IC1–2	4011 el 14011
H	högtalare	IC3	4528 el 14528
J6	24-pol kortkontakt	IC4	4013 el 14013
K1	Flatkabel med kontakter 20-pol.	IC5	4027 el 14027
S1	Omkopplare 1-pol växl. med nolläge. För kortmontage.	IC8	MC 6850
1	tangentbord Keytronic L1980	J1	7-pol. DIN-kontakt
1	tangent "alpha lock" till dito	R1, 14	3,3 k 5 % 1/8 W
1	låda	R2, 4, 7, 9	10 k
1	kretskort CÅ-803	R3, 13	220 k
2	skruvar ECS 3×40	R5, 20, 22	22 k
2	skruvar ECS 3×25	R6	1 M
12	muttrar M3	R8, 16,	2,2 k
KCS-modemet:		80–81	2,2 k
C1, 3–4,	0,22 µF pol.	R10	8,2 k
10		R11–12,	120 k
C2, 13–14	22 nF pol	15, 17	
C5, 15	1 nF pol.	R18, 23	100 k trimpot
C6	8,2 nF pol.	R19	1 k
C7, 9	10 nF pol.	R21	4,7 k
C8	10 µF 16 V tantal	T1, 3–4,	
		18, 19	BC 548 el motsv.
		T2	BC 558 el motsv.
		3	IC-hållare 14-pin
		2	IC-hållare 16-pin
		1	IC-hållare 24-pin

Tabell 10:1. Inkopplingen av stiften på den 25-poliga kontakten J2.

Stift nr	Signal-beteckning	Funktion och anslutning
1	GND	jord
2	TXD	sändningsdata från terminal till datorn
3	RXD	mottagningsdata från datorn till terminalen
4	RTS	request to send. Sändningsbegäran, används inte f n
5	CTS	clear to send, ingång. Måste vara på hög nivå för att terminalen skall kunna sända data till datorn
6	DSR	data set ready, ingång. Används inte f n
7	GND	jord
8	DCD	data carrier detect, ingång. Måste vara på hög nivå för att terminalen skall kunna ta emot data
9	+12	utgång, sammankopplas med CTS och DCD i de fall dessa signaler ej kommer från datorn
20	DTR	data terminal ready, utgång. Hög nivå indikerar att terminalen är redo att ta emot tecken från datorn. Låg nivå indikerar att terminalen är upptagen med tecken eller att den är i lokal mod.

TV:n kan användas som dataterminal

AV ÅKE HOLM

Att bygga en videomonitor blir oftast dyrare än att köpa en färdig TV av lämpligt slag. Vi skall här ge några tips om hur man kan anpassa sin TV till RT-datorn.

■ Den bästa lösningen på problemet att åstadkomma en videomonitor är att skaffa en färg-TV med kallt chassi som har videoingång av RGB-typ som standard. Med detta förfarande slipper man göra ändringar i TV-mottagaren, ändringar som sätter S-märkningen ur spel och därmed kanske ökar riskerna för den som använder TV:n. Det senare gäller speciellt då TV:n inte enbart används till datorn utan även skall fungera som andra-TV och skötas av de övriga familjemedlemmarna.

Separat chassi från nätet

Nöjer man sig med en svartvit datamonitor kan en liten TV med 12

tums bildskärm avsedd för batterimatning användas. TV:n bör dock även kunna matas med 220 volt och den måste *absolut* ha skyddstransformator eller annan typ av nåtdel, som separerar chassiet från nätspänningen. Eftersom vi skall använda TV:n som *video*-monitor måste signalen från videoterminalen kopplas till TV:ns videoförstärkare på något sätt. Oftast räcker det att via en kondensator gå till basen på videoförstärkarens utgångstransistor. Videodetektorn i TV:n skall kopplas bort, för att man inte skall få brus blandat med texten från terminalen.

Om man har en färg-TV med kallt chassi och vill göra en RGB-

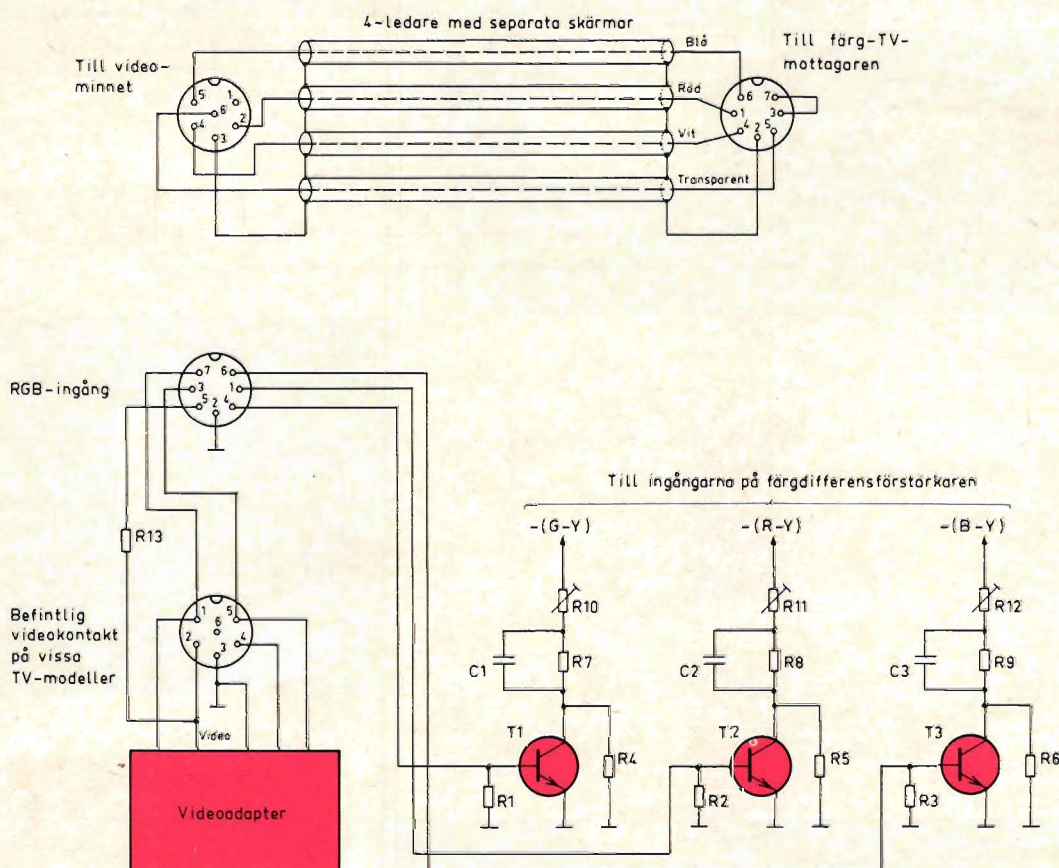


Fig 17:1. Principschema för RGB-ingång på en färg-TV med isolerat chassi.

ingång på denna, kan man använda kopplingen i fig 17:1. De flesta moderna färg-TV-mottagarna har en krets med beteckningen TBA530 el motsvarande. Signalerna från RGB-ingången kan då anslutas enligt fig 17:2. Har TV:n inte kallt chassi, dvs chassiet är nätspänningsförande, kan ett par optokopplare enligt fig 17:3 användas istället.

Till vissa färg-TV-mottagare finns videoingång att köpa som tillbehör. Hör efter hos TV-handlaren om detta finns till den TV som skall användas. TV:n bör dock inte vara alltför gammal, eftersom videoingång på TV är någonting relativt nytt inom TV-branschen.

Om det råder tveksamhet om hur en TV skall modifieras är det säkrast att fråga någon till råds. Ett felaktigt ingrepp i en TV kan faktiskt vara livsfarligt. Kom även ihåg att en TV som är ombyggd inte längre är S-märkt och alltså ej får säljas utan vidare. Skulle det bli aktuellt att sälja den måste den återställas exakt i sitt ursprungliga skick så att alla S-märkningskraven uppfylls. ■

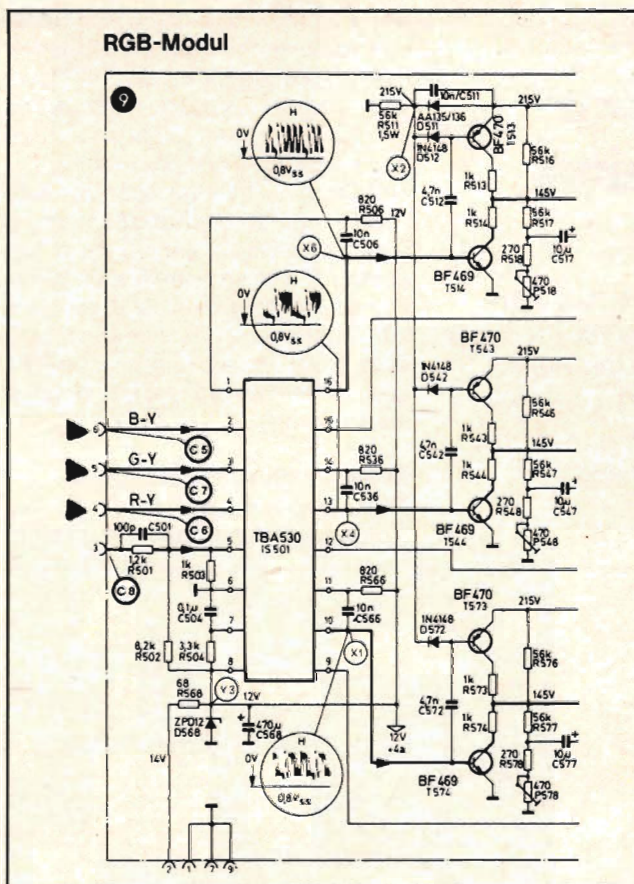


Fig 17:2. Inkopplingen av RGB-signalerna i en modern färg-TV.

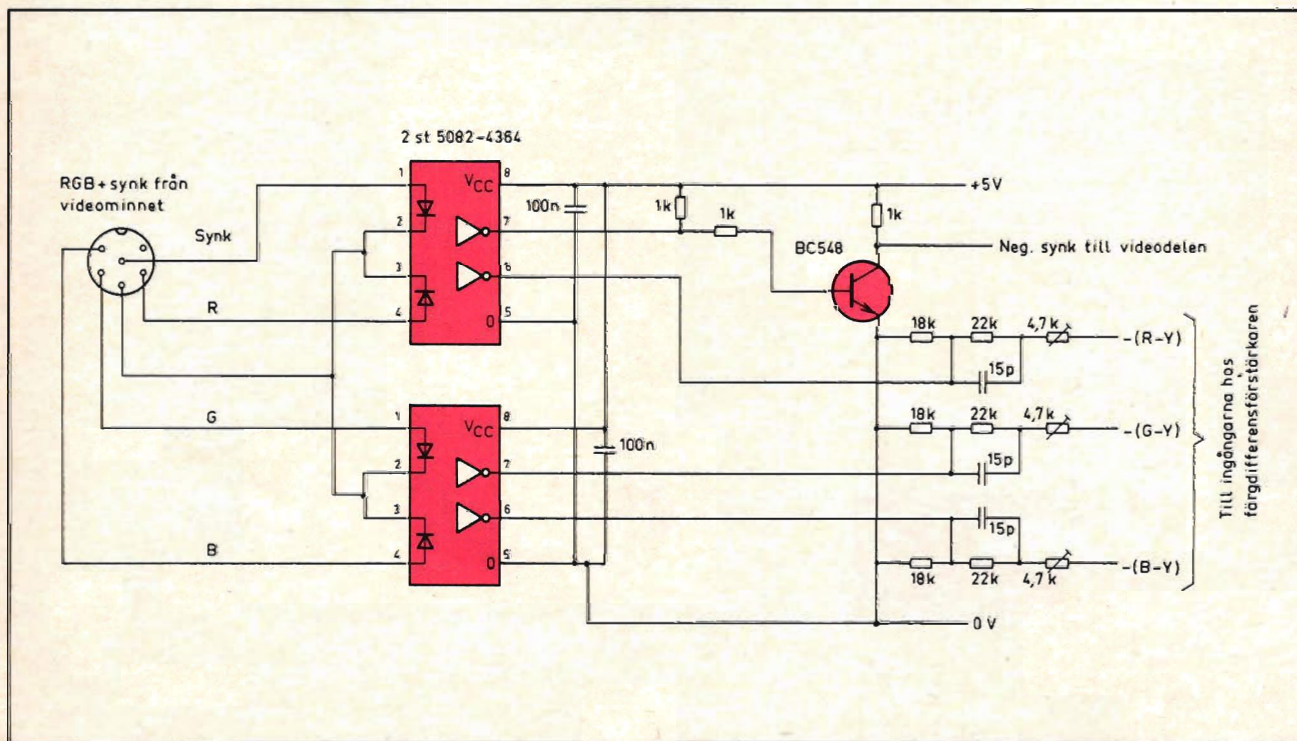


Fig 17:3. Principschema för RGB-ingång med optokopplare.

Programmerbara räknare på ett periferikort

AV ÅKE HOLM

Tre sextonbitars programmerbara räknare innehåller den krets, som här monteras på ett periferikort.

■ I vissa sammanhang behöver man använda en programmerbar räknare av något slag. Kretsen MC6840 är en sådan. Den innehåller tre 16-bitars programmerbara räknare, vilka kan användas på ett flertal olika sätt. Blockschemat för MC6840 återfinns i *fig 18:1*, och en fullständig beskrivning av kretsens alla funktioner kan inhämtas från Motorolas datablad över MC6840. I RT-datorn kommer kretsen till användning tillsammans med flexskivesystemet i en funktion kallad "printer spooling".

Det innebär, att en till datorn ansluten skrivare kan fås att skriva ut listor från textfiler på flexskivan samtidigt som man kan editera program eller köra ett basic-program, oberoende av skrivaren. Det fungerar genom en rutin i FLEX 2.0 (eller FLEX 9.0) kallad PRINT. Rutinen initierar tidkretsen så att den ger ett periodiskt avbrott, under vilket skrivarens utskriftsprogram tas om hand. Förutsättningen för funktionen är att det program som man själv kör inte använder IRQ-vektorn, samt att ingen åverkan får göras på de textfiler vilka skall skrivas ut på skrivaren. Det förutsätts också att skrivaren är ansluten till en egen ACIA på kortet 8001. En beskrivning av hur printer spooling startas och används medföljer dokumentationen till flexskiveprogrammen FLEX 2.0 resp FLEX 9.0.

Kretskortet återges i *fig 18:3* med komponentplacering. I *fig*

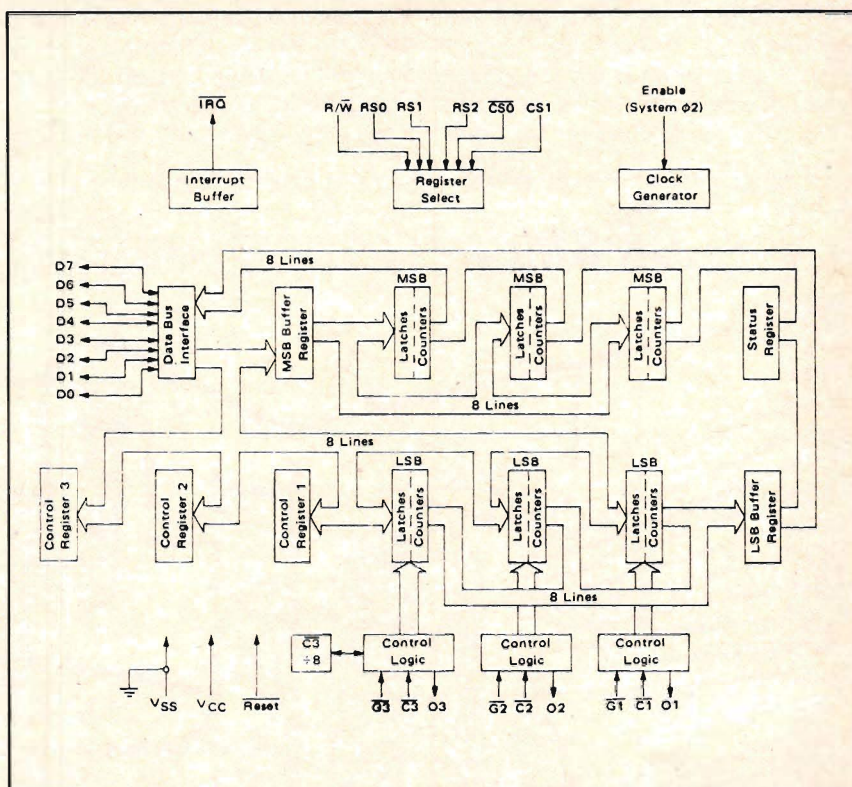


Fig 18:1. Blockschemat för den programmerbara räknaren MC6840.

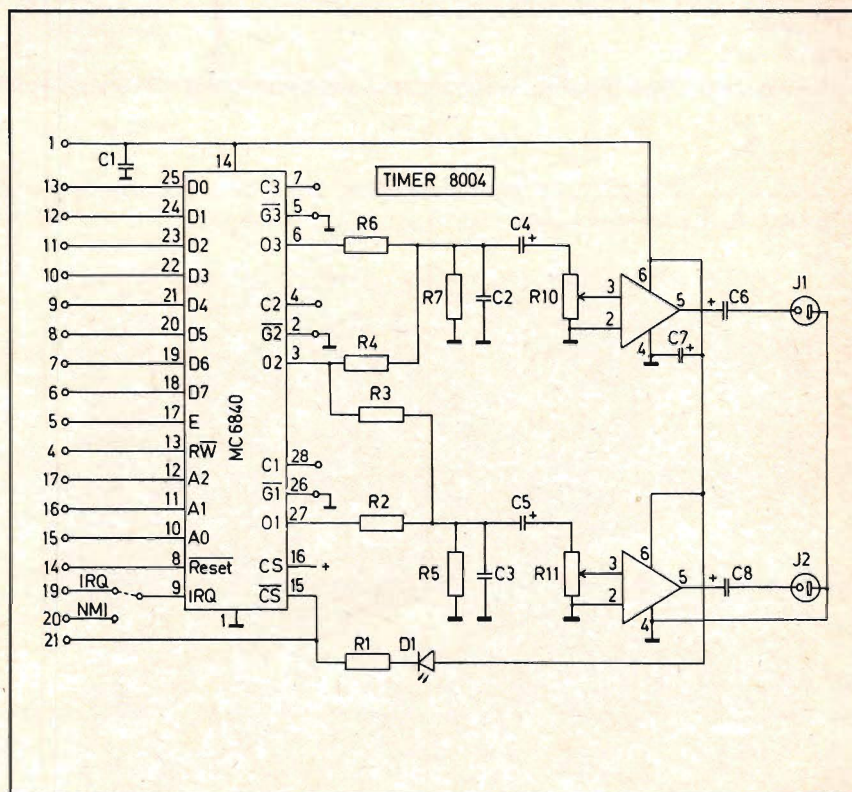


Fig 18:2. Principschema för RT-datorns timerkort.

18:2 återfinns principalschemat för RT-datorns timerkort med MC6840. Schemat är ganska enkelt och fordrar ingen större beskrivning. Till utgångarna O1–O3 finns

möjlighet att ansluta två LF-förstärkare för de tillämpningar, där man vill använda kretsen för tonalstring för att skapa musik.

De tre ingångarna G1–G3 på

MC6840 skall jordas vid normal drift. Detta fordrar att en bygel skall lödas in vid G1. För G2 och G3 finns foliebyglar på kortets undersida.

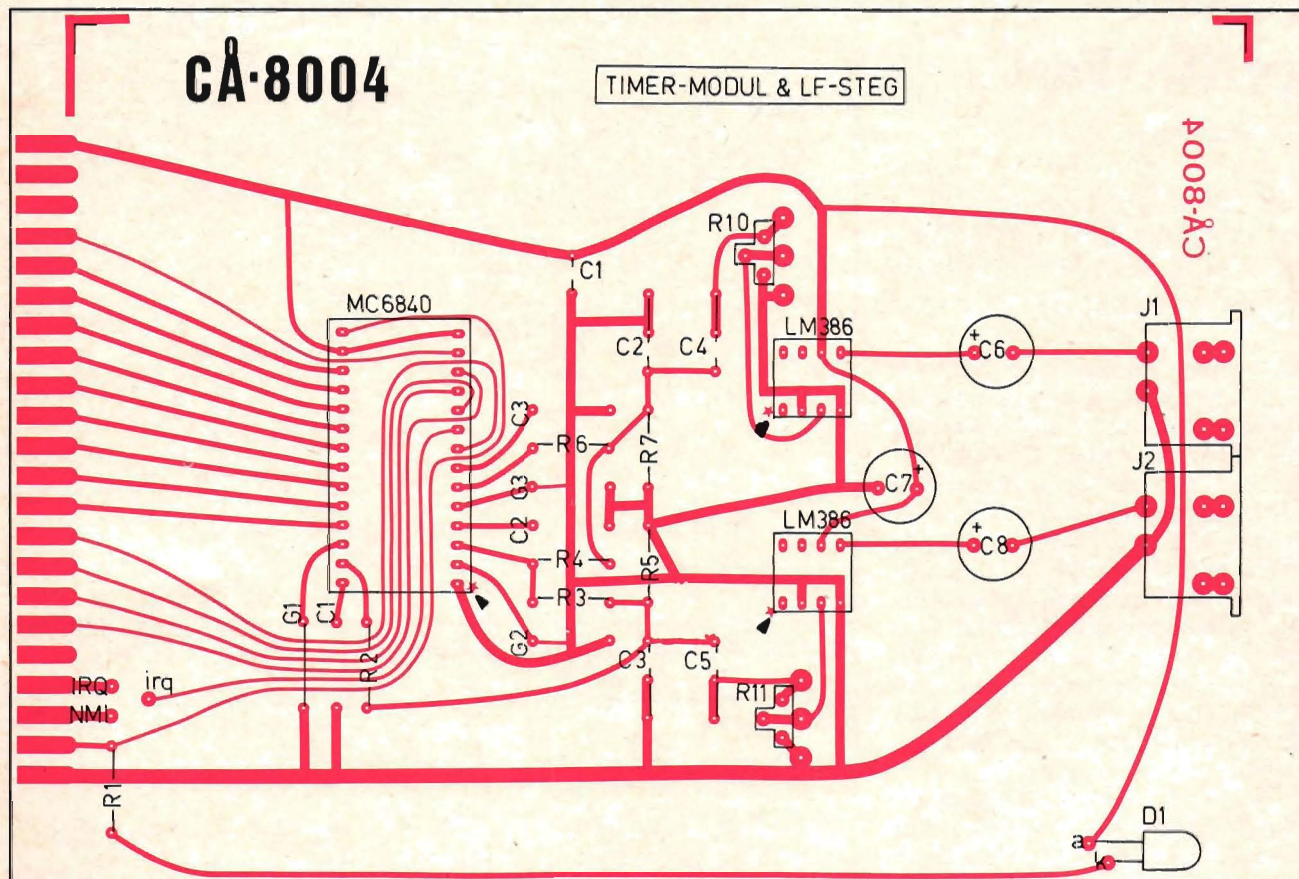


Fig 18.3. Kretskort med komponentplacering för timerkortet.

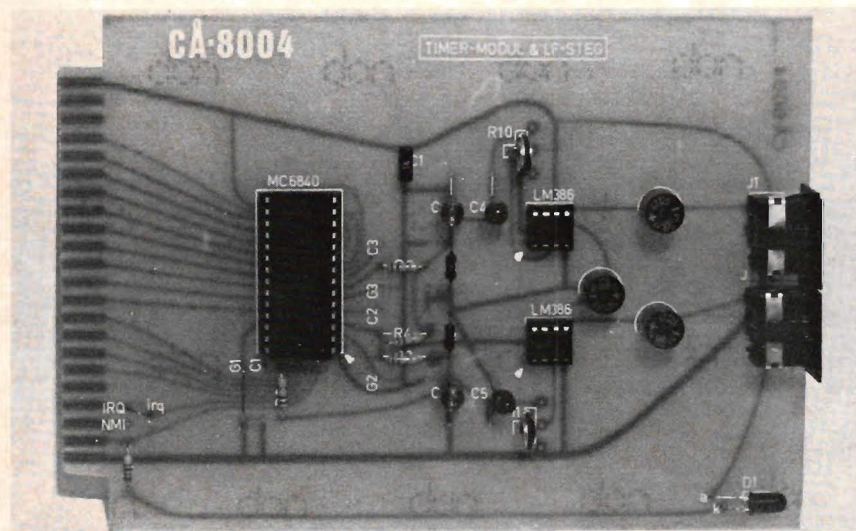


Fig 18.4. Bilden visar ett monterat kretskort.

Komponentförteckning:

C1	0,1 μ F polyester
C2–3	2200 pF pol.
C4–5	10 μ F tantal
C6–8	47 μ F 25 V el.lyt.
D1	lysdiod
IC1	MC 6840 (Motorola)
IC2–3	LM 386N (NS)
J1–2	2-pol. DIN-kontakt för kort
R1	220 ohm 1/8 W 5 %
R2–4, 6	220 k
R5, 7	10 k
R10–11	10 k trimpot stående
2	IC-hållare 8-pin
1	IC-hållare 28-pin
1	kretskort CA-8004

Mångsidiga labkort

AV ÅKE HOLM

Säkert kommer man att behöva ett specialkort av något slag. Det här labkortet tillåter egna kopplingar anslutna till datorns buss via parallell eller seriesnitt, dvs med kretstyperna PIA och ACIA.

■ För att möjliggöra för datorbyggaren att prova egna periferikopplingar finns två olika labkort framtagna. De är ganska lika till utseendet, men det ena är avsett för en

parallellsnittskrets (PIA) och det andra för en seriesnittskrets av typ ACIA, SSDA eller ADLC. ACIA står för Asynchronous Communications Interface Adapter och är den vanligaste kretsen för asynkron serieöverföring av data. SSDA betyder Synchronous Serial Data Adapter och används som sagt för synkron serieöverföring av data. ADLC betyder Advanced Data Link Controller och är en mycket speciell krets för dataöverföring med hög hastighet.

De båda korten har färdiga för-

bindelser mellan den 22-poliga kontakten och de nämnda kretstyperna. Båda korten har plats för en extra avkodare av typ 74LS138 om så önskas. Vidare finns en lysdiod för adressindikering samt en kortkontakt för anslutning till omvärlden. I fig 19:1 finns placeringsritning för PIA-labkortet och i fig 19:2 för det kombinerade ACIA/SSDA/ADLC-kortet.

Eftersom ACIA:n är den vanligaste kretsen av dessa tre, är kortet i första hand gjort för denna. SSDA-kretsen har nästan samma

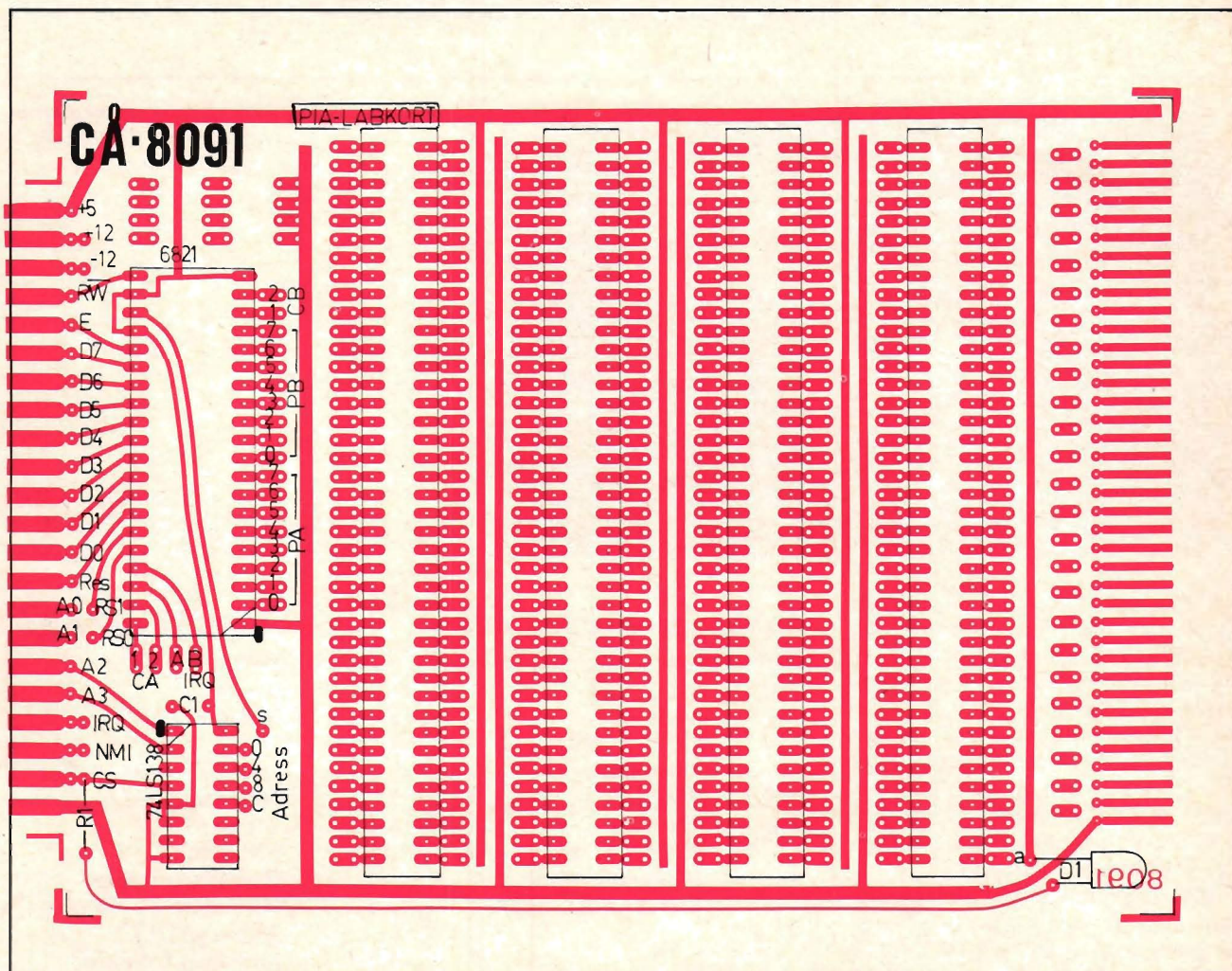


Fig 19:1. Kretskort med komponentplacering för labkortet med PIA-krets. Ett exempel på hur en sådan programmeras finner vi i avsnittet om assemblerprogram.

Analog-digitalomvandlare

AV ÅKE HOLM

Med en A/D-omvandlare kan datorn ta hand om och bearbeta analoga signaler från olika givare.

■ För att kunna ta hand om analoga signaler måste RT-datorn försees med en A/D-omvandlare. Den omvandlare som här beskrivs arbetar enligt principen succesiv approximation och har 16 ingångar. Mätvärdet får man som ett åttabits värde och det blir alltså 256 steg mellan noll och fullt utslag. I *fig 20:1* återges principalschemat. Periferikortet innehåller en PIA-krets (IC1) som ansluter A/D-omvandlarkretsen till mikroprocessorn. IC3 är klockfrekvensoscillator och IC4 är en extra adressavkodning som normalt ej behövs. De sexton

ingångarna har försetts med var sin trimpotentiometer för att individuellt kunna justera lämpligt värde och få anpassning till de olika inspänningarna.

Man kan exempelvis ansluta en skjoy-stick till två av ingångarna på A/D-omvandlaren. I *fig 20:3* visas hur en sådan inkopplas och i *fig 20:5* finns ett tillämpningsprogram för detta. Detta program plockar in två analoga signaler och återger dessa som två tal mellan 0 och 255 på de sex lysdiodindikatorerna. Programmet innehåller en rutin, som omvandlar från hexadecimalt (00-FF) till decimalt (0-255). Programmet kan knappas in med det lilla terminalkortet och användas som övningsprogram. A/D-omvandlingen startas genom att ut-

gången CB2 ger en hög puls till IC2. När IC2 har mätt klart, ger utgången EOC (stift 13) en puls till CB1 på PIA-kretsen. Genom att i programmet då testa på en flagga i kontrollregistret kan man avgöra, när en mätning är klar.

Det visade programmet kan modifieras för att återge de insamlade värdena på en bildskärm. Genom att då använda videoterminalens möjligheter till styrning av skrivmarkören kan mätvärdena skrivas på valfritt ställe av bildskärmen. Man kan till och med skriva dem i olika färger och sätta upp gränsvärden för insignalen. Detta kan vara värdefullt, om datorn skall övervaka ett antal spänningar. Blir någon för stor blinkar den med röd färg exempelvis. ■

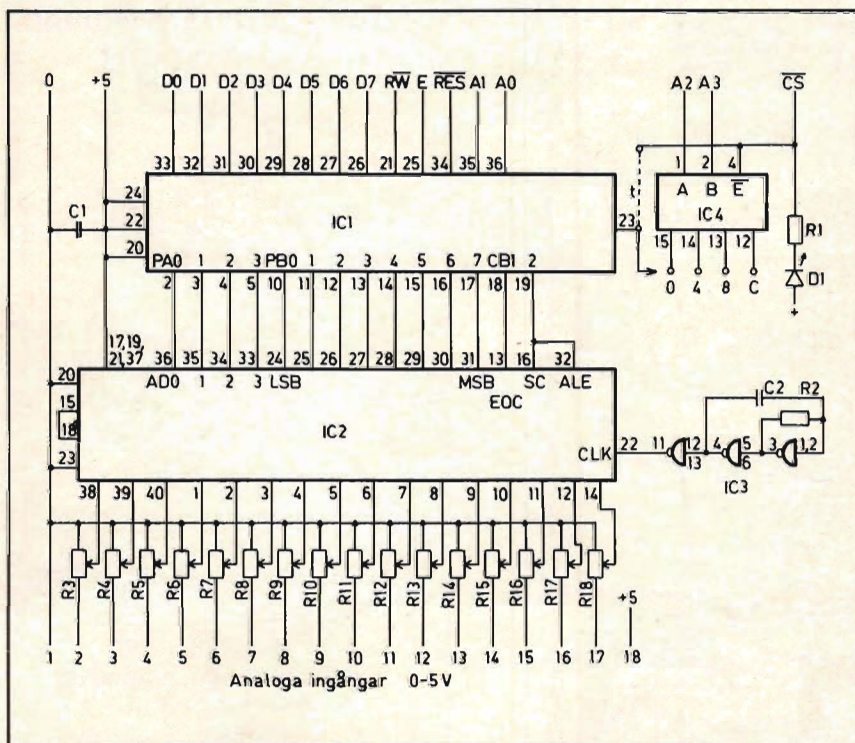
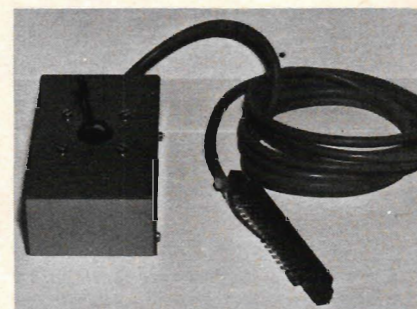
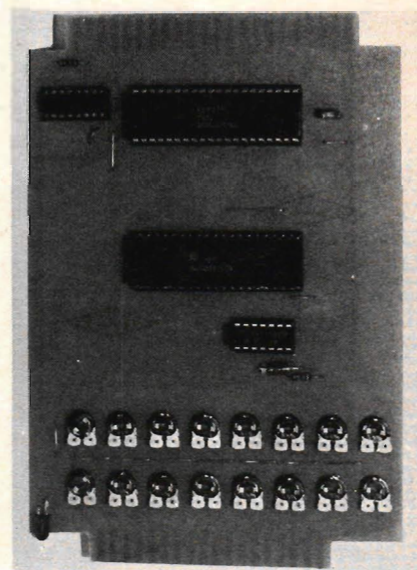


Fig 20:1. Principalschemat för A/D-omvandlaren.

Fig 20:2. Bilden visar ett monterat kretskort (prototypkort utan komponenttryck). Nedan till höger ser vi en monterad "joy-stic" med kabel och kontakter.



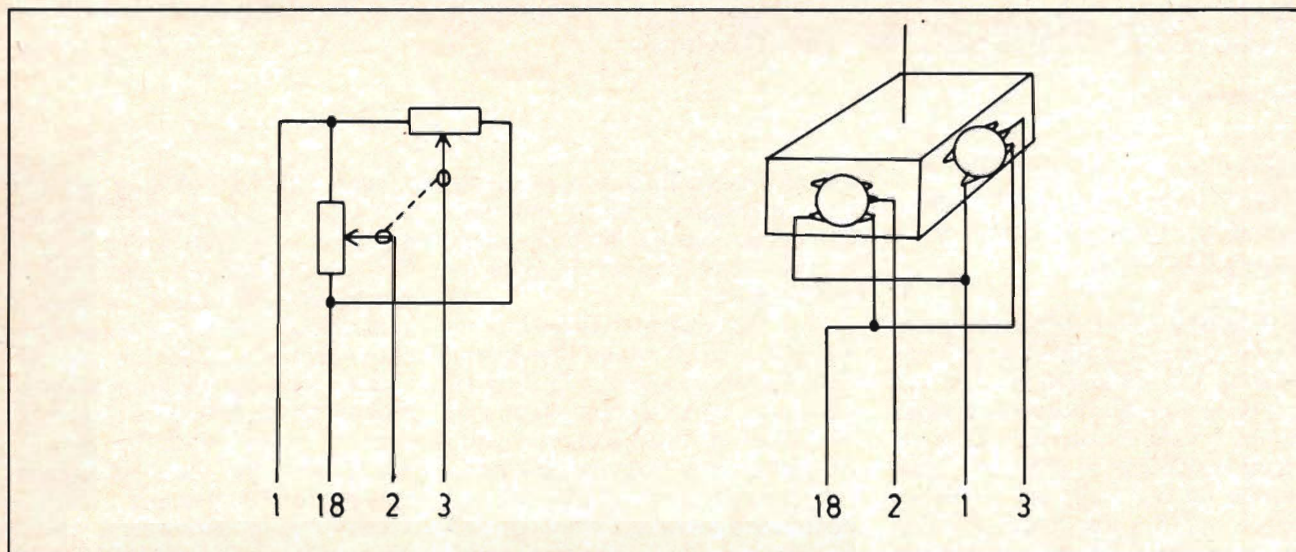


Fig 20:3. Schema för inkoppling av joy-stick. Till höger ser vi hur de olika anslutningarna på potentiometern ansluts.

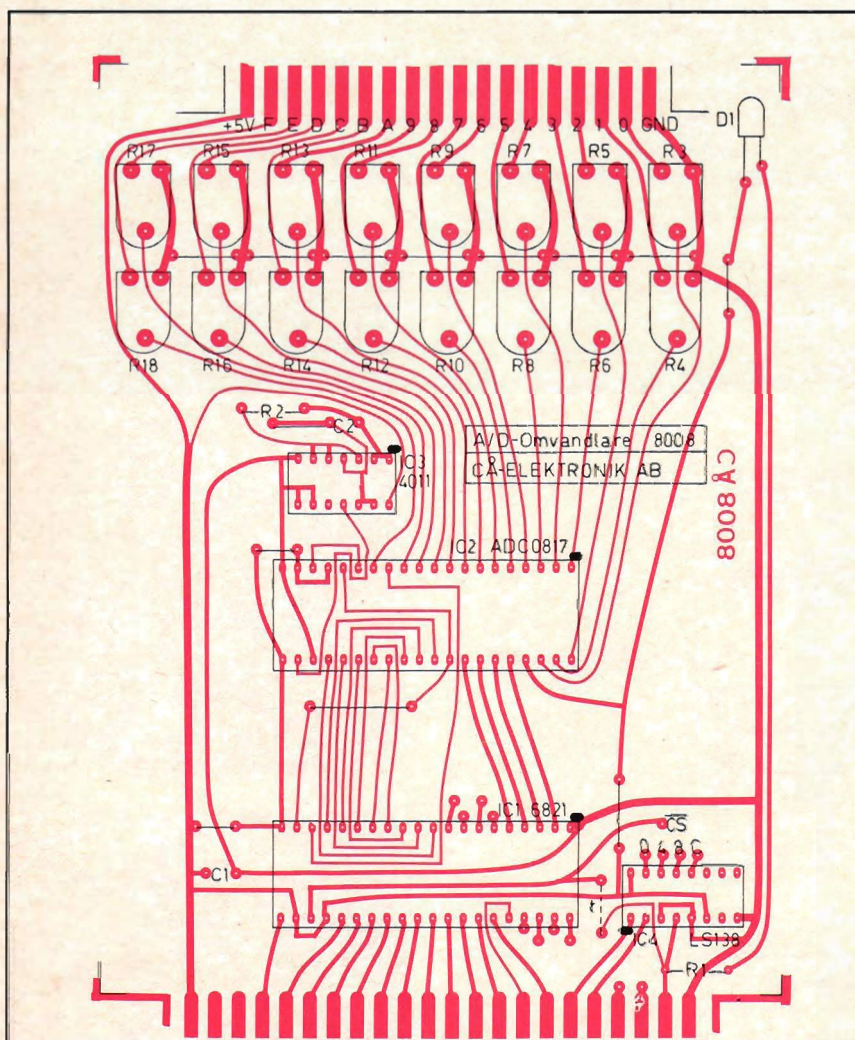


Fig 20:4. Kretskortet sett från komponentnsidan med ledningsmönstret i färg och komponenttrycket i svart.

Komponentförteckning:

C1	0,1 μ F polyester
C2	220 pF styrol
D1	röd lysdiod
IC1	6821
IC2	ADC 0817 el ADC 0816
IC3	4001 eller 4011
(IC4)	(74LS138)
R1	220 ohm
R2	4,7 k
R3-18	100 k trimpot
1	kretskort CÅ-8008
1	IC-hållare 14-pol.
2	IC-hållare 40-pol.

Komponentförteckning: joy-stick:

1	låda Telko A92
1	X-Y-potentiometer med knapp
1 m	4-ledare
1	18-pol. kortkontakt
4	skruvar ECS 3×5

Fig 20:5. Testprogram för A/D-omvandlaren och det lilla terminalkortet.

```

1  *
2  *
3  *
4  *
5  *
6  *
7  *
8  *
9  *
10 *
11 *
12 *
13 *
14 *
15 3080 PIAAD0 EQU #8080 DATAREGISTER ADRESSVAL
16 3081 PIAAD1 EQU #8081 KONTROLLREG. DITO
17 3082 PIAAD2 EQU #8082 DATAREGISTER MATVARDE
18 3083 PIAAD3 EQU #8083 KONTROLLREG. DITO
19 30E0 DISPL EQU #80E0 ADRESS TILL FÖRSTA DISPLAY
20 *
21 A100 * ORG #A100
22 *
23 A100 7F 80 81 START CLR PIAAD1 NOLLA KONTROLLREG.
24 A103 7F 80 83 CLR PIAAD3
25 A106 7F 80 82 CLR PIAAD2 PB0-7 INGANGAR
26 A109 86 36 LDA A #36 CB2 UTGANG, CB1 INGANG
27 A10B B7 80 83 STA A PIAAD3
28 A10E 86 0F LDA A #0F
29 A110 B7 80 80 STA A PIAAD0 PA0-3 UTGANGAR
30 A113 86 04 LDA A #4
31 A115 B7 80 81 STA A PIAAD1
32 *
33 A118 5F LOOP00 CLR B NOLLA FÖR INGANG 0
34 A119 CE A1 A3 LDX #BINTAB
35 A11C 8D 28 BSR INPUT HAMTA MATVARDE KANAL 0
36 A11E 8D 3C BSR BCDCON OMVANDLA TILL DECIMALT
37 A120 A6 02 LDA A 2,X
38 A122 B7 80 E0 STA A DISPL LÄGG UT PÅ FÖRSTA DISPLAYEN
39 A125 A6 03 LDA A 3,X
40 A127 B7 80 E1 STA A DISPL+1 HAMTA 2:1A SIFFRAN
41 A12A A6 04 LDA A 4,X LÄGG UT PÅ 2:1A DISPLAYEN
42 A12C B7 80 E2 STA A DISPL+2 HAMTA 3:1E SIFFRAN
43 A12F C6 01 LDA B #1 LÄGG UT PÅ 3:1E DISPLAYEN
44 A131 8D 13 BSR INPUT HAMTA FRÅN KANAL 1
45 A133 8D 27 BSR BCDCON
46 A135 A6 02 LDA A 2,X
47 A137 B7 80 E3 STA A DISPL+3 LÄGG UT PÅ 4:1E DISPLAYEN
48 A13A A6 03 LDA A 3,X
49 A13C B7 80 E4 STA A DISPL+4 HAMTA 2:1A SIFFRAN
50 A13F A6 04 LDA A 4,X LÄGG UT PÅ 3:1E SIFFRAN
51 A141 B7 80 E5 STA A DISPL+5 HAMTA 3:1E SIFFRAN
52 A144 26 D2 BRA LOOP00 LÄGG UT PÅ 6:1E DISPLAYEN
53 * FORTSATT
54 A146 F7 80 80 INPUT STA B PIAAD0
55 A149 86 3E LDA A #3E GE HÖG FULS PÅ CB2
56 A14B C6 36 LDA B #36
57 A14D B7 80 83 STA A PIAAD3 HÖG NU
58 A150 F7 80 83 STA B PIAAD3 LÅG IDEN
59 A153 B6 80 83 INPUTS LDA A PIAAD3 HAMTA STATUS
60 A156 2A FB BPL INPUTS VANTA TILLS AD-CONV. KLAR
61 A158 B6 80 82 LDA A PIAAD2 HAMTA MATVARDE
62 A15B 39 RTS
63 *
64 *
65 *
66 *
67 *
68 *
69 A15C FF A1 A1 BCDCON STX SAVEX SPARA PEKAREN
70 A15F CE A1 94 LDX #CONST TABELL FÖR KONSTANTER
71 A162 5F CLR B
72 A163 7F A1 9E B10 CLR SAVEA ATT BILÖV DECIMALTECKEN
73 A166 A0 01 B20 SUB A 1,X DRAG IFRÅN TILLS UNDERFLOW
74 A168 E2 00 SBC B 0,X TA MED CARRYIN
75 A16A 25 05 BCS B20
76 A16C 7C A1 9E INC SAVEA
77 A16F 20 F5 BRA B20
78 *
79 A171 AB 01 B30 ADD A 1,X
80 A173 E9 00 ADC B 0,X
81 A175 36 PSH A
82 A176 FF A1 9F STX SAVE0 SPARA AKTUELLT VARDE
83 A179 FE A1 A1 LDA SAVEX VAR SKALL DET LAGRAS
84 A17C B6 A1 9E LDA A SAVEA LÄGG IN DET
85 A17F A7 00 STA A 0,X I TABELLEN
86 A181 32 PUL A
87 A182 08 INX
88 A183 FF A1 A1 STX SAVEX SPARA NYA VARDEN
89 A186 FE A1 9F LDX SAVE0
90 A189 08 INX
91 A18A 08 INX
92 A18B 8C A1 9E CPY #NONEND SLUT PÅ KONSTANTERNA ?
93 A18E 26 05 BNE B10
94 A190 CE A1 A3 LDX #BINTAB PEKA PÅ TABELLEN
95 A193 39 RTS
96 *
97 A194 27 10 CONST FDB 10000
98 A196 03 E8 FDB 1000
99 A198 00 64 FDB 100
100 A19A 00 0A FDB 10
101 A19C 00 01 FDB 1
102 A19E EQU *
103 *
104 A19E RMB 1
105 A19F RMB 2
106 A1A1 RMB 2 ATT SPARA PEKAREN I
107 A1A3 RMB 5 TABELL FÖR RESULTAT
108 *
109 END START

```

NO ERRORS DETECTED

Flexskivesystemet – vägen till avancerad högnivåprogrammering

AV ÅKE HOLM

Med ett flexskivesystem blir vår dator klar för högnivåprogrammering.

Här framgår hur man bygger ett lämpligt anpassningskort för en flexskiveenhet.

■ Med den här etappen av datorbygget är datorn helt klar för avancerad programmering med högnivåspråk. I avsnittet skall vi beskriva kompletteringen med flexskivesystem och därmed är dörren öppen för en enkel programhantering av såväl stora som små program.

Ett flexskivesystem till datorn fyller två stora funktioner. Dels tjänstgör det som massminne med kort åtkomsttid och dels som ett programhanteringssystem. Det senare är helt enkelt vad ett sk operativsystem med tillhörande kommandon kan göra för att förenkla

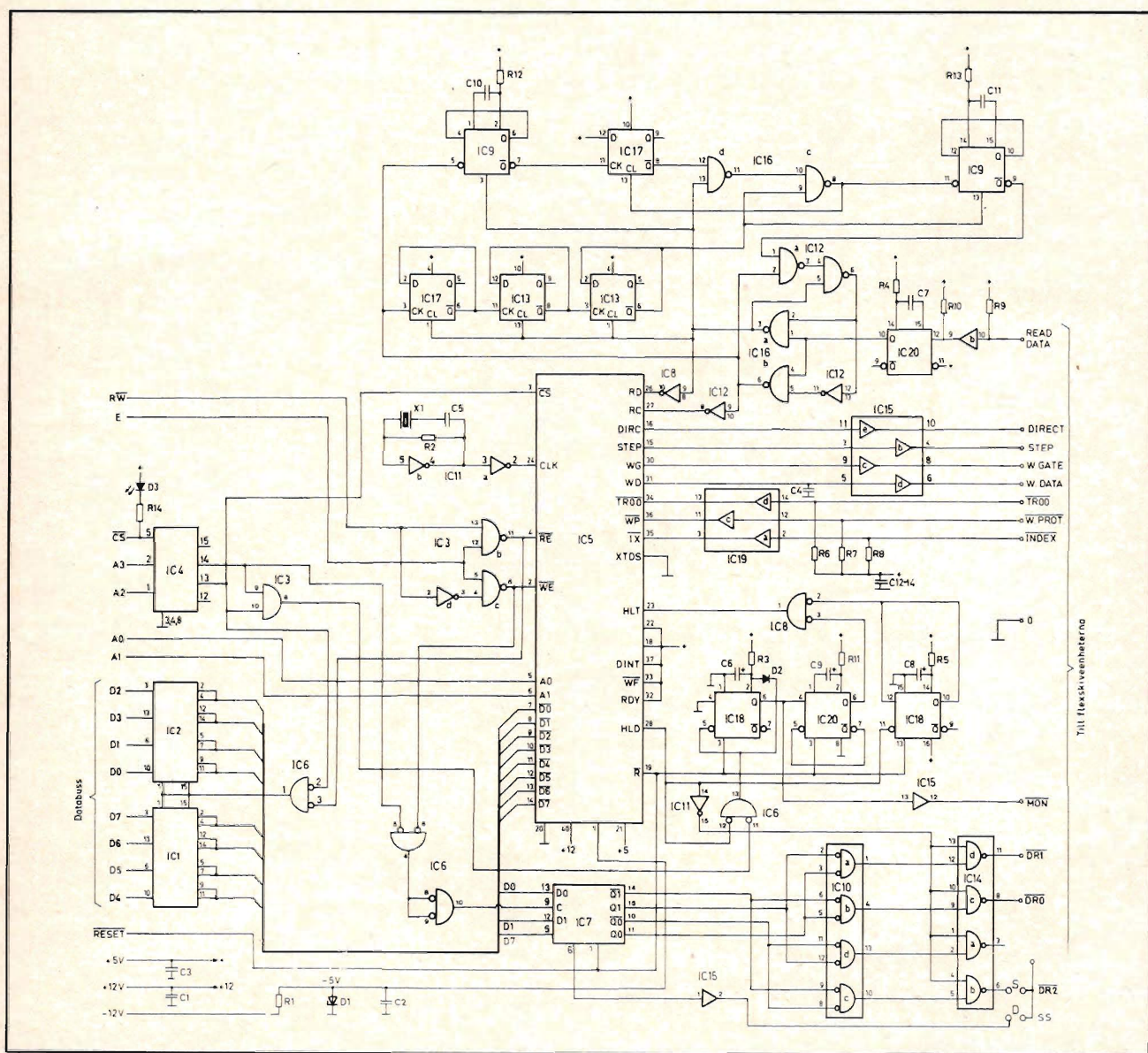


Fig 21:1. Principschema för anpassningskortet till miniflexenheter.

för datoranvändaren. All kommunikation med datorn sker med kommandon i klartext, det behövs inte längre några kryptiska maskinkodinknappningar.

Att systemet har kort åtkomsttid är något som är mycket värdefullt. Datorns eget programminne behöver bara kompletteras med en "boot-loader" för flexskivesystemet, totalt cirka 80 bytes. Allt annat finns lagrat på flexskivan och anropas vid behov.

Egentligen består kompletteringen för flexskiva av tre delar. Först behövs ett *anpassningskort* i datorn. Detta kort ansluts till flexskiveenheten, i vilken man slutligen stoppar i en flexskiva. För att det hela skall fungera måste det finnas minne i datorn på vissa adresser, se *tabell 21:1*. För att ge ett effektivt utnyttjande av den tillgängliga minnesarean har själva operativsystemet placerats på den höga adressdelen. På så sätt finns det upp till 32 k minne för användarprogrammet vid 6800-system och upp till 48 k minne vid 6809-system. Operativsystemet upptar 8 k i minne, och i det området finns plats för egna s k utility commands, dvs små program för olika typer av programhantering och hjälprutiner. Eftersom en flexskiva rymmer ca 80 k ord finns det mycket som kan rymmas.

Anpassningskort

Vi börjar emellertid med anpassningskortet. Det är ett litet krets-kort som skall placeras i kontakten J2 på moderkortet. Det kommer då att adresseras på adresserna \$8014–801B eller \$E014–E01B, beroende på vilken CPU som används. Principischemat återfinns i *fig 21:1*. IC4 kodar av adresserna så att IC5 nås på adresserna \$8018–801B (\$E018–E01B) och IC7 på adresserna \$8014–8017 (\$E014–E017). Varje gång kortet adresseras, triggas monovippan IC18 över IC3 stift 8 och IC6 stift 13. Utgången från IC18 går sedan via buffertkrets IC15 till flexskiveenheten och startar motorn i den. MON är en förkortning av motor on. IC7 är ett fyrabitars D-

Tabell 21:1	Mikroprocessor	
	6800/6808	6809
Minne för operativsystem	\$A000-BFFF	\$C000-DFFF
Erforderligt minne för att starta upp systemet	\$0000-2FFF	inget utöver ovannämnda
Erforderligt minne för att köra basic	\$0000-3FFF	\$0000-3FFF
Erforderligt minne för att köra den utökade versionen av basic	\$0000-7FFF	\$0000-7FFF
Maximalt användbart kontinuerligt minne	\$0000-7FFF	\$0000-BFFF

Tabellen visar hur mycket minne som behövs för olika tillämpningar.

register som väljer vilken enhet som data skall utväxlas med. Data-bitarna D0 och D1 representerar ett värde mellan 0 och 3 som direkt motsvarar numret på flexskiveenheten. Normalt används bara två enheter, men det finns möjlighet att ansluta upp till fyra stycken. D7 är till för de fall då dubbelsidiga drivenheter skall användas. Data-biten indikerar då vilken sida av flexskivan som data skall hämtas från.

För närvarande används inte denna signal, eftersom mycket få tillverkare kan leverera dubbelsidiga drivenheter i miniformat. Den finns dock med för att tillgodose framtida behov. IC10 och IC14 avkodar D0 och D1 samt driver styrningen av utgångssteg och magnet på varje flexskiveenhet. Längst ner till höger på schemat finns ett byglingsfält markerat D och S (S = single, D = double, SS = side select). Om maximalt två vanliga (= enkelsidiga) drivenheter skall anslutas behöver ingen av dem byglas.

IC5 är en LSI-krets som tar hand om merparten kommunikation mellan mikroprocessorn och flexskivan. IC5 innehåller register för de olika sektorerna, spåren och in/utdata. Att i detalj gå in på hur IC5 fungerar skall vi inte göra. Vi konstaterar bara, att den ansluts ungefär som vilken annan periferikrets som helst och att den fordrar vissa signaler från flexskiveenheten. IX

är en ingång för indexpulser, pulser som kommer från en fototransistor vilken känner av ett hål i flexskivan. För varje varv som flexskivan gör, kommer en puls från fototransistorn. Hålet kallas för indexhål och hjälper IC5 att hålla reda på var på flexskivan det kombinerade skriv/läshuvudet befinner sig. WP betyder write protected och talar om att den instoppade flexskivan är skrivskyddad. Försöker man spela in data på en skrivskyddad flexskiva skriver datorn ut ett felmeddelande. TR00 betyder track 00 och den signalen kommer från en mikrobrytare som känner av när skriv/läshuvudet står över spår nummer 00.

Till flexskiveenheten skickas signaler för att förflytta huvudet till olika spår (DIRC och STEP) samt skrivdata (WD) och skrivkommando (WG). Eftersom dessa ingångar är relativt lågohmiga krävs TTL-buffertarna IC15 för drivningen. Från flexskiveenheten kommer en datasignal vid avspelnning som innehåller en blandning av klockpulser och datainformation. För maximal säkerhet används en separat data/klockpulsseparator. Den består av samtliga kretsar mellan ingången READ DATA och de båda ingångarna RD och RC på IC5. RD är en ingång för data och RC är ingången för klockpulserna med vilka IC5 kan tyda informationen från flexskivan.

X1 är en kristall på 1 MHz som

lämnar klockpulser för inspelning av data. De tre monovipporna i IC18 och IC20 ger olika fördröjningstider för att kompensera för mekaniska trögheter hos flexskiveneheten samt för en viss gångtid hos motorn efter varje access till skivan. R3/C6 gör att motorn går cirka 45 sekunder efter senaste adressering av kortet.

Inga trimningar

Kretskortet med komponentplacering återges i fig 21:2. När alla komponenter är på plats kan kretsarna sättas in. Var extra aktsam om IC5, ty den är av MOS-typ och dessutom inte särskilt billig. Nu är det bara att koppla ihop kortet med en flexskivenehet. Det sker bäst med en flatkabel med 34-ledare. Har man tänkt att någon gång i framtiden skaffa en extra flexskivenehet kan det vara klokt att redan från början skaffa en kabel för två flexskiveneheter. Matningsspänningarna till flexskiveneheten kan också tas från datorn. En enhet drar ca 500 mA från +5 V och max 1A från +12 V.

Hur kontakten för matningsspänningarna skall kopplas framgår av kapitlet om datorlådans montering.

Eftersom inga trimpunkter finns är det nu bara att köra igång systemet. Det fordras först och främst att dataminnen finns installerat på rätta adresser. Det fordras också en flexskiva med operativsystem FLEX 2.0 eller FLEX 9.0 innan det hela fungerar. Om dessa båda villkor är uppfyllda, är det bara att stoppa i flexskivan, helst skrivskyddad med en liten bit tejp över slitsen i ena hörnet.

Om monitorprogram TBUG2 eller CBUG2 är igång, startas det hela genom att man trycker på F-tangenten. Flexskiveneheten skall då starta och lysdioden tändas. Efter några sekunder skall terminalen skriva ut:

FLEX 2.0

DATE (MM, DD, YY)?-

Operativsystemet har identifierat sig och frågar efter dagens datum. Detta gör den för att man skall kunna datummarka alla nya

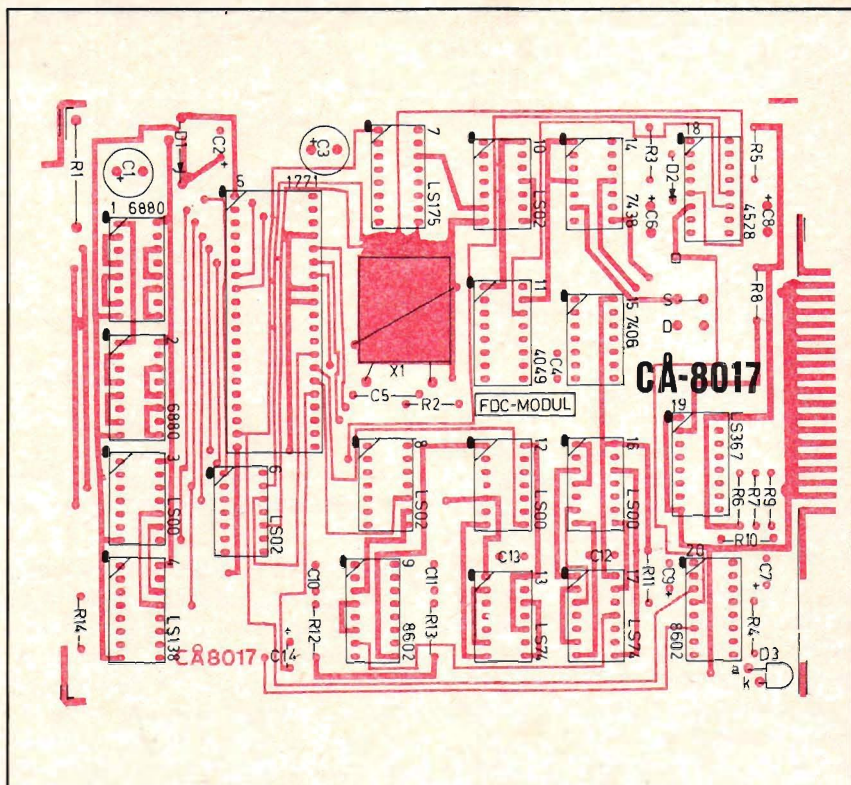


Fig 21:2. Kretskortet med komponentplaceringen tryckt i svart.

CATALOG OF DRIVE NUMBER 0
DISK. FLEX? #1

NAME	TYPE	SIZE	PRT
ERRORS	.SYS	9	
FLEX	.SYS	25	
PRINT	.SYS	1	
CAT	.CMD	3	
COPY	.CMD	5	
DELETE	.CMD	2	
P	.CMD	1	
LIST	.CMD	3	
ASN	.CMD	1	
RENAME	.CMD	1	
APPEND	.CMD	3	
BUILD	.CMD	1	
EXEC	.CMD	1	
NEWDISK	.CMD	6	
SAVE	.CMD	2	
TTYSET	.CMD	2	
O	.CMD	2	
LINK	.CMD	1	
JUMP	.CMD	1	
DATE	.CMD	2	
PRINT	.CMD	2	
PROT	.CMD	1	
QCHECK	.CMD	4	
VERIFY	.CMD	1	
VERSION	.CMD	1	
XOUT	.CMD	2	
I	.CMD	1	
EDIT	.CMD	28	
ASMB	.CMD	47	
SAVE	.LOW	2	
FLEX	.COR	22	

SECTORS LEFT = 157

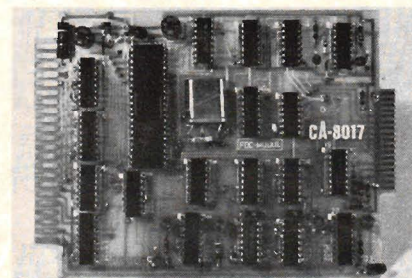


Fig 21:3. Bilden visar det monterade kontrollkortet.

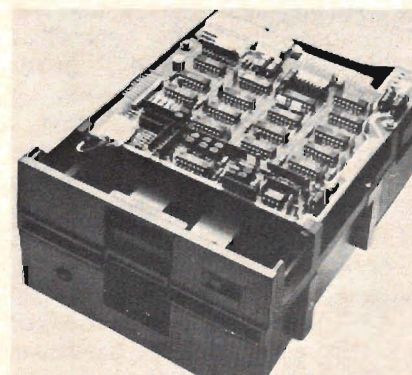
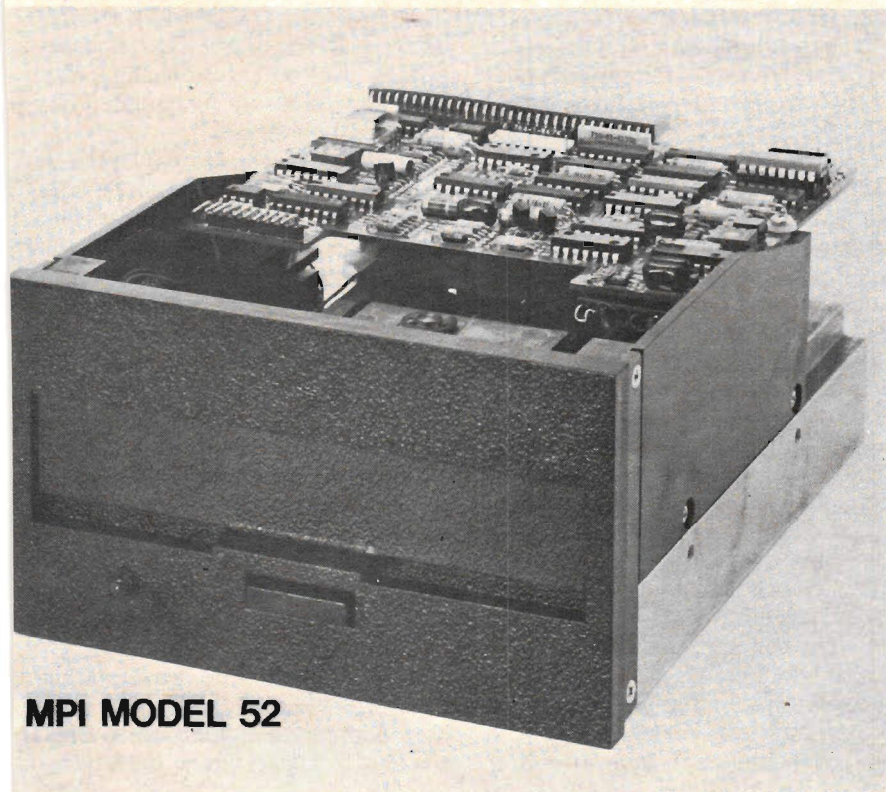


Fig 21:4. Bilden visar en miniflexskivenehet av fabrikat TEAC.



MPI MODEL 52

Fig 21:5. Bilden visar en flexskiveenhet av fabrikat MPI.

program som vi kanske kommer att skapa under denna körning. Efter att ha svarat med exempelvis 1,18,80 och retur-tangenten, skriver terminalen ut:

+++--

Detta är en indikering på att flexsystemet är klart att börja arbeta åt oss. Om vi vill veta vad som finns på flexskivan kan vi skriva CAT. Vi får då en katalog över innehållet på flexskivan med information om namnet på filen (en file är benämningen på ett program eller ett avsnitt på flexskivan), vilken typ av file det är, hur många sektorer den upptar samt om den är skyddad på något sätt.

I tabell 21:2 återges vad som finns på en flexskiva, vilken används som sk systemskiva. Den innehåller alla nödvändiga program och kommandon som fordras för editering, assemblering, programtestning, basicprogrammering m.m. Det egna programmet, som kan vara skrivet i basic eller vara källkod för assembler, finns lagrat på en flexskiva som placeras i den

andra flexskiveenheten (om två enheter används). Av allt som är uppräknat i tabell 21:2 är det bara FLEX2.SYS som alltid ligger i minnet. De övriga av typ CMD är användarkommandon vilka plockas ner vartefter det behövs. Om vi exempelvis har en skrivare ansluten till utgång B på ACIA-kortet och vill ha en CAtalog utskriven på den, ger vi operativsystemet kommandot P,CAT. Utskriften kommer då att temporärt styras till skrivaren. P-kommandot hämtar en file kallad PRINT.SYS vilken innehåller en utrutin för våra ACIA. Därefter utför CAT en uppräkning av innehållet på flexskivan och sänder ut denna "katalog" till vår skrivare. När det är klart, återgår programmet till FLEX och de tre plustecknen skrivs ut på nytt på vår terminal.

Den dokumentation som medföljer operativsystemet är mycket utförlig och varje kommando är tydligt förklarat. Det finns även beskrivet hur man kan skriva egna kommandon till operativsystemet.■

Komponentförteckning:

C1,3	47 μ F 25 V el.lyt
C2,8,14	10 μ F el 6,8 μ F tantal
C4	100 pF keram.
C5	47 pF styrol
C6	22 μ F tantal
C7	100 pF styrol
C9	47 μ F tantal
C10,11	2200 pF styrol
C12,13	0,1 μ F pol.
D1	5,1 V 1 W zenerdiod
D2	1N4148
D3	röd lysdiod
IC1,2	MC6880 el motsv
IC3,12,16	74LS00
IC4	74LS138
IC5	FD 1771
IC6,8,10	74LS02
IC7	74LS175
IC9,20	DM 8602
IC11	4049
IC13,17	74LS74
IC14	7438
IC15	7406
IC18	4528
IC19	74LS367
R1	91 ohm 1 W
R2	4,7 M 5 % 1/8 W
R3	1 M
R4	10 k
R5	22 k
R6-9	150 ohm
R10	2,2 k
R11	47 k
R12	7,5 k
R13	9,1 k
R14	220 ohm
X1	1 MHz kristall
1	kretskort CÅ-8017
10	14-pin IC-hållare
9	16-pin IC-hållare
1	40-pin IC-hållare

Dessutom fordras:

1 el 2	Flexskiveenheter
1	kablage (flatkabel med kontakter)
1	operativsystem FLEX 2,0 eller FLEX 9.0

Skrivare för RT-datorn

AV ÅKE HOLM
OCH GUNNAR LILLIESKÖLD

Förr eller senare får man behov av att få utskrift på papper av sina program och datalistor. Vi beskriver här några typer av skrivare som kan anslutas till RT-datorn.

■ Ganska snart kommer man fram till att en skrivare borde införlivas i datasystemet. Man kan då få ut dataprogram listade på papper för att kunna sparas eller kanske textmaskin i ett ordbehandlingssystem. Det finns en rad typer av skrivare med olika egenskaper för skilda användningsområden och i varierande prisklasser.

Ett problem för datoramatörer är att skrivaren fortfarande utgör en avsevärd del av den totala kostnaden för ett datasystem. Priserna är dock på väg ned och det kommer allt kapablare enheter.

När man väljer skrivare måste

man först försöka komma fram till vad skrivaren skall användas till och hur den skall användas. Är det viktigt att den skriver fort eller att den skriver väl? Man får då välja mellan två huvudgrupper: matris-skrivare eller s k skönskrivare. De förra arbetar med ett typhuvud som med nålar trycker punkter i en matris, vanligen 5×7 eller 7×9 punkter. Punkternas mönster bildar det önskade tecknet. I skönskrivare trycks ett typhuvud mot ett karbonband som i sin tur trycker mot papperet, dvs på samma sätt som i skrivmaskiner. Skönskrivare brukar antingen arbeta med kula eller med typhjul. Prismässigt ligger de mellan 10 000 och 20 000 kr. Matris-skrivarna är betydligt billigare och idag finns det ett flertal skrivare att välja på i prisläget 5 000 kr och uppåt.

Det finns ett sätt att få en billig skönskrivare, nämligen att bygga om en IBM skrivmaskin med kula och dragmagneter. Har man stor tur kan man hitta en sådan begagnad. En sådan kan relativt enkelt kompletteras för anslutning till RT-datorn.

Skriften blir lika bra som den från en skrivmaskin, men hastigheten blir låg. Ändå kommer den att

arbeta på ett sätt som ingen människa klarar: Kontinuerligt på gränsen till dess yttersta förmåga. Det innebär i sin tur att livslängden blir tämligen begränsad.

Matris-skrivarna skriver inte särskilt vackert men det finns de som är ganska tillförlitliga. Tillförlitligheten är A och O i professionella sammanhang, men en datoramatör får naturligtvis väga pris mot kvalitet. Ett exempel på utskrift från en matris-skrivare finner vi i avsnittet "assemblerprogrammering".

Bland de detaljer man skall se på vid val av skrivare är hur lättskötta de är. Kan man byta karbonband utan att bli kolsvart och måste man ha en bruksanvisning till hands fler gånger än den första? Vad arbetar den med för papper? Klarar den vanligt vitt papper eller måste den ha värmekänsligt eller elektrokänsligt papper? I de senare fallen kanske skrivarens pris blir lågt, men papperskostnaden desto högre. Hur många teckens bredd klarar skrivaren, hur många tecken per sekund eller hur många linjer?

Matris-skrivarens text — hur skall den se ut?

Vid val av skrivare bör man grans-

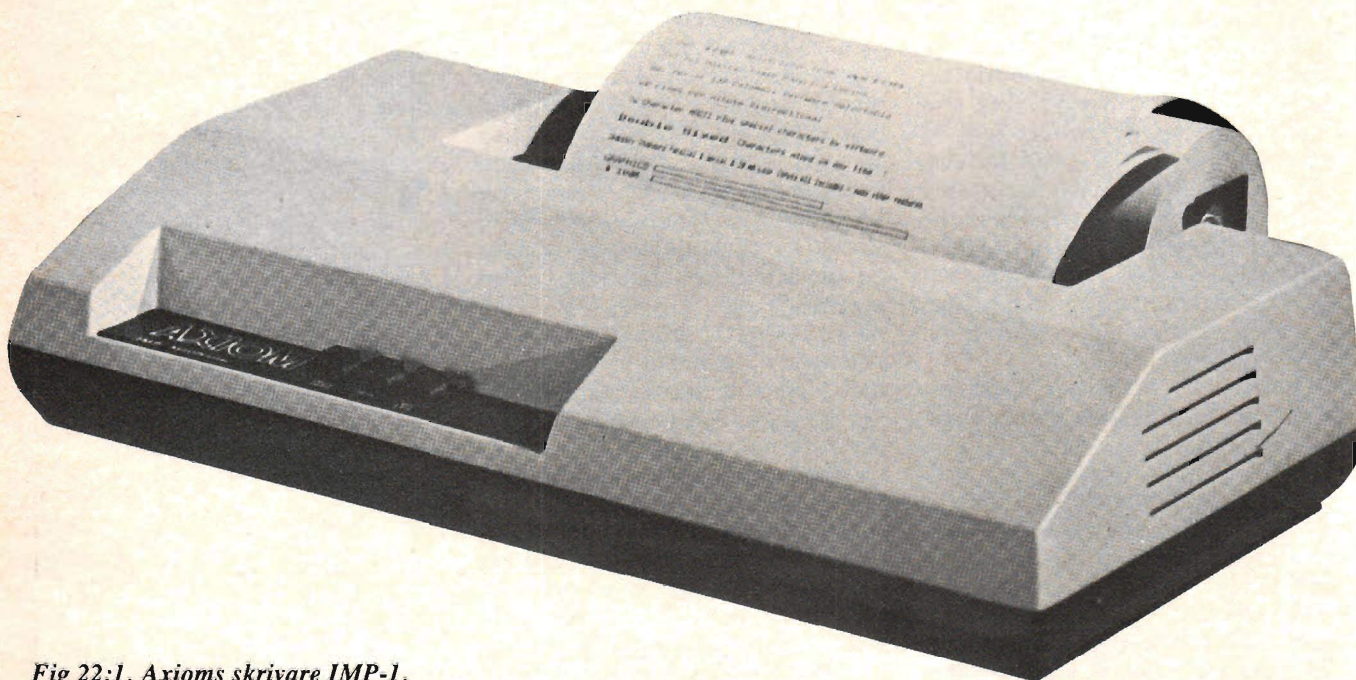


Fig 22:1. Axioms skrivare IMP-1.

ka utskriften. Den visar vad skrivaren är kapabel till. Låt den skriva på linne-papper av god kvalitet. Då kan man se om svärtan räcker till i alla sammanhang. Gör den inte det blir trycket grått i stället för svart. Prova med samma skrift och samma papper på flera skrivare. Granska tecknen med ett förstöringsglas. Då kan man se om svärtan varierar utmed tecknen eller om de har fula hörn.

Skriv ut en rad och skriv så ut samma text på samma rad dvs utan radframkallning. Texten skall då hamna på den gamla. Om inte får man ett reliefmönster. Jämför stora bokstäver med små tecken som punkt och komma. En högkvalitativ skrivare justerar automatiskt svärtan så att den blir lika i de olika fallen.

Priset är naturligtvis en viktig faktor, kanske den allra viktigaste för en datoramatör. Fråga gärna andra datorförare om råd. Skrivare innebär mekanik som i sin tur innebär slitage och service. Välj gärna en mekaniskt enkel men robust konstruktion.

Billigare och smartare

Skrivarna utvecklas f n kraftigt och man kan vänta både billigare och intelligentare enheter. De senare har ofta en inbyggd mikroprocessor som gör att skrivaren kan anslutas mera oberoende till vilken dator som helst. Vanligast är att antingen serieöverföring tillämpas enligt RS 232-standard eller har skrivaren s k Centronics-interface som arbetar med 7 bitar parallellt. Mikroprocessorn gör också att man lättare kan välja typsnitt.

En skrivartyp som kommer är den skönskrivande matrissskrivaren. Den arbetar med nålar, som i dagens skrivare, men raden skrivs om 2, 3 eller kanske 4 gånger då punkterna har förskjutits så att tecknen blir jämna. Speciella skrivarhuvud kommer också som ger jämna tecken.

En liten översikt

Vi skall titta närmare på några olika typer av matrissskrivare, som finns på marknaden. Utöver dessa

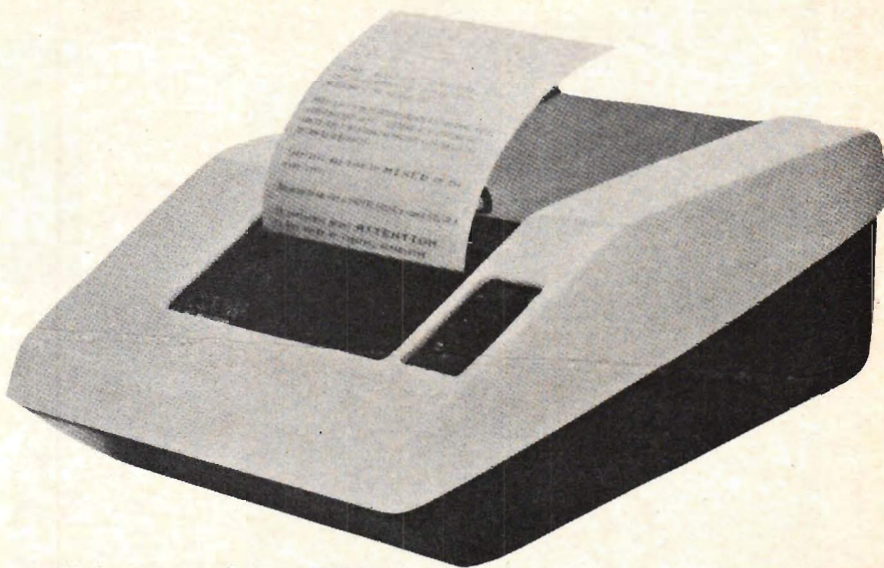


Fig 22:2. Axioms skrivare EX-801.

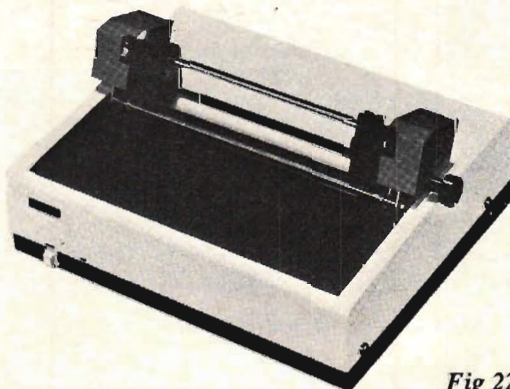


Fig 22:3. Epsons skrivare TX-80.

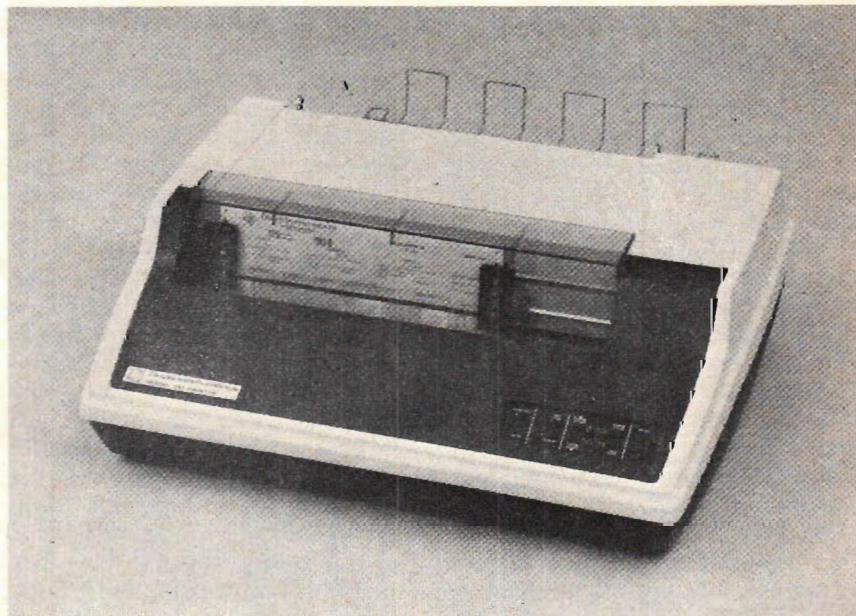


Fig 22:4. Texas skrivare 810.

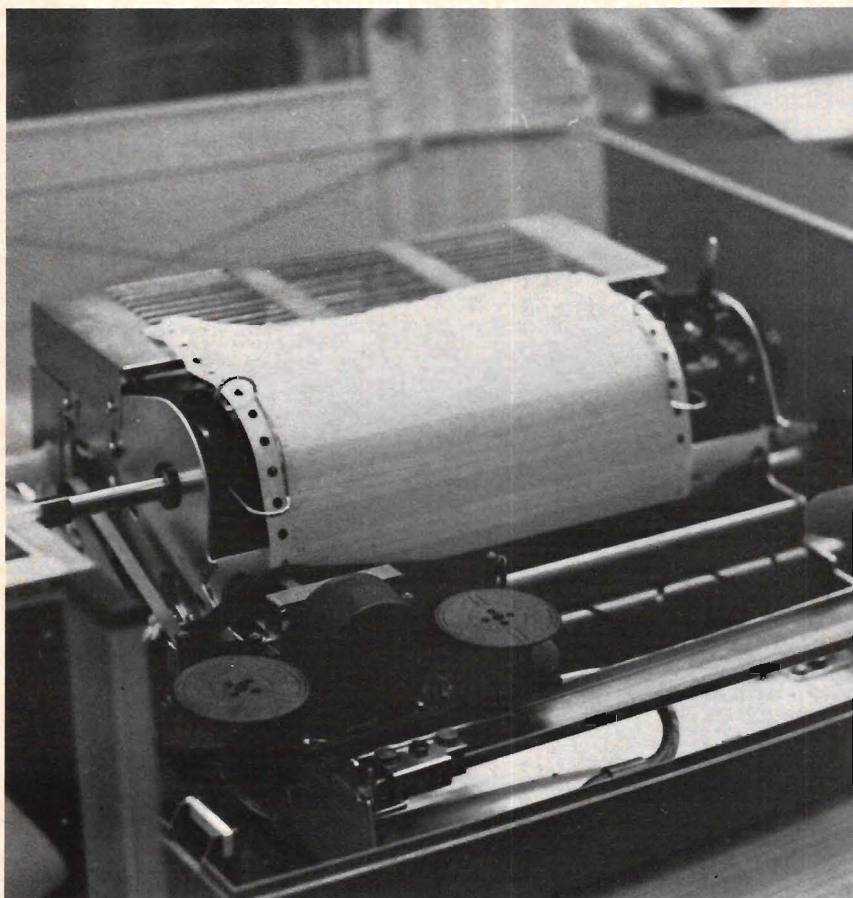


Fig 22:5. Itoh är en japansk skrivare i lågprisklass, här visad utan kåpa.

finns ytterligare ett antal, men eftersom det hela tiden kommer nya varianter och nya modeller nöjer vi oss med dessa.

Axiom tillverkar en liten skrivare med beteckningen IMP, se fig 22:1. Den skriver med 7×7 matris på papper med 80 eller 96 tecken per rad. Såväl små som stora bokstäver kan återges liksom Å, Ä och Ö. Skrivhastighet 50 rader per minut. Skrivhastigheten påverkas ej av antalet tecken per rad, eftersom skrivhuvudet matas över hela raden även om bara ett tecken skall skrivas. Anslutes till datorn med RS-232-C snitt, 50–1200 baud. IMP-skrivaren kostar (1980) ca 5 000 kr exklusive moms och kan även återge grafik.

EX-801 heter en annan skrivare från **Axiom**. EX-801 har 5×8 matris och skriver upp till 160 tecken per sekund. Möjlighet finns för komprimerad, expanderad, liksom

inverterad skrift. Svensk representant: **Saven AB**, Vaxholm.

Epson TX-80 är en skrivare, som har 5×7 matris och skriver 80 tecken per rad eller 40 tecken med dubbelbredd per rad. Skrivhastighet 70 rader per minut. Små och stora bokstäver samt möjlighet att använda egen teckengenerator för specialtecken. Svensk representant: **Scantele**, Stockholm.

Texas 810 heter en skrivare med 5×7 matris som skriver 150 tecken per sekund. Den har små och stora bokstäver och maximalt 132 tecken per rad på 15 tums pappersbredd. Skrivaren finns i olika utföranden för komprimerad text, för automatisk pappersmatning vid olika längder på papperet m m. Papperet är av typ löpande bana. 810-skrivaren kostar dock ca 16.000 kr exkl moms. Svensk representant: **Texas Instruments**, Stockholm.

Itoh 8300 är en japansk matris-skrivare (7×5 punkter) som presterar 125 tecken/sekund med 10 tecken per tum. Den har små och stora bokstäver och papperet matas fram med piggmattning. Upp till 9" bredd tillåts. Svensk representant: **Scandia Metric AB**, Solna.

Anadex DP800 klarar 112 tecken/sek och skriver med 80 teckens bredd på traktormatat papper. Den finns att få för alla system som har serie- eller parallellsnitt. Svensk representant: **Johan Lagerkrantz kb**, Upplands Väsby.

Heathkit H14 som är i byggsatform och dess färdigbyggda variant **HW14** är ett verkligt lågprisalternativ. Man kan välja mellan 80, 96 och 132 teckens bredd och maximal tryckhastighet är 135 tecken per sekund. Den har seriesnitt enl RS 232C eller 20 mA strömslinga och kan ställas in på överföringshastigheter mellan 110 till 4800 baud. Papperet kan vara från 2,5 till 9,5 tum brett. Svensk representant: **Heathkit Scandinavia AB**, Stockholm.

Emako 22 är en japansk matris-skrivare som bl a har å, ä och ö samt fyra olika stilar. Program-mässigt kan man även få tyska och engelska specialtecken enl ASCII-kod samt t o m Katakana. Svensk representant: **Swedish Electronics hb**, Uppsala.

OKI är en annan japan med ett brett sortiment skrivare. **Okidata Microline 80** heter den billigaste matris-skrivaren som presterar 80 tecken per sekund på upp till 132 teckens bredd. Papperet är friktions- eller traktormatat.

Facit 4520/4521 är svensktillverkade matris-skrivare i lågprisklass. 4520 arbetar med friktionsmatning och 4521 med pinnmatning. För övrigt är de identiska. De presterar 60 rader per minut med 80 nedslags bredd eller 100 tecken per sekund och skall klara kontinuerlig drift. Som standard finns både serie- (V24/RS 232C) och parallellsnitt (Centronics). Den trycker små och stora bokstäver och har totalt 95 ASCII-tecken. ■

Basic-dator på ett kort

AV BENGT GRAHN

En komplett basic-dator på ett kort! Det kräver dock en yttre terminal för strömslinga (20 mA), så som teletype.

Språket heter NIBL och är en slags liten basic speciellt framtagen för styrsystem.

Som processor används SC/MP från National Semiconductor som även tagit fram NIBL.

■ Vi skall här presentera en komplett dator på ett kort som bara behöver anslutas till en yttre terminal. Datorn arbetar med ett basic-

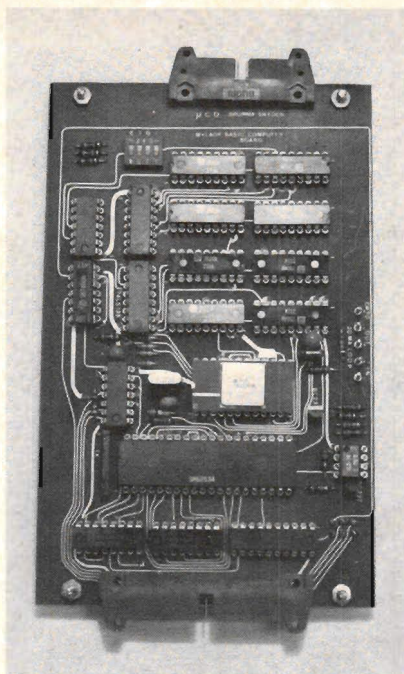


Fig 23:1. Hela basic-datorn rymms på ett Europakort.

liknande språk som kallas NIBL framtaget av National Semiconductor till deras mikroprocessor SC/MP. Datorn ingår i ett system som döpts till *Mylady* och består av ett flertal olika kort, alla i Europaformat, främst avsedda för industriella tillämpningar, men som med fördel, inte minst prismässigt, även kan användas av privatpersoner, skolor, kontor, smärre företag m m.

Här skall jag endast behandla basic-datorn, en av komponenterna i *Mylady*. Hela basic-datorn ligger på ett kretskort som bara kräver strömförsörjning och en teletype-terminal eller annan kommunikationsenhet som arbetar med 20 mA strömslinga enl V24-standard. Den matas med 5V enbart och består av ett litet antal standardkomponenter, vilket gör den lämplig som byggobjekt för den som kan handskas med lödpennan. Men även om den är enkelt uppbyggd,

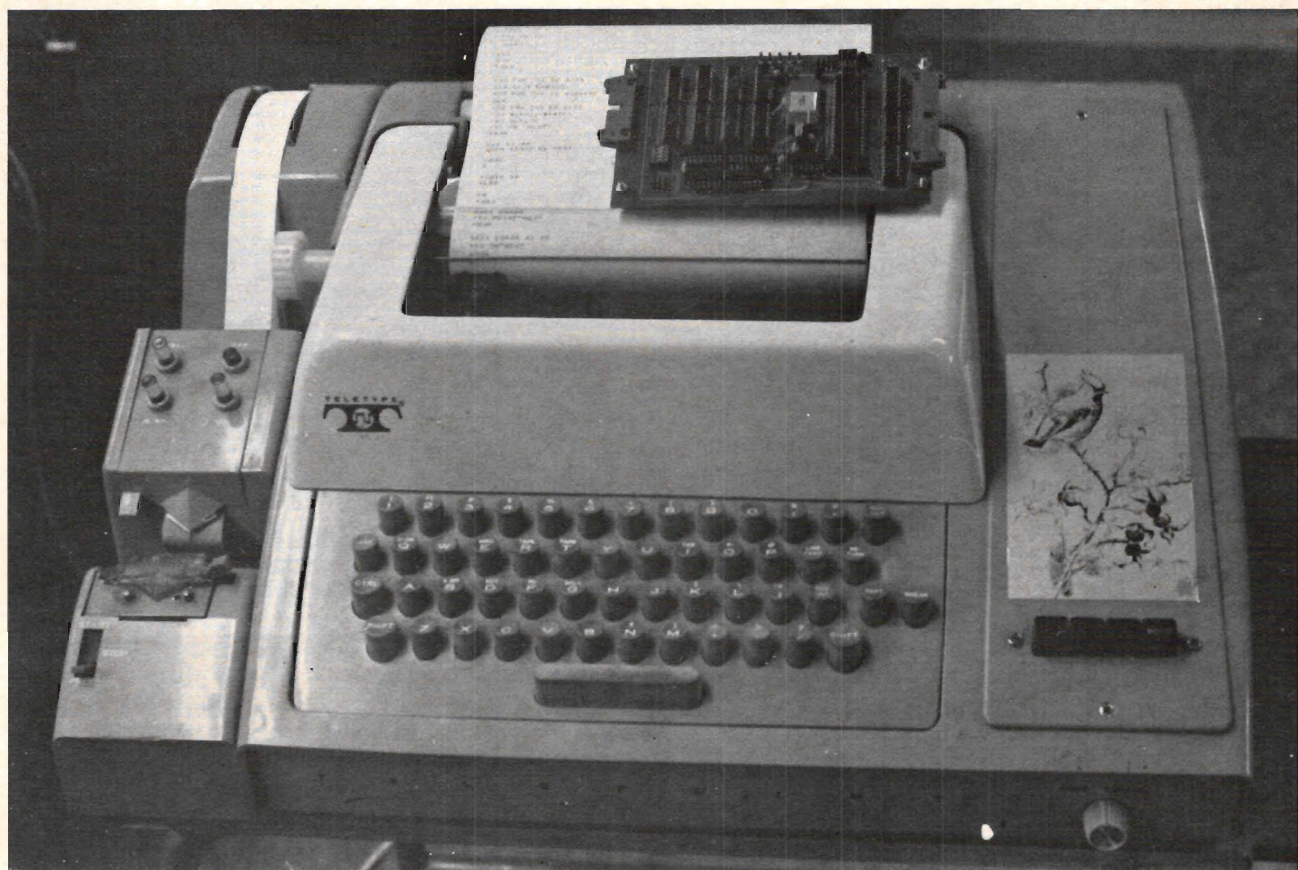


Fig 23:2. Datorkortet kräver en yttre terminal. Här ser vi det tillsammans med en Teletype-terminal.

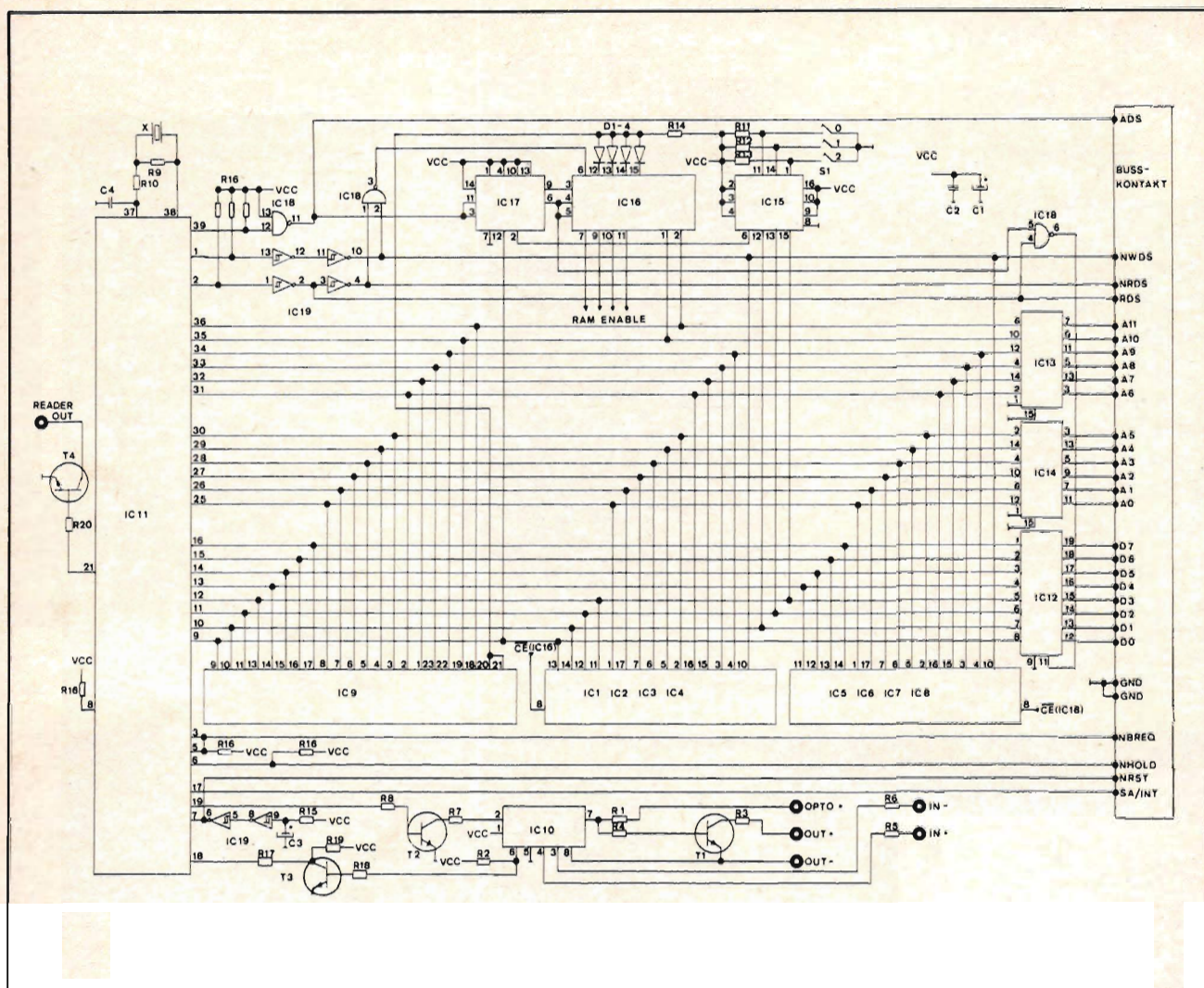


Fig 23:3. Principalschema för basic-datorn.

är den avsedd att ingå i Mylady-systemet och är därför utbyggbar med avseende på minnesutrymme etc. Kortets adressavkodare är så konstruerad att minnesarean om totalt 8 kbytes kan läggas var som helst inom maskinens 65 kbytes adressområde. Det innebär, att man om man så önskar, kan använda kortet som processorkort på vanligt sätt för att köra andra program än den basic-interpretator som finns på kortet. Då kan även datorns interna 4 kbytes RAM utnyttjas på vanligt sätt.

Moderkort ersätts av flatkabel

Den mekaniska uppbyggnaden är ett kretskort i europaformat,

160×100 mm, men systemet är så uppbyggt att man helt slipper ifrån de ofta kostnadskrävande europarackkonstruktionerna. I stället staplar man de olika korten i systemet ovanpå varandra med distanser och drar en flatkabel som ersättning för de konventionella moderkortskonstruktionerna.

Ännu en besparing, alltså! Det innebär att man är helt fri beträffande val av låda och andra mekaniska arrangemang. Datorn och även ett helt Mylady-system kan passas in i praktiskt taget vad som helst. Elektriskt sett är datorn ganska konventionellt uppbyggd. Den består grovt av processor, minne, adressavkodare, 20 mA interface och drivare till och från busskon-

takten, dit de andra enheterna kopplas (externa minnen, in/ut-enheter o dyl) om man vill ha sådana.

Originell processor med flera finesser

Processorn är, liksom de i de flesta MyCD-konstruktioner, av typ SC/MP eller som den även betecknas **INS 8060** från **National Semiconductor**. I den här applikationen är processorn kristallstyrd med en kristall på 4 MHz. Processorn sköter även mjukvarumässigt seriekommunikationen in och ut till yttvärlden, en facilitet som SC/MP är ganska ensam om bland de konventionella 8-bitsmaskinerna. Därvid används *Flag O* som utor-

En sak förtjänas dock att nämnas: Den som lusläser det stora kopplingsschemat kommer snart underfund med att beteckningarna på bussledningarna mellan processorn och RAM inte stämmer, vare sig på data eller adressbussen. Det har sin förklaring. Skillnaden mellan RAM och ROM/PROM är ju bland annat att man i RAM inte är bunden till i vilken ordning man lägger in sin information, förutsatt att man sedan läser den i samma ordning. Informationens rent fysiska placering i kapseln saknar betydelse. Då det gäller ROM/PROM är förhållandet i princip det samma, men dessa kretsar avses kunna fungera även i andra applikationer. Därför är det tillrådligt att göra ledningsdragningen enligt märkningen för respektive krets (och då det gäller ROM är det helt nödvändigt), medan man kan ta sig lite större friheter med RAM.

Ett RAM skall inte läsas av någon annan maskin än den det sitter i. Friheterna får dock inte sträckas för långt, databussen måste kopplas till datain/utgångarna, adressbussen måste kopplas till adressingångarna, $\overline{CE}$ och R/W måste kopplas rätt. Ordningsföljden hos ledarna på respektive buss är däremot fri. Detta förhållande har utnyttjats vid utformning på det här kortet vid övergången från ROM-kapseln till RAM-arean, eftersom

pinkonfigurationen stämmer dåligt mellan de två kretstyperna. En detalj alltså som helt saknar praktisk betydelse.

Ena halvan av en byte (de 4 mest signifikanta bitarna) tas om hand av IC5-IC8, medan andra halvan är kopplad till IC1-IC4. I det stora kopplingsschemat är kretsarna av tydlighetsskäl sammanförda till två stora block, ett för respektive bytehalva. Det är dock att observera att $\overline{CE}$ (stift 8) på IC1 är inkopplad med $\overline{CE}$ på IC5, IC2 med IC6 osv. Detaljerna ur denna aspekt framgår av fig 23:5.

Bussledningarna är buffrade

Bufferkretsarna är avsedda att isolera datorn från den yttre bussen, samt att strömförstärka signalerna från processorn. 8097 är en vanlig s k tri-state buffer med sex förstärkare i. Var och en av dessa tar hand om sin adressledning från processorn. Ledningarna är sedan kopplade till busskontakten, till vilken i sin tur eventuella kompletteringsmoduler är kopplade. Det är ju så, att i ett s k general purpose-system som detta, vet man aldrig på förhand hur stor belastning som kommer att kopplas till bussledningarna, och därför är det lika bra att vara på den säkra sidan. Om relativt hög belastning skall kopplas in, kan 8097 direkt bytas mot den något kraftigare buffern 8797.

Kretsarna har tri-state kapacitet, men de är alltid tillslagna (stift 1 och 15 jordade). Det innebär, att även om datorn inte avser kommunicera med någon yttre enhet, finns ändå alltid signaler på dessa ledningar.

I system där ingenting skall kopplas till bussarna, kan de två 8097 helt utelämnas, då de inte fyller någon intern funktion. Samma sak gäller motsvarande bufferkrets på databussen, 8304. Den är dubbelriktad, dvs informationen kan flyta i båda riktningarna beroende på status på dess tillredningar (stift 11). Även den har tri-state-kapacitet men är alltid tillslagen genom att stift 9 är jordat. Drivkapaciteten mot den externa bussen är hög, 48 mA enligt databladet. Den fyller inte heller någon intern funktion och kan alltså i vissa fall utelämnas. Kretsen har åtta in/utgångar i vardera riktning och klarar ensam av hela databussen. 8304 är vidare kopplad så, att i samtliga fall, utom då processorn avser läsa någon yttre enhet, strömmar data i riktning ut mot externbussen. Det hänger samman med det hårdvarumässiga adresseringsättet hos SC/MP.

Fyra adressledningar på databussen

Det finns en signal, NADS (inverterad och buffrad i en grind i 7400,

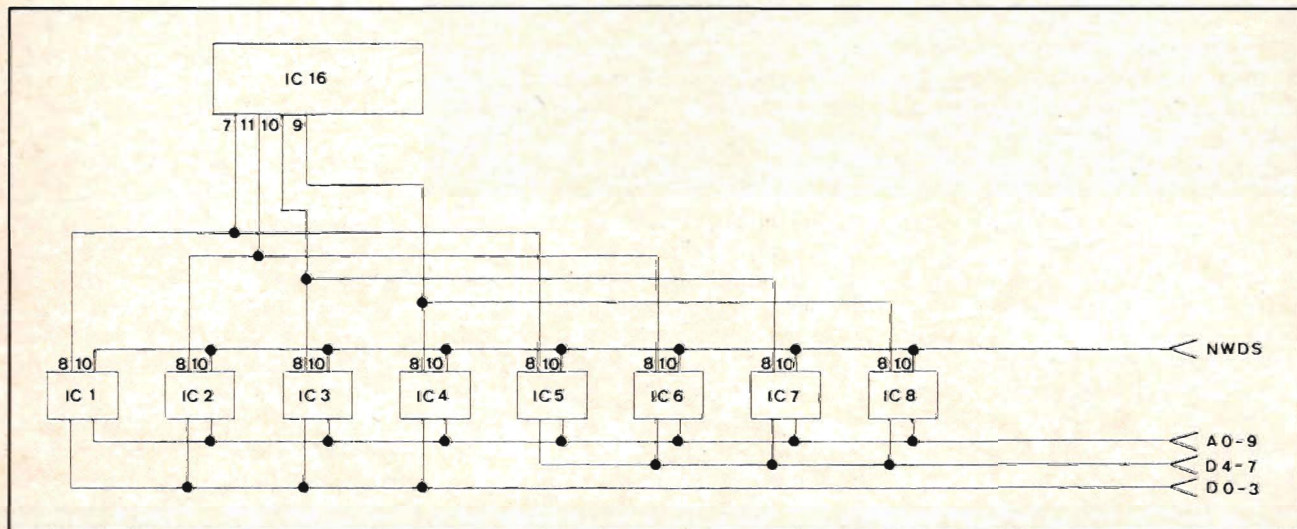


Fig 23:5. Avkodarens anslutning till minneskretsarna.

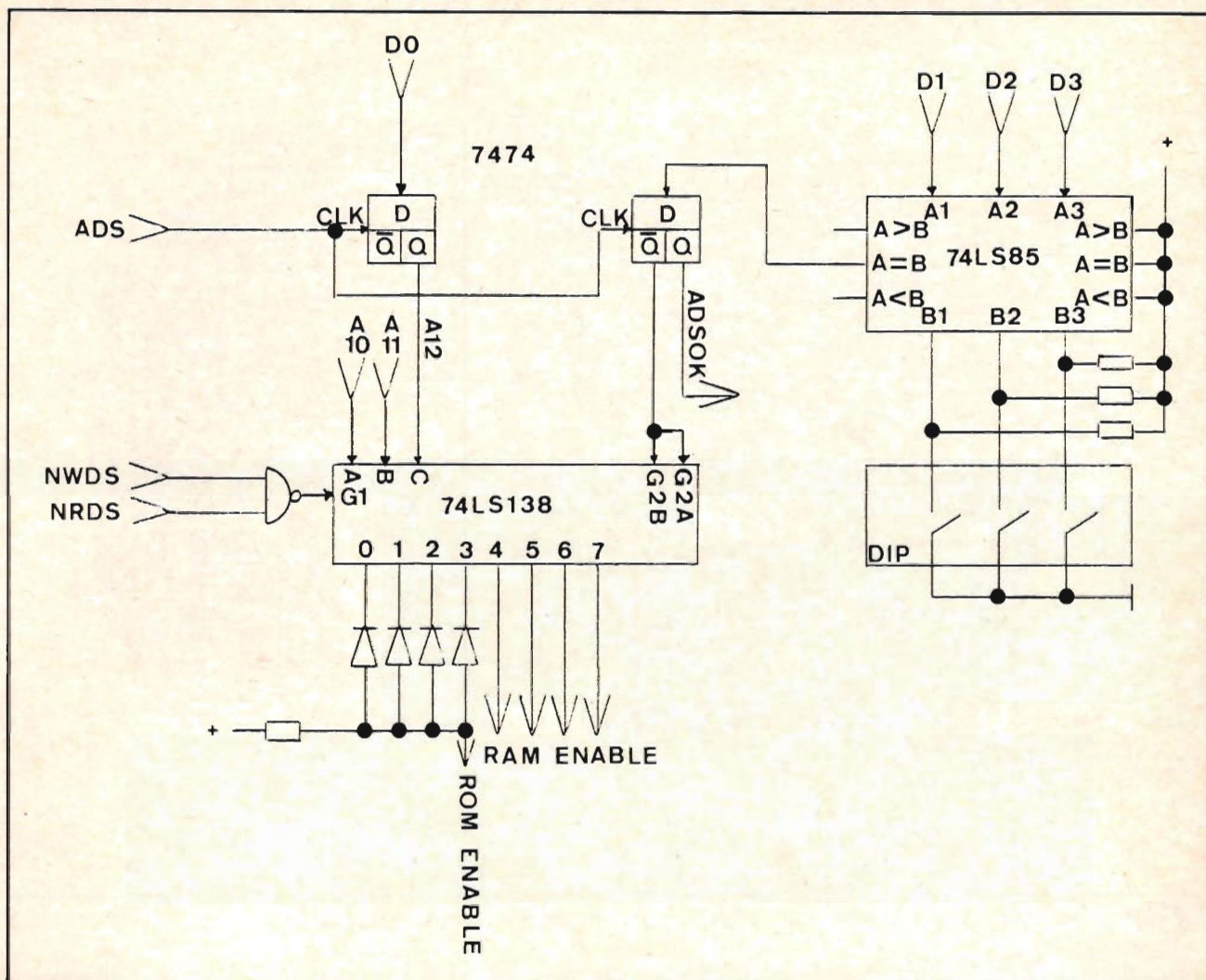


Fig 23:6. Avkodarkretsarnas uppbyggnad i detalj.

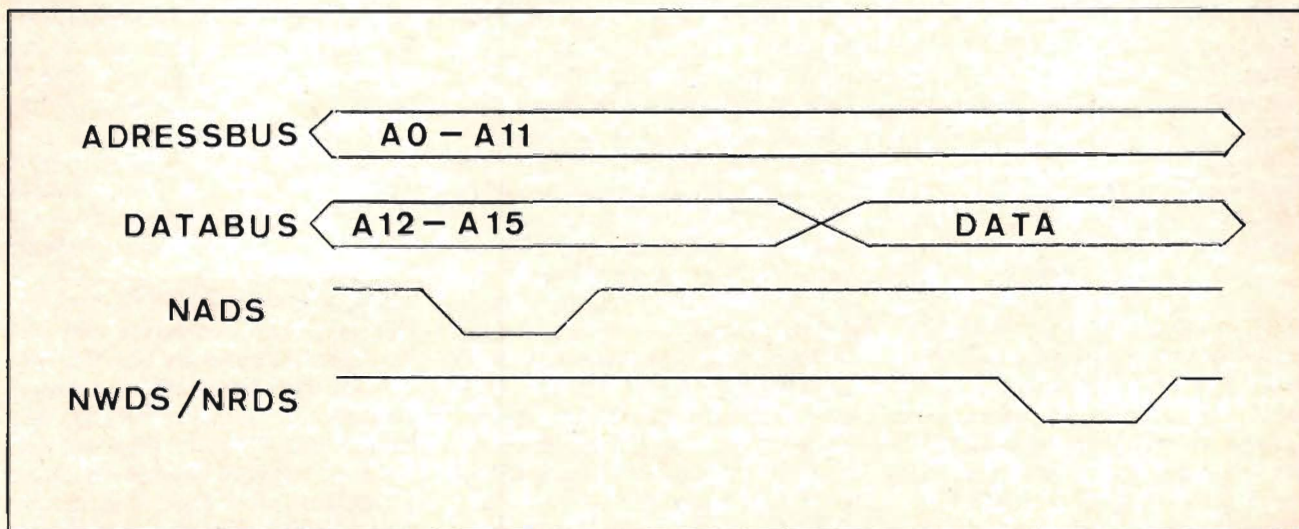


Fig 23:7. Tiddiagram som visar avkodningen i en SC/MP-dator.

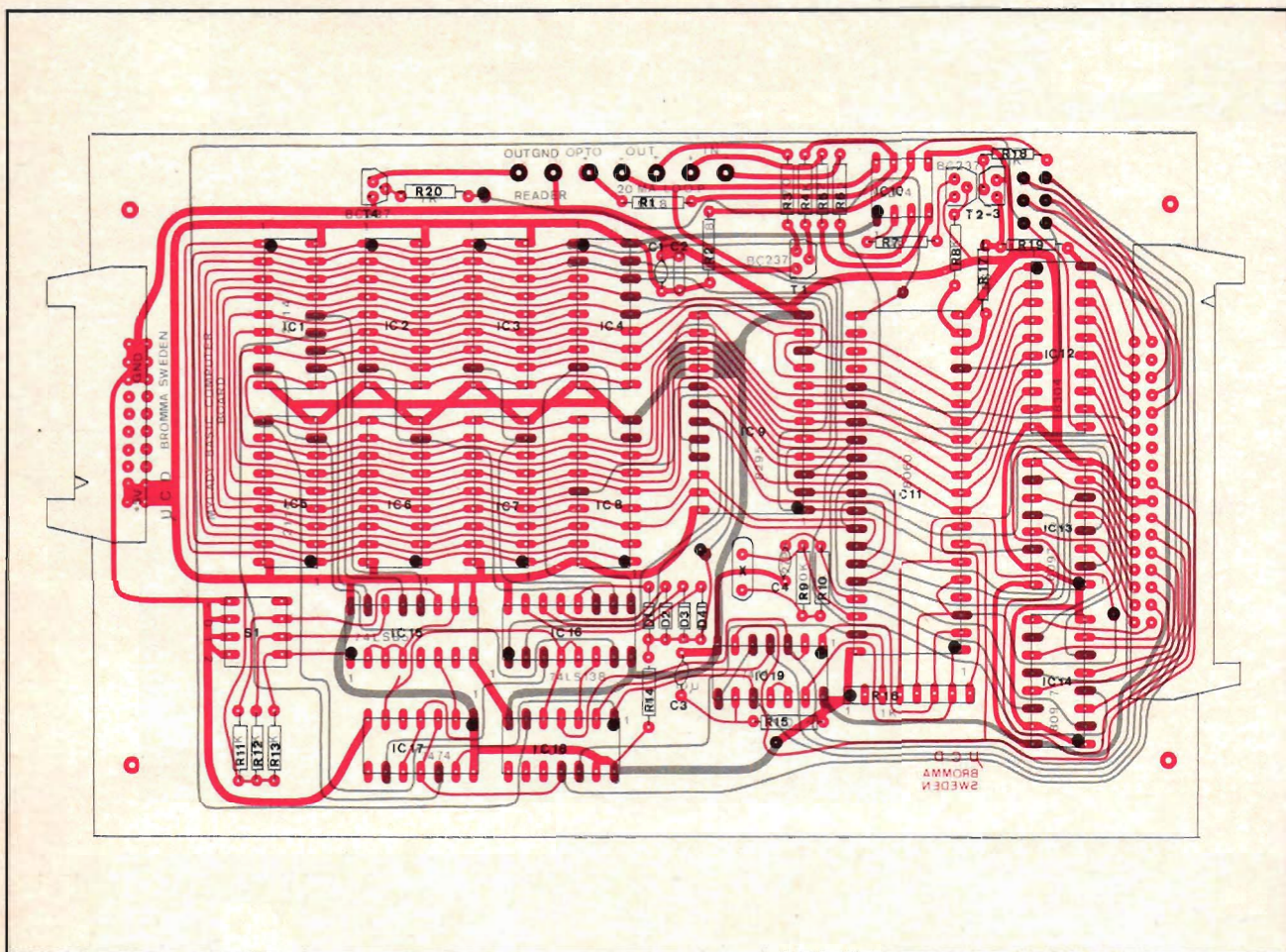


Fig 23:8. Komponenternas placering på kretskortet. Kretskortsmönstren visas i appendix.

IC18), som anger för yttre enheters adressavkodare, att adressbussens fyra mest signifikanta bitar (A12–A15) finns att hämta på databussen i ett visst ögonblick. (Metoden innebär, att endast tolv adressledningar behövs för adresseringen, men å andra sidan måste även databussen avkodas under tiden NADS är aktiv.) Vidare innebär metoden att buffern på databussen, 8304, måste avge databussens information under NADS-tiden, och enklaste sättet att ordna det är att låta data strömma ut kontinuerligt utom just då processorn skall läsa av någon yttre enhet. I den här datorn är den saken ordnad med en annan grind i IC18, vars ena ingång är kopplad till RDS (som går hög då processorn läser) och vars andra ingång är kopplad till IC17, en del av adressavkoda-

ren. (Närmare detaljer om denna nedan). IC17 stift 6 avger en positivt gående signal då någon yttre enhet skall läsas. Med båda ingångarna i grinden höga blir utgången låg, och signalen T/R till 8304 skiftar dataströmmens riktning. IC19 är en krets av typ 7414, dvs en motsvarighet till 7404 men med hysteres. Två av inverterarna används för att alstra en negativt aktiv återställningssignal, NRST, som dels går till processorn och dels är kopplad till busskontakten för att ge samma service åt yttre enheter. Vid spänningstillslag är C3 urladdad. Därför är stift 6, IC19 låg. C3 laddas emellertid upp via R15. Efter ett tag har då spänningen över C3 nått omslagsspänning för IC19, varvid dess utgång, stift 6 är hög. Därvid är man försäkrad om att processorns alla interna re-

gister är nollställda och börjar läsa ur minnet (i detta fall 8295) från början. Genom att jorda IC19, stift 6, kan man även när som helst avbryta processorn i sitt arbete och få den att börja från början igen. (I fallet med 8295 behövs dock inte den åtgärden).

Kretsen 7414 har sex inverterare varav vi har behandlat två. Två av de övriga används som buffer enbart, och sitter i ledningen NWDS från processorn. Av de två återstående, som sitter i NRDS, fungerar den ena som inverterare och buffer, medan den andra bara buffrar.

Som har framgått av resonemang om dataströmmens riktning från IC12, är det ofta praktiskt med en positivt gående signal då processorn skall läsa. Därför finns både NRDS och dess inverterade motsvarighet RDS utdragna till

busskontakten. Båda är alltså buffrade och kan driva nio TTL varde-
ra, liksom även NWDS. (NRDS
står för Negative Read Data Stro-
be. NWDS står för Negative Write
Data Strobe).

Adressavkodning med tre kretsar

Så till datorns mest invecklade del,
adressavkodaren. Den består i hu-
vudsak av tre kretsar, IC15, IC16
och IC17. Avkodaren har till upp-
gift att välja ut det av de på kortet
befintliga minnena som processorn
önskar kommunicera med, samt
att till detta minne alstra en nega-
tivt aktiv signal som är kopplad till
minnets Chip Enable, $\overline{CE}$. För att
få veta vilket minne som skall akti-
veras, känner man av de mest sig-
nifikanta adressbitarna (A12–A15
hämtas från databussen). A13–A15
bestämmer om något av
minnena på kortet skall aktiveras
eller ej, och A10–A12 bestämmer i
så fall vilket av minnena det är frå-
ga om. Avkodaren får därigenom
ett adresseringsområde om 8 kby-
tes (4 k ROM och 4 k RAM) och en
upplösning, som det heter, om 1
kbyte, dvs en utgång från avkoda-
ren täcker 1 kbyte. Slutligen är av-
kodaren så konstruerad att man
med en DIP-switch med tre kon-
takter kan välja inom vilket adress-
område man vill att kortet 8 kbytes
skall ligga. Det kan nämligen hän-
da att man i något sammanhang in-
te vill använda sin Basic-interpre-
tator. Då kan man genom att ställa
DIP-switchens kontakter i någon
annan kombination "flytta" minnet
i adress så att processorn efter re-
set börjar läsa någonting annat,
yttre minne, som man i stället givit
adressen 0000.

Avkodarens kopplingsschema
framgår av fig 23.6. NADS från
processorn inverteras i en NAND
och byter namn till ADS. Denna
signal går hög samtidigt som adres-
sens mest signifikanta bitar finns
att hämta på databussens D0–D3.
D0 är kopplad till D-ingången på en
7474, som låser in informationen
på D0 till Q-utgången då ADS går
hög. Signalen på Q-utgången är
alltså A12. Denna är kopplad till en

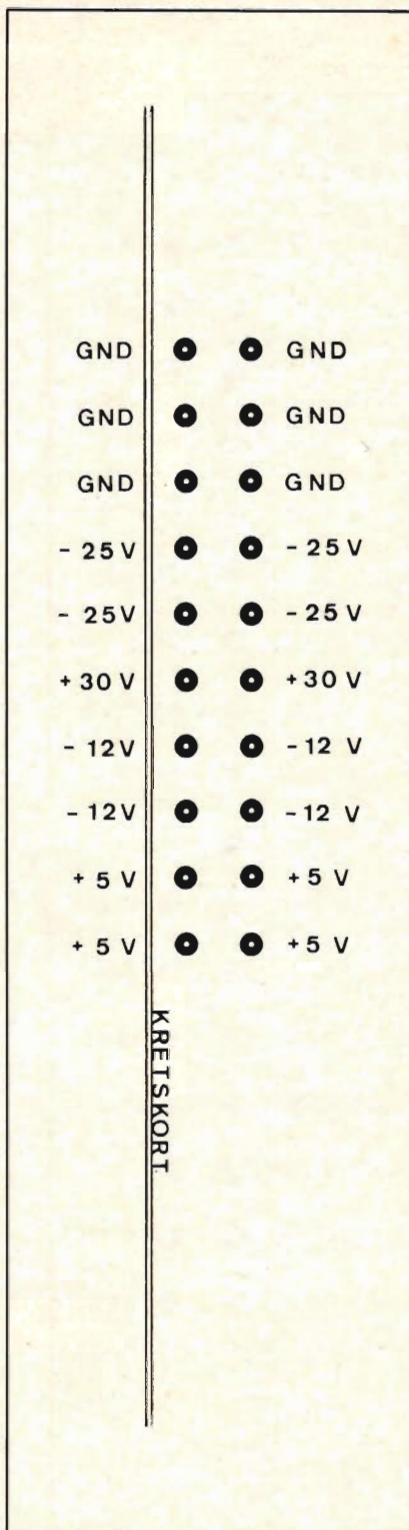


Fig 23.9. Spänningsmatningens kontaktkonfiguration i μ CD-korten. I basic-kortet används endast GND och +5V. Kontakten ses rakt bakifrån.

av de tre adressgångarna i 74LS138, avkodaren. De andra två adresserna, A10 och A11, tas direkt från adressbussen. G1 är en positivt aktiv enable-ingång på LS138. Till den är kopplad utgången på en NAND, vars ingångar får NWDS och NRDS. Det innebär, att eftersom både NWDS och NRDS är negativt aktiva och de inte kan vara aktiva samtidigt, kommer G1 att vara låg normalt men gå hög så fort processorn önskar skriva eller läsa. G1 är emellertid grindad med G2A och G2B, som är negativt aktiva. (Grindningen sker internt i kretsen.) Villkoret för att något skall hända på någon av utgångarna är alltså att G1 är hög samtidigt som både G2A och G2B är låga. Då går den utgång låg, som svarar mot kombinationen på ingångarna A, B och C.

74LS85 är en 4-bits binär komparator. Den har bland annat en utgång som heter A=B. Den går hög endast då ingången A1 har samma status som ingången B1, A2=B2, A3=B3 och A4=B4.

I alla andra fall är utgången A=B låg. Ingångarna A4 och B4 används inte utan är lagda konstant höga, och är därför inte utritade i fig 23.5. Om DIP-switchens alla kontakter är slutna, är kretsens B-ingångar låga. Då måste även A-ingångarna vara låga för att utgången A=B skall gå hög. Det inträffar om D1, D2 och D3 på databussen är låga, dvs de tre mest signifikanta adressbitarna A13, A14 och A15. Samtidigt som NADS är aktiv. Varje omkopplare i DIP-switchen svarar alltså mot sin adressbit. Vid switchen på kretskortet finns märkningen 0, 1 och 2. 0 svarar mot adressbit 13, 1 mot A14 och 2 mot A15. Den fjärde omkopplaren är märkt -, och är inte inkopplad.

Nu antar vi att DIP-switchens inställning stämmer överens med adressen. Utgången A=B på 74LS85 går då hög. Därefter går ADS hög, och D-ingångens på den ena vippan information kopieras över till Q-utgången, dvs den går hög. Andra utgången ger då ifrån sig inverterad information, dvs en nolla, som är kopplad till G2A/G2B

på 74LS138. Därefter går ADS åter låg, medan informationen i utgångarna på 7474 kvarstår oförändrade. Då sedan tiden är mogen för processorn att läsa eller skriva på den begärda adressen, går NWDS eller NRDS (aldrig båda samtidigt) låg, varvid G1 går hög, och den utgång hos 74LS138 som svarar mot kombinationen på A, B och C går låg, och det adresserade minnet avger sin information. Allt detta går mycket snabbt, det är fråga om delar av en mikrosekund.

Fyra av utgångarna från 74LS138 är kopplade till varsin diod, som tillsammans bildar en AND-konfiguration, dvs endast då alla fyra utgångarna samtidigt är höga, är punkten ROM ENABLE hög. Denna koppling är nödvändig därför att adressavkodarens upplösning är 1 kbytes, medan 8295 (ROM:et) omfattar 4 k. Därför måste även en signal som är kopplad till $\overline{CE}$ på 8295 ha upplösning 4 k, vilket kan uppnås på detta sätt.

Signalen ADSOK i schemat, *fig 23:5* används som tidigare nämnts för att reglera dataströmmens riktning i buffern 8304. (ADSOK är en signal som förekommer i de flesta datorer). I MyCD:s konstruktioner definieras den som: Processorn önskar kommunisera med någon av enheterna på detta kort".

Det kompletta kopplingsschemat framgår av *fig 23:3*. På kortet skall även monteras två kontakter: En 34-polig för busskontakten och en 20-polig för spänningsmatningen. Ett passande nättaggregat ger 5 V, 12 V, 25 V, och 30 V.

Då Mylady inte används med NIBL (Basic-interpretatorn), skall den kunna kommunicera med andra organ än en teletype. Tangentbordet till Mylady använder dels 5 V, dels 12 V, därav den senare spänningen. Vidare är Mylady anpassad för printerinterfacet PICO, som styr 14-kolumners värmetryckverket 4506 från Facit, som kräver 25 V. Vidare finns i Mylady-serien ett EPROM-kort om 16 k med programmeringsmöjligheter på kortet. Detta kräver 30 V. I övrigt är Mylady så långt möjligt avsett för 5 V-system.

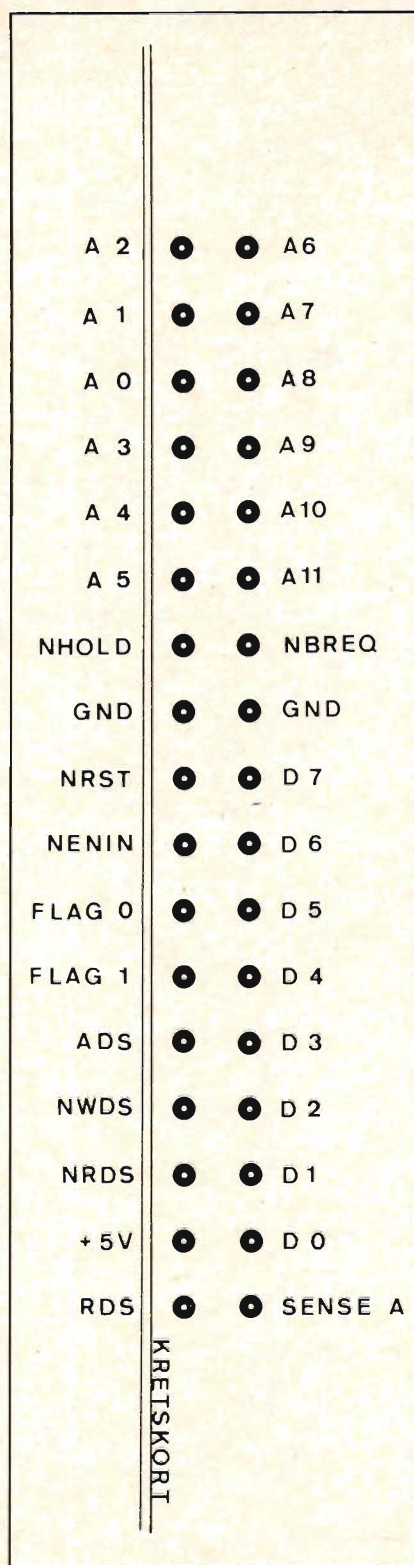


Fig 23:10. Busskontaktens konfiguration. Följande signaler används ej i basic-datorn: + 5V, Flag 0 Flag 1.

Montera komponenterna på kretskortet

Själva monteringen av kretskortet bör inte vålla den händige många bekymmer. Enklast är att montera de diskreta komponenterna först, löda dessa och klippa benen på undersidan. Men spara på skräpet!

Det kommer till nytta då stiftningarna skall göras. På kortet finns nämligen åtta punkter där en genomföring skall göras från kortets ovansida till dess undersida. Det kan enkelt utföras genom att de överblivna benen från motstånd och liknande sticks genom dessa hål och löds på kortets båda sidor. Punkterna är i komponentplacerings-skissen, *fig 23:6*, utmärkta med ofyllda cirklar. Observera även att motståndet R14 i ena änden också skall lödas på båda sidor av kortet. Detta gäller alltså om du inte kostat på genompläterade kort som är ganska mycket dyrare.

Ur servicesynpunkt är det mycket lämpligt att sätta kretsarna på hållare. De hållare som rekommenderas av MyCD finns att inhandla hos Elfa, betecknade *Augat serie 700*. Hållare sätts på plats och löds på kortets båda sidor. Var noggrann med lödningen på ovasidan, så att du undviker kortslutning mellan två intilliggande ben eller missar någon lödning eller gör en kalllödning! Då hållarna är lödda på plats är det dags att avlägsna en plåtbit på kretshållarna som håller stiften på plats. Därefter kan man sätta kretsarna på plats. Var noga med att se till så de vänds rätt (i komponentplaceringsskissen är stift 1 på varje krets utmärkt med en punkt), att resp krets hamnar på rätt ställe, samt att kretsarna behandlas med stor försiktighet med avseende på statisk elektricitet.

Undvik att ta på kretsarnas ben eller att ta på ledningsdragningen på ett monterat kort. Använd lödpennan med fin spets och jorda spetsen, om du måste löda på ett monterat kort. DIP-switchens (S1) ben är för grova för att gå ner i de rekommenderade hållarna, och det finns heller ingen anledning att sätta den på hållare. Beträffande R16, är det i

sig en integrerad krets, innehållande 7 st motstånd med samma värde. På kretsen finns en punkt som utmärker stift 1. Se till att den vänds rätt efter markeringen på kortet och i komponentplacerings-skissen.

Till spänningsmatningskontakten finns en motsvarande kabelhona, som med hjälp av ett enkelt skruvstycke kan pressas på en regnbågskabel. Efter kontroll att allt står rätt till med kretsarna, kopplas spänning in enligt fig 23:8. För att undvika spänningsfall i kabeln, bör alla ledare märkta GND kopplas till noll och alla ledare märkta +5V kopplas till 5-volten. Kommunikationen med teletype kopplas enligt anvisningarna för denna och i enlighet med märkningen på kortet och komponent-placeringsskissen. Då matnings-spänningen slagits till, är det bara att köra . . .

Inkoppling av externa enheter

Mylady kräver inget Europarack eller moderkort för att kunna byggas ut. I stället finns på varje kort fyra monteringshål, ett i varje hörn. Avsikten med dem är, att det understa kortet monteras med distanser på lådans botten, och därefter övriga tillsatsmoduler ovanpå detta. Mellan botten och första kortet kan man använda 10 mm distanser, men mellan korten rekommenderas 20 mm distanser för luftväxlingens skull.

Vid den elektriska sammankopplingen mellan korten används busskontakten, som är utformad enligt fig 23:7. (Observera att figurerna gäller hela systemets konfiguration. Vissa signaler och spänningar används inte av Basic Computer. Dessa är dock markerade i figurerna). De externa enheterna som i dag ingår i Mylady-systemet är:

Processor Board: Processor, 4 k × 8 EPROM, kristall (4 MHz), buffertar.

EPROM Board: 16 k × 8 EPROM, buffertar. Möjlighet att programmera minnena på sina respektive platser.

RAM Board: 8 k × 8 RAM. Switchade till standby då de inte läses eller skrives. Buffertar.

I/O Board: Avsett för applikationer utan NIBL eller modifierad NIBL. Ingång från tangentbord med parallellsnitt, utgång till skrivaren PICO, in- och utgångar för kommunikation med kassettbandspelare.

Video Board: Det presenterat 1024 tecken på en videomonitor eller över modulator på en TV-skärm.

PICO: Skrivaranpassning till Facits tryckverk 4506. Kommunice-rar med I/O Board.

Mylady power pack ger spänning till samtliga hittills framtagna enheter till *Mylady*. Samtliga enheter är busskompatibla med varandra och ansluts helt enkelt genom att en ny kontakthona pressas på busskabeln och spänningsmatningskabeln varefter kontaktarna ansluts till kortet. ■

Komponentförteckning

IC1, IC2, IC3, IC4, IC5, IC6, IC7, IC8	2114	1 k × 4 RAM
IC9	8295	4 k Basic-interpretator
IC10	ILD-74	Optokopplare, 2-kanals
IC11	8060	8-bits microprocessor
IC12	8304	8-bits dubbelriktad buffer/ driver
IC13, IC14	8097	6-bits enkelriktad buffer
IC15	74LS85	4-bits komparator
IC16	74LS138	3-bits avkodare
IC17	7474	dubbel D-vippa
IC18	7400	quad 2-input NAND
IC19	7414	hex inverter med hysteres
S1		DIP-switch
T1, T2, T3, T4	BC237	Transistor
C1, C3		Tantalelektrolyt, 10 µF, 16V
C2		Skiv, 100 nF
C4		Skiv, 27 pF
D1, D2, D3, D4	1N4148	Dioder
R1, R2		6,8 k, 1/4 W
R3		470 Ohm, 1/4 W
R4, R8, R10, R11, R12, R13, R14, R18, R19		1 kOhm, 1/4 W
R5, R6		82 Ohm, 1/4 W
R7, R17		56 Ohm, 1/4 W
R9		100 kOhm, 1/4 W
R15		10 kOhm, 1/4 W
R16		Resistansnät, 7 × 1 kOhm, 1/4 W
X		Styrkristall, 4,0 MHz
Övrigt		Busskontakt, 34-polig Spänningsmatningskontakt, 20-polig Lödstift till teletypekom- munikation (motsv) Kretskort

Byggsatser säljs inte, däremot mönsterkort som kostar 175 kr. Kortet kan köpas monterade och testade för 1800 kr. Moms och frakt tillkommer. Försäljningen sker genom tillverkaren: **HB MyCD, Micro Computer Development**, Björnsonsgatan 175-177, 161 56 Bromma, tel. 08-37 54 66.

Programmering i maskinkod

AV TOMMY BLADH

Maskinkod eller maskinkod-programmering är ett uttryck vi ser ofta.

Här berättar vi om varför det kan vara lämpligt att arbeta på en så låg språknivå ibland, trots maskinspråkets begränsningar.

Läs gärna först avsnittet "Hur fungerar mikrodatorn". Där framgår hur maskinspråskoderna används och var de har för verkan. Se även avsnittet "Assemblerprogrammering."

■ I tidigare avsnitt har vi beskrivit hur datorn får sina direktiv. Hur all programmering till syvende och sist syftar att ge vår dator en aldrig sinande rad av maskinkod. Det enklaste och smärtlösa sättet att generera denna eftertraktade kod är att använda en assembler eller ännu bättre – en kompilator.

Nu är vi inte alltid så lyckligt lottade att vi omedelbart har tillgängliga resurser för att köra en assembler, än mindre än en kompilator. Den senare kräver dessutom vanligtvis ett flexskrivsystem med en någorlunda vettig printer. Även i en sådan miljö av "mjölk och honung" kan man ha glädje av att tillverka bytarna "the hard way".

Programmering i maskinkod innebär att man själv får skriva in i datorns minne de bitmönster som utgör dess instruktioner. Vi får själva välja rätt operationskod, rätta adresser vid hoppinstruktioner och vid dataminnes manipulationer. Utan hjälp av något stödsystem måste vi själva avgöra adresseringsmoder och beräkna värden för relativa adressoffset, både vad gäller positiva och negativa tal. Det säger sig självt att detta inte blir någon lätt uppgift om kodvolymen blir stor. Främst är kanske svårigheten att rätta programmen

men även dokumenteringen kan bli ett problem. Därför är det inte så vanligt att större mängder kod byggs upp genom maskinkodprogrammering. Visserligen har man hört exempel på kilobytevis av maskinkodade program. Sådana heroiska ansträngningar hör dock förmodligen till undantagen. Vanligtvis knappar man in små snuttar för att kunna testa en dators olika delar strax efter hopskruvandet av den. Då kanske bara monitorns lilla "scratch"-minne är att lita på. Däri lägger man då små korta skriv/läs-slingor som adresserar den misstänkta delen. Sedan kan ett oscilloskop med bra trigg få undersöka de olika pulserna på adress och data buss samt kontrollsignalerna. Att använda datorn på detta sätt för att felsöka i sig själv är mycket effektivt.

Maskinkodprogrammering under test av program

Ett annat läge då maskinkodprogrammering kan vara vettig är vid *patchning* av program under uttestning. Patch betyder lapp och det man gör är att ändra felaktig kod direkt i datorns minne. Man lappar temporärt koden eftersom detta upp till en viss gräns går betydligt snabbare än att gå hela svängen över editering och assemblering. I bland måste vi kanske kompilera och länka innan ny objektkod föreligger. Att dra ut stora listor är heller inte försumbart i tid. Även med relativt snabba printar kan det uppstå irritation över väntetiden. Först när antalet fixar blivit tillräckligt stort eller ett problem är orealistiskt att bearbeta för hand, kallar man in editorn och korrigerar felen på källkodsnivån.

En del fel kan direkt ändras på befintlig plats i minnet. Den nya operationskoden eller adressen skriver bara över den gamla koden. Vanligare är dock att en eller flera instruktioner måste inskjutas

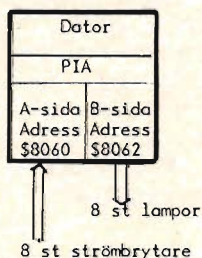
någonstans bland redan befintlig kod. Enklaste sättet att klämma in de saknade instruktionerna är genom ett uthopp till ledigt minnesutrymme. Där lägger man kanske först dit den eller de instruktioner som måste utgå för att ge plats åt själva uthoppsinstruktionen. Sedan är det bara att fortsätta med den nya sekvensen. Skall programkontrollen därefter återgå till originalkoden direkt efter uthoppet avslutar man lämpligtvis med en RTS, Return from Subroutine. Det förutsätter att uthoppet skedde med en JSR, Jump to SubRoutine, eller motsvarande typ av instruktion. Om programmet nere i patchkoden måste grena sig och återgå till olika ställen i originalkoden, hoppar man lämpligen ut och tillbaka med JMP instruktionen. Då sker ingen förändring på maskinstacken som vid subrutinanropen.

Skapa blanketter för dokumentation

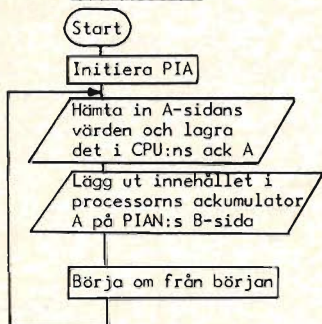
Som tidigare sagts är maskinkodprogrammering av större kodmängder något man helst bör undvika. För att överkomma problemen med dokumentationen kan man skapa blanketter med fält för de delar som en assemblerlista brukar bestå av. Exempelvis maskinadress, operationskod och parameterdel i hexadecimal notation. Därefter fält med symboliska namn för lägen, mnemonics och instruktionens parameterdel. Slutligen frikostigt med utrymme för kommentarer.

Eftersom det är svårare att snyggt införa ändringar efter avslutad kodning, bör man vara mer noggrann vid systemeringen och skrivbordstesta flitigt. Att bygga upp systemet genom nedbrytning i väl utprovade och definierade subrutiner är alltid viktigt. Kanske är det ännu förtjänstfullare vid programmering i maskinkod.

Uppgift



Flödesdiagram



Mnemonics

Initieringen överhoppad i exemplet. Den framgått av datablad och assemblerprogrammet, sid 101 ger viss ledning.

LDA A \$8060

STA A \$8062

JMP \$0100

Maskinkodprogram

Adress	Kod
0100	B6
0101	80
0102	60
0103	B7
0104	80
0105	62
0106	7E
0107	01
0108	00

Koder: LDA A = B6
STA A = B7
JMP = 7E

Här ser vi ett enkelt exempel på maskinkodprogrammering. Datorns uppgift är att ta in värdena från 8 strömbrytare och sedan presentera dem på 8 lampor. Lösningen av problemet tillgår så att vi först ritat ett flödesdiagram. Sedan tar vi reda på vilka instruktioner som datorn behöver för att fullgöra uppgiften. Slutligen skriver vi upp de koder som svarar mot instruktionerna. Det färdiga maskinprogrammet ser vi längst till höger. Programmet har vi här låtit börja på adressen 0100. Initieringen av PIA innebär att vi bestämmer om de olika portarna skall fungera som in eller utgångar.

Man måste kunna datorns arbetssätt

För att kunna maskinprogrammera måste man ha känsla för hur datorn arbetar med operationskoder och olika parametrar. En instruktion, exempelvis en LDAA vilken överför en byte från minnet till ackumulator A, har olika kod beroende på hur aktuell data skall adresseras. Data kan hämtas med kort eller full adressdel. Ett indexregister kan vara del i adressen, osv. Alla de här sakerna kan man läsa om i bruksanvisningar som fabrikanter och tillverkare mer eller mindre pedagogiskt komponerat. Ett bra komplement till läsandet i en manual är att studera en komplett assemblerlista. Där kan man lätt se hur de vanligaste instruktionerna

omvandlas till hexadecimal maskinkod i olika kombinationer. Har man väl fattat principerna räcker kodnyckeln i dragspelsformat oftast till för att välja rätt instruktion.

Vissa instruktioner har som parameter ett offsetvärde i tvåkomplementform. De villkorliga hoppinstruktionerna, exempelvis BEQ, är av den typen. Offsetvärdet består av skillnaden mellan nästa instruktions maskinadress och adressen dit man (eventuellt) skall hoppa. Hoppa man framåt är talet positivt, går hoppet däremot mot lägre adresser blir offsetvärdet negativt. I 6809 förekommer dessutom både korta (8 bit) och långa (16 bit) offset. För att vi skall slip-

pa den här besvärliga uträkningen är alla de i RT bygget aktuella monitorprogrammen (TBUG/CBUG/F0BUG etc.) utrustade med en funktion som automatiskt kan göra offsetberäkning. I 6809 fallet levereras ett 16 bit offset om 8 bit inte räcker.

● Sammanfattningsvis kan man säga att visst går det att framgångsrikt programmera i maskinkod. Men så fort som lite större mängder kan förutses, bör man allvarligt fundera på en mnemonic assembler. Den besparar med säkerhet en från många onödiga frustrationer. Förutsättningen är förstås att vi har en alfanumerisk terminal till datorn. Med en terminal för enbart hexkod kan vi bara arbeta med maskinkod. ■

Assemblerprogrammering

AV TOMMY BLADH

Assemblerprogrammering innebär bl a att vi arbetar med memosymboler, mnemonics, i stället för hex- eller binärkod som i maskinkodprogrammering. Språkmässigt ligger assemblern på en högre nivå.

Genom att vi inför hjälpmedel som assembler och editor i vårt system kan vi skapa program som kan dokumenteras bättre, göras lättare att ändra och vi når snabbare en acceptabel tillförlitlighet i systemet.

■ Det vanligaste språket som en dataamatör programmerar i, näst *basic*, är *assembler*. Assembler-språket betraktas inte som ett högnivåspråk, likt exempelvis *fortran*, trots att vi flyttat oss till en miljö där symboler används i stället för absoluta referenser. Det beror på att vi med *assembler* fortfarande befinner oss på en maskinnära nivå där målsystemens alla egenskaper kan maximalt utnyttjas.

När vi använder en kompikator för omvandlingen av högnivåspråkets satser (källkod) till exekverbar ("körbar") kod, objektкод, får vi räkna med att nedbrytningen sker på ett standardiserat sätt. Vissa generella mallar kommer att anbringas och bestämma målsystemens sammansättning. Kompilatorer kan dock skrivas så att den kod som produceras blir minst lika bra som vad den genomsnittlige assemblerprogrammeraren kan åstadkomma på samma uppgift. Men det sker på en bråkdel av tiden och med betydligt mindre antal fel i programmet. Det är här som den stora vinsten med högnivåspråk ligger: Bättre möjligheter till att strukturera vilket ger vinster

genom snabbare konstruktionsfas med mindre felprocent, bättre förståelighet och därmed lättare underhåll. Då alla de här sakerna är betydelsefulla i projekt där tid och pengar är viktiga, försöker man undvika assemblerprogrammering och så kvickt som möjligt hitta ett lämpligt högnivåspråk. På stora företag grinar man illa när assembler kommer på tal. Man vet hur svår den är att strukturera vettigt och känner problemen med dokumentation och underhåll.

Nu är vårt lilla hemdatasystem inte direkt jämförbart med de stora firmornas och deras problem är inte alltid samma som våra. Därför behöver vi inte tveka utan kan hugga i med uppgiften att lära oss assemblerprogrammeringens ädla konst.

Symbololiska referenser

Genom att införa assemblerspråket slipper vi programmera med absoluta referenser. Det innebär att instruktionerna som datorn skall få inte behöver anges med hexadecimala tal, att adresser till variabler och tabeller inte måste vara kända då programmet skrivs. I stället använder vi *symboler* för att referera till de olika storheterna. En symbol är här vanligtvis ett namn som består av vanliga bokstäver. De brukar kunna vara upp till sex tecken långa. I regel måste symbolen inledas av en bokstav (A–Z). Exempel på symboler kan vara *ACIA*, *SLASK* och *LAB120*. De speciella symboler, mnemonics, som representerar operationskoder kan se ut som: *LDA A*, *ASR*, *LEAX* etc. Assemblatorns uppgift blir sedan att lösa ut symbolernas värden och sammanställa den maskinkod i ettor och nollor som är det enda vår dators centralenhet begriper. Jämfört med maskinkodprogrammering har vi nu fått en betydligt lättare uppgift. All tidsödande och felproducerande uträkning av operationskoder och referenser sköter

assemblatorn om. Vi kan istället koncentrera oss på det som är vår uppgift, nämligen att konstruera ett program för att lösa ett specifikt problem. Här är vi utelämnade till oss själva. Datorn är i sig dum som ett spån. Det är endast tack vare människan, vilken med möda knäpat i hop det program som datorn utför, som mer eller mindre "intelligens" kan skönjas hos denna samling av halvlederbrickor, kablar och annat järnskrot.

Systematisering

Vid all programmering är det viktigt att systematisera. Assemblerprogrammeringen utgör därvid inget undantag, snarare tvärtom. Genom att vi fortfarande befinner oss på en maskinnära nivå utför vi med varje instruktion relativt lite. Det mesta som sker är laddningar och undanlagringar av data bytes, testningar efter utförda enklare manipulationer och olika åtgärder som resultat därav. Resultatet av de många små operationer kan dock bli ett stort och intressant program. Men det kan lika gärna bli rena rama smörjan: Ett rättbo av kod som nätt och jämt håller samman, som knappt går att rätta i än mindre begripa funktionen. Vilket av alternativen som skall gälla beror på programmeraren, han/hon och ingen annan.

Det första som händer när man ser den uppgift som skall lösas, kan vara en känsla av lätt förtvivlan. – Hur ska jag kunna ro i land det här, tänker man kanske. I stället för att nu kasta sig över tangentbordet, sätter sig den erfarne programmeraren vid skrivbordet och försöker *systematisera* problemet. Han/hon börjar att bryta ned den från början komplicerade uppgiften i mindre, lätthanterligare delbitar. Varje bit skall ha ett naturligt "ansvarsområde". Sådan nedbrytning kan fortsätta ytterligare i flera nivåer tills så småningom varje del blivit enkel att program-

2 JULI 1980

```

1  * ( MAINPRO )
2  *
3  * EV TBL 80-07-02 REV 1
4  *
5  * BESKRIVNING:
6  *
7  *
8  *
9  * MAIN ÄR ETT KOMMANDO UNDER FLEX 2.
10 * OMKOPPLAREN HASTAR I ATT ÅSKA EN ÅTTA
11 * OMKOPPLARE ANSLUTNA TILL EN PIA OCH
12 * OCH TANDA RESPEKTIVE SLACKA ÅTTA
13 * LYSDIODER SÅ ATT DE SPEGLAR OMKOPPLARNAS
14 * STATUS. DA SÄMMLIGA ÅTTA OMKOPPLARE
15 * ÄR TILL ANSLUTNA MAIN. UTSKRIFTEN "KLART"
16 * SKRIVS UT OCH KONTROLLEN ÅTERGÅR TILL FLEX.
17 *
18 *
19 * EXTERNA REFERENSER:
20 *
21 * PSTRNG EQU $ADIE SKRIV UT TEXTSTRANG (FLEX).
22 * WARM EQU $AD05 ÅTERGÅNGSPUNKT FLEX.
23 *
24 *
25 * LOKALA REFERENSER:
26 *
27 * PSTART EQU $A100 PROGRAMMETS START ADRESS.
28 * PIAACR EQU $8061 KONTROLLREGISTER SIDA 8
29 * PIAACR EQU $8063 KONTROLLREGISTER SIDA 8
30 * PIAACR EQU $8060 DATAREGISTER SIDA 8
31 * PIAACR EQU $8062 DATAREGISTER SIDA 8
32 *
33 * ORG PSTART OF ASSEMBLERN STARTADRESSEN
34 *
35 * A100 20 01 START BRA STRT HOPPA ÖVER NÄSTA BYTE
36 * A100 01 FCR 1 VERSIONSNUMMER, FLEX KONVENTION
37 * A100 80 A1 16 STRT JSR INIT SATT UPP PLEN
38 * A100 80 A1 20 MAIN0 JSR OKTEST TESTA OMKOPPLARNA
39 * A100 27 05 BEQ SLUT TESTA OM AVSLUTA
40 * A100 80 A1 39 JSR LIGHT TAND LYSDIODEN
41 * A100 20 F6 BRA MAIN0
42 *
43 * A110 80 A1 10 SLUT JSR FINISH AVSLUTA
44 * A110 76 A1 03 JSR WARM
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *

```

```

60 * INGA
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *

```

mera. Varje sådant element i strukturen får ett definierat snitt mot omvärlden. Dess uppgift är noga definierad. Indata och utdata levereras på specificerat sätt. Rutinen meddelar självt anropande program genom en returkod i fall den har kunnat utföra sin uppgift eller om något felfall uppträtt.

Subrutinen

Element av den här typen kan vara

en subrutin, vilken anropas med en speciell instruktion (ex JSR), och avslutas med en RTS instruktion. Vid anropet av subrutinen läggs automatisk adressen till efterföljande instruktion på maskinstack. Detta gör att programkontrollen kan återvända då instruktionen RTS i subrutinen utföres. Den instruktionen flyttar återhoppssadressen från stacken över till programräknaren. Tack vare det kan gene-

```

120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *

```

NO ERROR(S) DETECTED

SYMBOL TABLE:

```

FINISH A100 INIT A116 LIGHT A139 MAIN0 A106 OKTEST A137
OKTEST A130 PIAACR 8061 PIAACR 8063 PIAACR 8065 PIAACR 8062
PSTART A100 PSTRNG A011 SLUT A510 START A100 STRT A105
TEST05 A144 WARM A005

```

Den här programlistan visar hur ett assemblerprogram kan se ut. Programmet talar för sig självt genom möjligheten att lägga in kommentarer. Närmast till höger om programradens nummer, t ex 34, ser vi adresskoden, A100, instruktionen, 20, som innebär att vi skall hoppa och efter den det antal steg vi skall hoppa, i det här fallet 01 steg. Sedan har vi lagt in en kommentar att vi befinner oss på programmets START. Efter det ser vi BRA vilket är memosymbol för instruktionen 20 som vi redan nämt och som alltså innebär hopp i programmet. På sid 25 finns en instruktionslista för 6800. Granska gärna den jämsides med den här assemblerlistan!

rella subrutiner anropas från olika ställen i överliggande program och programkontrollen återgår alltid till rätt ställe. Har man sedan lyckats testa ut rutinen med diverse olika fall och fått den felfri, kan den användas över allt där dess funktion passar.

Dokumentation och deklarationer

Vid programmering i assembler

skall man vinnlägga sig om att dokumentera tillräckligt frikostigt för att man själv, eller någon annan i efterhand kan förstå vad programmet gör. Absolut förkastligt är att lova sig själv att "det fixar jag när det fungerar juste". Den kommentar som inte skrevs in direkt, kommer med största sannolikhet aldrig att se dagens ljus! Sådant är programmerarens psyke. Den typ av assembler som används i **FLEX** systemet ger direkt objektкод. Ingen möjlighet till länkning av separat assemblerade delar finns. Därför kan inte lokala namnliska variabler deklarerar i olika rutiner och assembleras tillsammans. Det gör att man bäst samlar deklarationerna i början på huvudprogrammet. Inledande bör en beskrivning av programmet vara. Då kan man snabbt förstå huvudfunktionerna. Detaljuppgifter kan läggas i respektive underrutin.

Programmeringsexempel

Eftersom vi inte av utrymmesskäl kan ge mer än en kortfattad orientering i assemblerprogrammeringens mysterier, skall vi avslutande visa ett exempel på hur det kan ta sig ut. Ett kort exempel bara. *Förutsättning:* En **PIA** har åtta omkopplare anslutna till A-sidan och åtta lysdioder på B-sidan.

Uppgift: Vid tillslag av en omkopplare skall motsvarande lysdiod tändas. Vid fråslag skall den slockna. Om alla omkopplare slås till skall vi lämna rutinen. När detta sker skall ordet "klart" skrivas ut på bildskärmen.

Den lösning vi tillämpar här avspeglar resonemanget tidigare. Anbringat på vårt lilla exempel blir resultatet möjligtvis lite omständigt. Men det gör inget eftersom det är principen som skall visas. Programmet har en huvudrutin kallad **MAIN**. I den anropas först en initierings rutin, **INIT** för uppstartning av **PIA:n**. När det är klart anropas en rutin, **OKTEST** som testar omkopplarna. Därefter får lysdioderna tändas av **LIGHT**. Avslutningen sker i rutinen **FINISH**. Låt oss nu se hur en assemblerlista kan se ut.



Bästa sättet att lära basic är att praktiskt öva vid datorn.

Någonstans måste man dock börja, och vi ger här en grundläggande introduktion.

Artikeln kan vara en bra första läsning för den som inte sysslat med programmering tidigare. Vi går igenom några grundbegrepp och resonerar ungefär så som man gör i det praktiska arbetet.

■ En dator är, liksom andra maskiner, värdelös om vi inte kan styra den att göra vad vi vill. Det en dator kan göra är att räkna ettor och nollor och kombinera dem på olika sätt. När vi styr en dator, ber vi den alltså att behandla ettorna och nollorna på ett önskat sätt. Det kan man göra med *maskinspråk*, som direkt talar om hur olika värden skall skyfflas runt bland register och ledningar. De instruktioner man ger en dator i maskinspråk är mycket svåra att läsa och överblicka för människor, men maskinen förstår kommandona desto bättre.

100010110

Ettorna och nollorna kan kombine-

ras till siffror och tecken, och programmet bildar sålunda långa tröstlösa rader av oläsliga sifferkombinationer.

Vi säger maskinspråk, och låter alltså påskina att datorns maskinspråk kan liknas vid mänskligt språk. Till en viss del kan man också karakterisera ett datorspråk på samma sätt som ett mänskligt. Framför allt ser vi språket som ett sätt att organisera den information vi vill överföra, och då kommer också en del av informationen att ligga i språkets form, precis som i det talade, mänskliga språket.

Maskinspråk är alltså svårörförståeligt för människor. Vi skall här i stället ägna oss åt ett s k högnivåspråk som kallas **BASIC**. Bokstäverna står för *Beginners All Purpose Symbolic Instruction Code*, men det engelska ordet *basic* betyder ju dessutom grundläggande.

Med ett högnivåspråk menar vi att språket i första hand skall vara lätt att förstå för operatören, dvs den människa som skall använda det. För att maskinen skall kunna förstå det krävs att den själv översätter basic till maskinspråk. Det sker vanligen i en s k *interpreter*, interpretator, eller tolk. I stället för att människan skall översätta maskinspråket till något som hon begriper översätter alltså maskinen basic till något som *den* begriper.

Det här innebär att man avlastar arbete från människan och lägger på maskinen. Maskinen, datorn, blir alltså mera belastad och behöver främst ett större minne för att utföra samma uppgift som med ett maskinspråkprogram. Dessutom tar uppgiften längre tid att utföra, dels därför att datorn ju måste översätta alla instruktioner, dels därför att programmet blir "grövre" och kanske tar ut onödiga "slängar" när det utförs.

Som mänskligt språk betraktat är basic synnerligen torftigt, men i konversation med en dator kan det vara mycket effektivt. Man bör komma ihåg att basic är ett *enkelt* språk, eftersom vi ger instruktioner i basic till datorn, medan den i regel inte svarar på det språket, utan på ett helt godtyckligt språk. Vi återkommer till detta!

Inlärnin g av datorspråk effektivast genom praktik

Det bästa sättet att lära ett datorspråk är att praktisera det med reella övningar. Inget annat sätt kan ge de omedelbara erfarenheter som underlättar förståelsen. Inget annat sätt är heller så lustbetonat – det är ju roligt att programmera!

Varför nu just basic? Jo, språket är mycket spritt och går att använda även på ganska små datorer. Det är lätt att lära och kan därför tjäna som inkörsport till andra språk, om man så vill.

Nu talar vi om basic som om det vore ett entydligt språk. Så är det emellertid inte. Generellt kan man säga att en stor dator (och med stor menar vi här kanske 32 kbyte minne) kan ha ett mycket utbyggt språk, medan en liten dator med *tiny basic* (på kanske blott 4 kbyte) förstår ganska lite och likaså kan göra ganska lite. Språket är alltså inte entydigt, utan kan delas upp i en mängd varianter eller dialekter. Tiny basic är alltså minsta möjliga basic, och det arbetar i regel bara med heltal t ex medan stora basic-tolkar t ex kan arbeta med korsningar som kallas fortranförstärkt basic.

Denna språkliga konfusion gör

att man egentligen inte kan beskriva ett språk utan att tala om vilken dialekt man rör sig i. Vi skall här försöka att i första hand tala om den delen av språket som är gemensam för de flesta dialekterna och därmed de flesta maskinerna, men vi kommer också att visa hur olika basic ser ut i vissa detaljer.

Enkelt exempel visar uppbyggnad

Låt oss nu börja samtala med vår dator på klingande basic! Som ett enkelt exempel antar vi att vi vill multiplicera 5 med 6. Vi måste då veta att datorn förstår våra vanliga arabiska siffror och att tecknet för multiplikation är *, asterisk. Vi bör också veta att multiplikationstecknet fungerar som vi är vana vid, eller med andra ord att det skall placeras mellan faktorerna. När datorn är i sådant läge att den kan ta emot kommandon från tangentbordet, visar den en *prompt*-symbol eller ett klartecken. Tecknet kan variera mellan olika maskiner, men vi kommer att använda * som klartecken. Tecken och resultat kan antingen skrivas ut på skrivare eller på bildskärm. Små datorer använder oftast bildskärm som utorgan, och våra resonemang knyter främst an till detta. En basic-dator med skrivare fungerar dock på i stort sett samma sätt. Vi skriver nu in vårt exempel i datorn om den visar klartecken:

5*6

När vi skrivit så långt, måste vi tala om att vi är färdiga. Annars väntar datorn på flera instruktio-

ner. Färdigkommandot består av en vagnretur, *carriage return*, och när man slår in den börjar maskinen arbeta.

Om vi nu ger datorn en vagnretur med ovanstående inslaget kommer den att svara

*

eller något annat av liknande innebörd. Det var inte precis vad vi ville ha. Har vi då gjort något fel? Både ja och nej. Datorn har helt visst räknat ut produkten av 5 och 6, så så långt är det rätt.

Men nu är dags att avliva en gammal myt om datorer som kanhända fortfarande lever kvar någonstans: Det sägs att datorn är väldigt intelligent, och att den kanske genom sin intelligens rentav är ett hot mot oss. Ett hot kan den möjligen var i fel sammanhang men i så fall inte tack vare sin intelligens, utan genom sin dumhet! Ty, som någon har sagt, *en dator är en komplett idiot men med en fantastisk förmåga att räkna!*

Vad vi nämligen bad datorn att göra i exemplet var att multiplicera, och det gjorde den. Vad vi där emot inte bad den att göra var att tala om svaret. Eftersom datorn saknar sunt förnuft måste vi tala om för den *allt* den skall göra. Den förstår inga underförstådda instruktioner som vi gör. På frågan – Hur mycket är 5 gånger 6? – svarar vi 30 därför att vi inser att frågaren vill veta svaret, inte bara att vi skall utföra beräkningen. Det här kommer igen hela tiden när man programmerar:

Allt måste talas om för datorn. För en ovan programmerare ger



100010110

det upphov till många lärorika fel. Så småningom lär man sig dock att tänka i mera fixa tankebanor efter en sträng logik utan intuition och nåd. I vårt fall vill vi alltså veta resultatet av multiplikationen. Instruktionen för datorn att tala om något heter **PRINT**. I likhet med övriga basic-instruktioner ligger denna mycket nära det engelska språket. **PRINT** kan alltså betyda att något skall skrivas ut på papper – men även att det skall visas på en bildskärm. Vi skriver nu

PRINT 5*6

och avslutar med vagnretur. Datorn svarar då strax

30

vilket vi väl bör vara nöjda med. Räkna är alltså datorn bra på, och de fyra räknesätten tillhör de grundläggande färdigheterna. Tecknen för dem är +, -, / och *, och vi kan använda dem alla i ett exempel

PRINT 3+4/2*4

11

Här kan man tänka sig ett antal rimliga svar, och olika datorer kan tolka uppgiften olika. Här får man konsultera bruksanvisningen för att se hur en aktuell maskin bär sig åt. För det mesta får man det svar som ges i exemplet, men för att vara på säkra sidan kan man sätta parenteser och avskilja de operationer som skall utföras först. Vi kan förändra exemplet genom att sätta in ett parentespar

PRINT (3+4)/2*4

14

och så vidare. Det finns också i basic en hel del färdiga matematiska funktioner som direkt kan användas i beräkningar och vi kan koka ihop ett exempel där vi använder ett antal av dem:

**PRINT INT(LOG(5)+
SQR(TAN(4.5)))**

3

Det där ser ju inte alltför glasklart ut, men är ändå inte så svårt att bena ut. En del lärdomar kan vi också skaffa oss genom exemplet. Första instruktionen är **INT** som står för integer, dvs heltalsdel av

något. Detta något är det som ryms inom parentesen som börjar efter **INT**. Alla matematiska funktioner som **INT**, **LOG** osv är så konstruerade, att argumentet, dvs utgångsvärdet för beräkningen, skall stå inom en parentes. Man får sedan inte glömma att avsluta parentesen, för då säger maskinen ifrån att man glömt något.

LOG står för logaritmen av ett tal. I regel betyder **LOG** den naturliga logaritmen med e som bas, men inte alltid. 10-logaritmen, om den finns, kan tecknas **LGT**. Om endast naturliga logaritmer finns, får man bilda 10-logaritmer genom att dividera den naturliga logaritmen för det sökta talet med den naturliga logaritmen för den önskade basen.

SQR står för *square root*, kvadratroten och **TAN** för tangent. Oftast skall vinkeln för trigonometriska funktioner ges i radianer, och man får då först räkna fram ett radianvärde om man har ett gradtal att utgå från. En del övriga matematiska funktioner visar vi i en särskild tabell. Observera, att vi som

decimalkomma måste använda punkt. Ett komma ger helt fel information!

Flera instruktioner bildar ett program

Låt oss nu återvända till vår första beräkning av 5 gånger 6! Vi kan sätta in siffrorna i ett konkret sammanhang och låta 6 betyda priset på en vara i en butik och 5 betyda 5 st av samma vara. På det totala beloppet vill vi också lägga på moms. Uträkning av priset före moms blir ju precis det exemplet vi haft tidigare. Förutom det priset vill vi dock ha priset med moms. Hur nu göra detta? Jo, vi har talat mycket om att *programmera* men än har vi faktiskt inte gjort så. Med programmera menar vi att ge datorn en räkka instruktioner att utföras efter varandra. Med den nya uppgiften måste vi nog göra ett program. Alla tidigare uppgifter skulle i och för sig lika gärna kunnat räknas på någon enkel räknedosa, men när det kommer till program krävs effektivare don. Hur



skall vi nu få datorn att förstå att vi nu vill lägga in ett program? Helt enkelt genom att ge varje instruktion, eller rad, ett nummer. När vi sedan vill använda programmet kommer instruktionerna att utföras i nummerordning. Vi skriver alltså

```
* 1 PRINT 5*6
```

och avslutar med sedvanlig vagnretur. Nu svarar datorn inte med produkten, och det beror på att vi har ett radnummer framför instruktionen. I stället lagras det hela i maskinens minne och finns tillreds för kommande bruk. Skillnaden mellan en programrad och en instruktion som skall utföras omedelbart (ett kommando, *command*) är alltså att programraden har ett nummer. Verkan av de båda informationerna blir således helt olika. Men vi fortsätter i programmet (och underförstår från och med nu klartecknet framför varje rad):

```
2 PRINT 5*6*1.2063
```

```
3 END
```

Nu har vi fått fram en följd av tre instruktioner som skall utföras efter varandra. Den tredje instruktionen markerar att programmet är slut. Ofta kan man utesluta den, och allt fungerar bra ändå. Datorn genererar då själv en **END**-instruktion på det sista radnummer som går att använda. Om den inte gör det, kan datorn råka in i diverse besvärligheter om man glömmat **END**. Tag därför som vana att avsluta programmen med **END**!

Verkställighet av instruktion olika startkommando för program

Hur skall vi nu köra programmet? Tidigare verkställde vi ju ett kommando med vagnreturen, men det går uppenbarligen inte här. Instruktionen för att köra ett program heter i stället **RUN**, och vi skriver så och ser vad som händer. **RUN** måste i sin tur avslutas av verkställighetskommandot vagnretur.

```
RUN
```

```
30
```

```
36.189
```

```
END AT LINE 3
```

Ett alldeles korrekt resultat.

Om vi tittar på vårt program, ser det dock en smula fånigt ut. Först har vi räknat ut vad 5 gånger 6 blev, och sedan har vi räknat ut det en gång till för att multiplicera resultatet med momsfaktorn. Bättre än så kan vi om vi inför en *variabel* i programmet. Med variabel menar vi här en symbol som vi tilldelar ett visst värde. Låt oss kalla priset utan moms för **X** så kan vi skriva om programmet på följande sätt, med en ny instruktion dessutom:

```
1 LET X = 5*6
```

```
2 PRINT X
```

```
3 PRINT X*1.2063
```

```
4 END
```

Det programmet ger samma resultat som det tidigare, men med mindre antal operationer utförda. I detta enkla fall spelar det naturligtvis ingen roll om vi räknar produkten en eller två gånger, men har man stora beräkningar blir väntetiden kanske prövande.

Den nya instruktionen **LET** innebär att vi definierar en variabel, i det fallet **X**. Som variabelnamn kan man använda en bokstav, som **X**, eller en bokstav plus en siffra, **X1**. Det finns dessutom hos utvecklade former av basic ytterligare variabelformer. Likhetsstecknet i **X=5*6** är egentligen inget likhetsstecken utan en markering för att **X** sätts till något. Vi kan t ex definiera en variabel som **X=X+1**, och det strider mot alla matematiska förutsättningar. Som definition i basic är det dock helt riktigt och betyder att värdet på **X** ökas med ett. **LET** är en instruktion som ofta inte behöver skrivas ut. Man kan i stället helt enkelt skriva

```
1 X=5*6
```

Vi har nu gjort ett i vårt tycke fiffigt program. Frågan är bara om det verkligen är så fiffigt. Vi förutsatte ju att programmet skulle användas av en affärsman som sålde saker och vill beräkna priset på dem. Om kunden envisas med att vilja köpa en vara med ett annat pris, eller ett annat antal varor än vad vi förutsett måste han faktiskt programmera om datorn för att kunna använda den! Och det kan ju knappast vara så fiffigt i längden.

Men det finns bot på nära håll. Vi kan föra in en ny instruktion i programmet. Instruktionen heter **INPUT** och gör att programmet kan stanna och vänta på värdet av en variabel. Vi måste då skapa en variabel för antalet varor, **A**, och en för priset, **P**. Med de nya variablerna och **INPUT**-instruktionen skriver vi ett nytt program

```
1 INPUT A
```

```
2 INPUT P
```

```
3 X=A*P
```

```
4 PRINT X
```

```
5 PRINT X*1.2063
```

```
6 END
```

När vi skriver **RUN** och verkställer får vi följande resultat

```
?
```

och datorn väntar då att vi skall svara med den första variabeln, följd av vagnretur. Likadant blir det vid andra **INPUT**-satsen, och hela svarsatskriften, med de inmatade värdena blir

```
?5
```

```
?6
```

```
30
```

```
36.189
```

men vi upprepar inte det i det följande.

Nu kanske vi skulle vilja sära in och utdata i utskriften för att få lite snyggare presentation. Det kan man åstadkomma genom att ge en **PRINT**-instruktion utan ytterligare information efter. Man får då en tom rad på skärmen eller papperet. En sådan **PRINT**-instruktion skulle behöva hamna före rad 4 i vårt program men efter **INPUT**-instruktion i rad 2, dvs den skulle behöva hamna ungefär på rad 3. Rad 3 är ju dessvärre upptagen, så för att få in vårt **PRINT** skulle vi vara tvungna att skriva om programmet efter rad 2 helt och hållet.

Programmets radnummer anger operationsföljd

Siffrorna i början av varje rad står där för att datorn skall veta i vilken ordning instruktionerna skall utföras. Det är emellertid inte nödvändigt att siffrorna följer direkt på varandra, man kan med andra ord ha mellanrum mellan programradnumren utan att det påverkar pro-

grammet. De programradnummer vi inte använder finns helt enkelt inte och tar inte upp någon minnesplats. Det för alltså inte med sig några nackdelar att numrera raderna med t ex intervallet 10 i stället för 1. Däremot för det med sig stora fördelar: Man kan lätt lägga till nya programrader mellan de tidigare utan att behöva skriva om någon del av programmet. I praktiken skriver man nästan alltid basic-program med 10 mellan radnumren. Om vi gör det här kan vi lätt sätta in vår extra **PRINT**-rad.

```
10 INPUT A
20 INPUT P
30 X=A*P
35 PRINT
40 PRINT X
50 PRINT X*1.2063
60 END
```

Observera att det inte spelar någon roll *var* vi skriver rad 35. Vi kan gärna skriva den sist i programmet, eller långt efter det att det övriga programmet skrevs. Det är bara radnumret som räknas av datorn. Utskriften blir då

```
?5
?6
30
36.189
```

förutsatt att vi matar in värdena 5 och 6 på rätt plats. Utskriften ser inte alltför klipsk ut, faktum är att man i efterhand knappast kan se vad som har gjorts. En räknedosa skulle klara uppgiften lika bra, ja även om beräkningarna vore mycket komplicerade så vore inte alltför mycket vunnet över en enklare beräkningsmaskin. Men, menar vi, det är just här som en av datorns stora förtjänster kommer in.

Vi vill gärna se datorn som ett hjälpmedel för hjärnan liksom hävstången är det för musklerna. Någon självständig "varelse" är datorn inte och därför är det hela tiden *programmeraren* och inte datorn vi konverserar med när vi använder ett datorprogram. Konversationen kan störas av bristfälligheter i datorns konstruktion, men programmeran är den som talar hela tiden. Många utländska terminaler klarar inte Å, Ä eller Ö, några

datorer med små minnen kan inte hantera text på meningsfullt sätt alls, men i regel kan programmeraren kommunicera alldeles utmärkt genom sitt program – om han vill. Den programmerare som med berätt mod lämnar en utskrift som den vi fick här ovan är närmast oförsämd mot användaren, och han kan inte sägas ha använt hjärnans hävstång på något klyftigt sätt. Likafullt ser man alltför ofta datautskrifter som har det principiella utseendet: Svårtydbara sifertabeller med hemliga koder som måste översättas. Typen är särskilt vanlig på räkningar och Myndighets meddelande angående skattepliktigt inkomst av jordbruksfästighet upptagande svin och liknande. I rättvisans namn måste sägas att de datautskrifter vi får har blivit bättre med åren, men fortfarande tycks det vara så, att det datorn ger ut skall vara lite svårt och chifferaktigt.

Vi tycker *inte* det lika lite som en tryckt text skall bestå av kodsiffror med nyckel för varje bokstav tillgänglig på sista sidan. Nej, Klarhet och Reda är honnörssord i vår kampanj på det här området! Och återigen: Om utskriften från ett program ter sig enfaldig så är det oftast inte datorn som står för enfalden.

PRINT-instruktionen

ger datorn talförmåga

Om vi analyserar **PRINT**-instruktionen närmare finner vi att den skriver ut de siffror eller *värdet* av

de variabelnamn vi skriver efter **PRINT**. Vi kan dock även skriva text med **PRINT** och för att särskilja texten från variabler där varje bokstav har ett värde skriver man då texten inom citationstecken.

PRINT "TEXT"

ger som resultat

TEXT

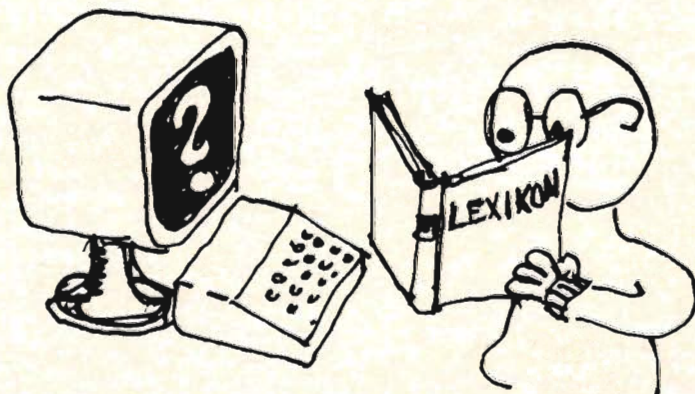
Därmed öppnas stora möjligheter att göra sig förstådd. Vi kompletterar vårt handlarprogram med diverse förklarande texter:

```
5 PRINT "ANTAL
VAROR"
10 INPUT A
15 PRINT "PRIS PER
STYCK(KR)"
20 INPUT P
30 X=A*P
35 PRINT
37 PRINT "SUMMA UTAN
MOMS"
40 PRINT X
45 PRINT "SUMMA MED
MOMS"
50 PRINT X*1.2063
60 END
```

Programutskriften med inmatad mängd och pris blir då

```
ANTAL VAROR
?5
PRIS PER STYCK (KR)
?6
SUMMA UTAN MOMS
30
SUMMA MED MOMS
36.189
```

En betydande förbättring, men det går att snygga till det hela ytterligare. För varje nytt **PRINT** eller **INPUT** skiftar datorn till en ny

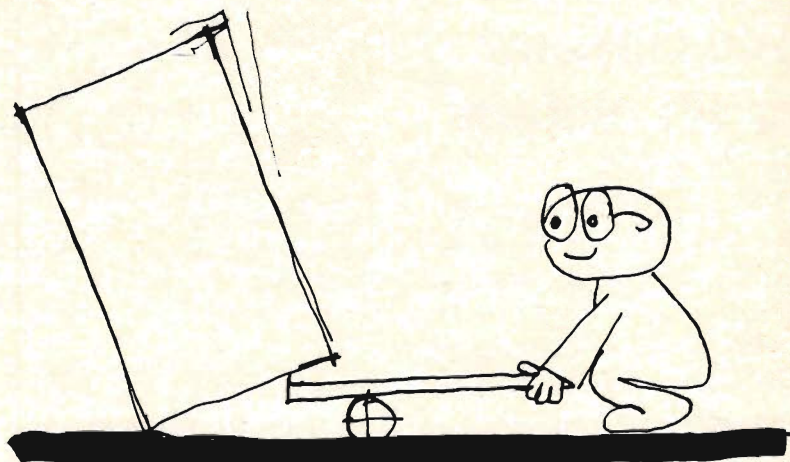


rad. Detta kan förhindras om vi avslutar den tidigare raden med ett semikolon ;. I utskriften kommer då de raderna att länkas samman, så att utskrifterna kommer på samma rad. Om vi gör det och fyller på några "KR" här och var, kommer utskriften att se ganska mänsklig ut. Det är ju också en människa som gjort den, om än med hjälp av en dator. Först programmet och därefter utskriften:

```
5 PRINT "ANTAL
VAROR";
10 INPUT A
15 PRINT "PRIS PER
STYCK (KR)";
20 INPUT P
30 X=A*P
35 PRINT
37 PRINT "SUMMA UTAN
MOMS";
40 PRINT X;
42 PRINT "KR"
45 PRINT "SUMMA MED
MOMS";
50 PRINT X*1.2063;
55 PRINT "KR"
60 END
ANTAL VAROR?5
PRIS PER STYCK (KR)?6
SUMMA UTAN MOMS
30 KR
SUMMA MED MOMS
36.189 KR
```

Där har vi en ganska väl doku-
menterande utskrift. Nu kan vi
emellertid förbättra själva pro-
grammet en smula. Vi har flera
PRINT efter varandra som vi vill
ha utskrivna på samma rad. Vi kan
faktiskt skriva flera saker efter
varje PRINT-kommando och på så
vis göra programmet lättare och
enklare att tolka för operatören.
Alltså:

```
5 PRINT "ANTAL
VAROR";
10 INPUT A
15 PRINT "PRIS PER
STYCK (KR)";
20 INPUT P
30 X=A*P
35 PRINT
40 PRINT "SUMMA UTAN
MOMS";X;"KR"
50 PRINT "SUMMA MED
MOMS";X*1.2063;"KR"
60 END
```



Det blir alltid något klarare så. Observera, att vi satt semikolon mellan de olika sakerna som skall skrivas i PRINT-satserna. Vi kan i stället separera med komma, men då får vi ett större mellanrum mellan de olika delarna. Hur stort mellanrum man får, beror av den använda basic-dialekten.

En del dialekter har en speciell variant av INPUT-instruktionen. Den har i sig inbyggd en möjlighet till text. Det innebär att man kan slopa PRINT-instruktionen för INPUT och få ett enklare program. I vårt fall skulle det se ut som följer:

```
10 INPUT "ANTAL
VAROR";A
20 INPUT "PRIS PER
STYCK (KR)";P
30 X=A*P
35 PRINT
40 PRINT "SUMMA
UTAN MOMS";X;"KR"
50 PRINT "SUMMA MED
MOMS";X*1.2063;"KR"
60 END
```

Vi befinner oss fortfarande på en beräkningsmässigt låg nivå, men vi kan ge våra beräkningar en god utskrift och vi kan kommunicera vettigt med datorn och vi kan, slutligen, bygga upp ett enkelt program.

Mycket typiskt för basic-språket är det interaktiva sättet att arbeta där man hela tiden har en "dialog" öppen med maskinen och bygger upp programmet mer och mer med

de erfarenheter man gör under provkörningarna och de ytterligare önskingar man kommer på att man har. Detta är en bra metod att arbeta med när man arbetar med små, överskådliga program. Stora komplicerade databehandlingar kräver att man planerar bättre och gör ordentliga flödesdiagram.

Vi slutar emellertid här och hoppas att den första inblick i basic som vi givit kan vara stimulerande för fortsatta praktiska övningar. ■

Tabell över några matematiska funktioner i basic.

Bokstaven X står här som symbol för en variabel, ett uttryck eller ett tal.

ABS(X) ger absolutbeloppet av X

ATN(X) ger arcus tangens av X

COS(X) ger cosinus för X

EXP(X) ger e^x

INT(X) ger heltalsdelen av X

LGT(X) ger 10-logaritmen av X.
Ej vanligt förekommande

LOG(X) ger naturliga logaritmen av X

SIN(X) ger sinus för X

SQR(X) ger kvadratroten av X

TAN(X) ger tangens för X

Vissa basic-dialekter saknar en del av dessa funktioner, och många har dessutom andra matematiska funktioner.

Färdig programvara för RT-datorn

AV TOMMY BLADH

När vi byggt färdig vår dator kommer vi inte långt utan systemprogram, dvs diskoperativsystem för flexenheten, tolkar för högnivåspråk, assemblerprogram m m. Lyckligtvis finns det en mängd färdig programvara att tillgå som framgår här.

■ Utan programvara är en dator lika livfull som fläskkotletten i snabbköpets frysdisk. Ett datorsystem skall inte bara bedömmas efter hur många kilobytes som kan adresseras, eller hur snabba instruktionerna är. Alla sådana uppgifter blir tämligen meningslösa om det inte går att få tag i högkvalitativ programvara. Visst finns det ett fåtal personer som är i stånd att skriva sitt eget diskoperativsystem. De flesta av oss har dock varken kunskap, tid eller tålamod för att gå i land med en sådan uppgift. I stället vill vi så snabbt som möjligt komma i gång med vårt system. Säkert finns det intressanta egna problem att stänga pannan blodig mot. Låt oss därför granska vad marknaden har att erbjuda oss.

TSC – Mjukvaruhuset

Den systemprogramvara som oftast används till 6800/6809 system kommer från TSC, *Technical Systems Consultants Inc* i USA. Deras operativsystem för flexskivesystem har i stort blivit standard för datorer med mikroprocessorerna 6800 och 6809. TSC sysslar inte med maskinvarukonstruktion, utan har specialiserat sig på programvara. Vi börjar därför genomgången med TSC:s programvara. Först granskar vi FLEX, diskoperativsystemet som är hjärtat, eller kanske snarare hjärnan i vår kära burk.

FLEX

Tidigare sålde SWTPc ett operativsystem med namnet mini-FLEX. Nu finns FLEX1, FLEX2 och FLEX9. FLEX1 arbetar mot de större 8" skivorna medan de andra två har 5 1/4" format. FLEX9 är avsedd för 6809 processorn men den har samma funktioner som FLEX2. FLEX finns även i versioner för andra system. Exempel är Motorolas *exorciser*, SWTPc, Smoke Signal Broadcasting, Percom m fl.

Uppgraderingen till version 2 från mini FLEX innebar bl a:

- 32k minne tillgängligt (48k för FLEX9).
- Printer spooling (utmatning av listor kan ske samtidigt med och oberoende av andra exekveringar).
- Random Access filer. Medger adressering direkt till önskad data.
- Uppdateringsmod. En fil kan öppnas för både skrivning och läsning samtidigt.
- Filer kan skyddas mot radering genom mjukvarumässig märkning.
- Filer kan datummärkas.
- Mera data per diskett genom större sektorer.

Hårdvaruförutsättningar

FLEX2 behöver RAM från 0000 och kontinuerligt uppåt. Minst 12k bytes krävs. Dessutom måste man ha 8k i området A000–BFFF. För FLEX9 ligger detta område högre upp: Mellan C000 och DFFF. I det 8k stora minnesområdet ligger FLEX resident.

Vidare krävs en monitor liknande SWTBUG. Det är främst in/ut tecken rutinerna som efterfrågas. MIKBUG kompatibilitet duger således. (TBUG) för att det skall fungera. I de fall man använder spooling behövs förutom timerhårdvara även interrupt vektorerna på rätt RAM-ställen.

Dokumentationen

När man köper FLEX får man en systemdiskett samt en omfattande bruksanvisning över medsänd systemprogramvara. Det skall på en

gång sägas att all dokumentation är ypperlig. Med mycken förklarande text och exempel beskrivs diskoperativsystemets ingående komponenter.

I den tjocka manualen (ca 200 sid) finns följande delar:

- *The FLEX disk operating system*. Innehåller allmänna förut-sättningar för att komma i gång. Kompatibilitet mot det äldre systemet m m.
- *FLEX User's Manual*. Beskriver de olika filtyperna, kommando hanteringen, utility rutinerna, felmeddelanden, FLEX i/o subrutiner, bootningen m m.
- *FLEX Editor*. Användarbeskrivning av texteditorn.
- *FLEX Assembler*. Användarbeskrivning av TSC Mnemonic Assembler för 6800.
- *FLEX Programmer's Manual*. Beskriver disk operativsystemet, filhanteringssystemet, diskett struktur, printer rutiner, interrupt i FLEX m m.

Operativsystemets beståndsdelar

FLEX operativsystemet består huvudsakligen av tre delar:

- ① The File Management System. Förkortat till FMS.
- ② The Disk Operating System. Även kallat DOS.
- ③ The Utility Command Set. Icke övåntat kallat UCS.

Kortfattat kan man säga att DOS processerar kommandon, FMS tar hand om och administrerar filer samt att UCS är användbara anropbara rutiner. Exempel på detta sista kan vara en **DELETE**-funktion.

Ett diskoperativsystem skall vara så konstruerat att det helt sköter hanteringen av dataflödet utan att användaren behöver bekymra sig om vart bitarna slutligen hamnar.

Filer i FLEX

Alla filer på disketten lagras i sektorer. Varje sektor består i version 2 av 256 bytes. Varje byte kan motsvaras av ett tecken i det fallet

att filen innehåller text. Det kan även vara en byte av maskin-information. Maximalt ryms 340 av användaren tillgängliga sektorer på en enkelsidig minidiskett. En dubbelsidig har dubbelt så många vilket systemet också håller reda på.

En fil är alltid minst en sektor lång men kan expanderas till så många sektorer som finns på skivan. Sektorerna som en fil är uppbyggd av behöver inte ligga i kontinuerlig ordning. Behövs mer än en sektor i en fil lägger FMS ut en adresslänk till den nya sektorn. På detta sätt kan en fil som skapas i ett läge då det uppstått luckor i sektorerna, bli utspridd över hela skivan. Luckor blir det normalt när man tar bort en fil. Detta märker inte användaren eftersom FLEX sköter detta automatiskt. Att en fil är utspridd resulterar dock i längre accesstider. Huvudet måste då fara längre över skivan vilket tar sin tid.

Filnamn och suffix

Alla filer har ett namn. Ex:

REGISTER
TIC-TAC
GLUFFS

Varje gång man refererar till en fil använder man sig av dess namn. Det kan röra sig om en editering, skapning eller borttagning. Maximalt åtta tecken, varav det första skall vara en bokstav, kan namnet bestå av.

En fil måste även ha ett s k suffix. Suffixet definierar typ av information som lagras i filen. Exempel är **TXT** för textfiler, **BIN** för maskinkodsfiler, **CMD** för utility kommando rutiner (hjälpprogram) och **BAS** för *basic* källprogram. Ett suffix kan bestå av upp till tre st tecken. Det första måste vara en bokstav.

De flesta av FLEX utilityprogrammen förutsätter ett standardvärde på suffixet. Det gör att man inte behöver speciellt ange ett sådant. Att editorn förutsätter en **TXT** fil är inte att undra över. Likaså att **SAVE** som flyttar minnesinnehåll till en diskfil vill ha suffixet **BIN**.

Exempel på fullständiga filnamn:

REGISTER.TXT

TIC-TAC.BAS

GLUFFS.BIN

Genom att suffix existerar och att hjälpprogrammen förutsätter vissa default (standardvärden), elimineras en massa tråkigheter. Ett program som man arbetar med kan således samtidigt existera som **TXT** källkod, **BAK** som är en sparad back-up kopia, **BIN** som assemblerat binärformat och **CMD** om det försetts med transferadress (att starta automatisk exekvering på).

Nummer på drivenhet

Förutom att en fil kan förses med namn och suffix för identifikation kan även ett nummer för flexskivenhet ges (disknummer). Det gör att en fil med samma namn och suffix kan existera samtidigt på olika flexskrivenheter.

Anges inget nummer, kommer disk nr 0 att anropas om man inte själv har definierat system- och arbetsstatus annorlunda. Numret kan ges antingen före eller efter filspeifikation. Exempel:

REGISTER.TXT.1

0.GLUFFS.BAS

Beteckningarna systemdisk och arbetsdisk behöver kanske några ords förklaring. Ett systemkommando med suffix **CMD** har som standard hemvist på disk nummer noll. Det betyder att systemet försöker hämta den kommandofilen från disk noll. En textfil som man arbetar med antas ligga på "arbetsdisken".

När systemet initieras den första gången får både system- och arbetsdisk en tilldelning till disk nr 0. Genom ett kommando kan arbetsstatusen tilldelas disk 1. Härigenom behöver man inte själv ange ett disknummer efter de filer man arbetar med. Allt som inte har suffixet **CMD** kommer då att sökas på disk nr 1.

Version 2 av FLEX har även begåvats med en annan finess: Automatisk sökning. Om systemet är satt i denna auto-mod, kommer en filsökning att börja på disk 0 och sedan automatiskt fortsätta på de övriga drivenheterna. Då det tar en

tid innan drivmotorn nått sitt stabila varvtal (den går ju bara vid accesser till skivan), är det förstås att föredra att rätt disk utpekas från början.

Inmatning av kommandon

FLEX startas upp i flera steg. Det första som skall ske är att man från sin monitor anropar en "boot-strap loader". Den är ett mycket litet program som kan ligga i PROM eller knappas in i RAM. (P)ROM-baserad inmatning är naturligtvis att föredra. Laddarens enda uppgift är att från ett förutbestämt ställe på skivan draga ned en större laddare i minnet. Den större laddaren tar sedan hand om uppstartningen av resten av operativsystemet. När allting så småningom är klart dyker tre prompttecken upp på skärmen (+++). Då vet man att FLEX är redo för användning.

Det man skriver från sitt tangentbord hamnar först i en inbuffert. Felaktiga nedslag kan därför korrigeras innan operativsystemet sätts att analysera vad man skrivit. Exekvering av en rad sker först efter tryckning av RETUR-knappen.

En kommandorad består vanligtvis av ett inledande kommando samt en lista av namn och parametrar. Flera än ett kommando kan skrivas på samma rad. Man separerar dem bara med ett ":" tecken. Gränsen är dock maximala 128 tecken i inbufferten. Exempel på kommandorader kan vara:

+++CAT 1

+++LIST REGISTER

Den första raden ger en utskrift över filkatalogen på drive nummer ett. Den andra raden ger en listning av textfilen REGISTER på bildskärmen.

Styrning mot radskrivare

I många fall önskar man få de av nyttoprogrammen producerade utskrifterna mot en skrivare i stället för på bildskärmen. Till hjälp för detta finns en speciell hjälpprogram (utility) som heter P. Den ser till att de utmatningar som skall till skrivaren hamnar där, samt att det

som fortfarande skall till skärmen också kommer dit. Felmeddelanden och liknande som fordrar operatörens uppmärksamhet styrs till skärmen. Dokumentationen är så utformad att man kan skriva en egen skrivarrutin som passar just den "häck" man kommit över biltigt på överskott. Exempel på ett kommando med radskrivarutmatning:

+++P LIST REGISTER

Finesser vid uppstartningen

När FLEX har laddats och är redo kanske man har önskemål om att vissa saker skall utföras automatiskt. Det kan t ex vara definition av systemparametrar, egenskaper hos ens bildskärm, utskriftsformat, specialtecken definitioner etc. Man kanske även önskar dra igång Basic-interpretatorn med en gång. Bara lugn; de har tänkt på den saken också!

Det första FLEX gör efter att ha initierat sig är att söka i diskens innehållsförteckning (directory) efter en fil med namnet **STARTUP.TXT**. Om filen inte kan hittas så kastar FLEX ut de tre promptarna och är redo för användarkommandon. Om däremot filen finns, kommer den att läsas och behandlas som en enstaka kommandorad. Vill man nödvändigtvis ha sin *basic* snurrande helt automatiskt, räcker det således att bygga upp en fil med namnet **STARTUP.TXT** som endast innehåller kommandot **BASIC**.

Det kan tyckas vara en begränsning att bara kunna fördefiniera ett kammando. Det verkar ju som om man inte både kunde anropa **TTYSET** för definition av systemparametrar och sedan dra igång *basic*. Ingen panik, det finns medicin mot detta med, nämligen en hjälprutin som heter **EXEC**. Rutinen gör ett anrop av en **TXT**-fil som kan innehålla hur många kommandon som helst. **EXEC** rutinen lurar operativsystemet att tro att de kommandon som det utför helt enkelt kommer från operatörsterminalen.

Vad man behöver göra är nu i stället att låta den enradiga **STARTUP.TXT** innehålla ett an-

CAT

CATALOG OF DRIVE NUMBER 0
DISK: #0

NAME	TYPE	SIZE	PRT
ERRORS	.SYS	9	
FLEX2	.SYS	25	
PRINT	.SYS	1	
CAT	.CMD	3	
LIST	.CMD	3	
COPY	.CMD	5	
TTYSET	.CMD	2	
RENAME	.CMD	1	
DELETE	.CMD	2	
ASN	.CMD	1	
APPEND	.CMD	3	
EDIT	.CMD	24	
ASMB	.CMD	25	
BUILD	.CMD	1	
P	.CMD	1	
JUMP	.CMD	1	
SAVE	.CMD	2	
PROT	.CMD	1	
EXEC	.CMD	1	
DATE	.CMD	2	
I	.CMD	1	
LINK	.CMD	1	
O	.CMD	2	
QCHECK	.CMD	3	
VERIFY	.CMD	1	
VERSION	.CMD	1	
XOUT	.CMD	2	
NEWDISK	.CMD	7	
PRINT	.CMD	2	
SAVE	.LOW	2	
FLEX	.COR	24	

SECTORS LEFT = 181

Fig 27:1. Katalogutskrift (bibliotek) från Flex2 systemskiva.

rop på **EXEC** med tillhörande fil.

Hjälprutiner i FLEX

Vid beställningen fick man som sagt tidigare en systemdisk. Den innehåller inte enbart diskoperativsystemet utan även en rad hjälprutiner, s k utility programs. I *fig 1* ser vi en utskrift av innehållsförteckningen från denna systemdisk. Utskriften är begärd med kommandot **CAT**. En liten kort genomgång av samtliga hjälprutiner kanske faller i smaken?

APPEND.CMD Kommandot nyttjas för att sammanfoga, konkatera, två eller flera filer. Alla typer av filer kan sammanfogas. Normalt är det bara vettigt att lägga samman filer av samma typ, exempelvis binärfiler. En ny fil skapas för resultatet och alla ingående filer förblir intakta.

ASN.CMD Kommandot används för att tilldela de tidigare nämnda "system"- och "arbets"-numren. Ger även utskrift av aktuell status.

BUILD.CMD Används för att snabbt bygga små textfiler. Exempelvis **STARTUP**-filen. Är en mycket anspråkslös editor för nya filer.

CAT.CMD Katalog kommando för utskrift av diskettens bibliotek. Ger den utskrift som finns i *fig 1*. Visar filens namn och suffix samt skyddsflaggor. De senare är "delete" resp "write protect" för filen.

COPY.CMD Används för kopiering av, antingen enstaka filer eller filer med vissa suffix. Vidare kan hela disketter kopieras m m. Om en fil haft sina ingående sektorer spridda, kommer datorn att försöka omgruppera dem för bättre sammanhållning.

DATE.CMD Används för att visa och/eller ändra **FLEX** interna datumregister.

DELETE.CMD Kommandot används för att plocka bort (delete) en fil från disketten. Rår inte på en

fil som man själv satt delete protect på.

EXEC.CMD Som vi sett tidigare används detta kommando för att starta kommandoströmmar. Hämtar upplagda kommandorader från en fil och exekverar dem i tur och ordning.

Get Detta är inte en hjälprutin utan en funktion i DOS. Kommandot ligger resident i det av DOS använda minnet. Med GET kan binärfiler laddas ned till minnet.

I.CMD Medger att man kan hämta inmatningstecken från en fil i stället för från terminalen. Kan användas i en STARTUP-rad som anropar en hjälprutin vilken ställer en fråga till operatören och vill ha svar innan den går vidare. Svaret, exempelvis "YES" kan då automatiskt hämtas från en diskfil.

JUMP.CMD Används för att starta exekvering av ett program som redan ligger i datorns minne. Man ger som enda parameter den hexadecimala adress där exekveringen skall börja.

LINK.CMD Används för att informera laddningsprogrammet var själva diskoperativsystemet ligger på skivan. Laddningsprogrammet är ju en "dum" rutin som bara vet att det skall läsa ned från en viss sektor. Flyttar man FLEX2.SYS genom kopiering e dyl måste en LINK-rutin utföras på den nya skivan.

LIST.CMD Används för listning av basic eller text-filer på terminal eller skrivare. Man kan även specificera listning mellan vissa rader eller från viss rad.

MON Detta kommando är i likhet med GET ingen hjälprutin. Det är ett resident kommando vars enda uppgift är att hoppa till monitorns kontrollslina. Med monitor avses ex.vis MIKBUG, TBUG eller dyl.

NEWDISK.CMD Sätter upp och formaterar en ny diskett. Läger

upp viss systeminformation samt kontrollerar att inga bristfälliga sektorer finns på skivan. Sektorer som ej går att använda märks som felaktiga. De kommer sedan att automatiskt uteslutas av operativsystemet. Har man otur kan förstås en sektor som systemet självt behöver vara felaktig.

CO.CMD Används för att överföra den information som skulle kunna ha presenterats på snabbskrivare, som ett resultat av ett resultat av ett hjälpprogram, till en fil i stället för till terminalen. Se vidare PRINT.CMD.

P.CMD Detta är systemets skrivarrutin. Ombesörjer utskrift på en skrivare av resultaten av kommandon.

PRINT.CMD Används för att mata ut tidigare lagrade utskrifter samtidigt som datorn utför andra uppgifter. Kräver hårdvara för att fungera, bl a en tidskrets (timer). Medger t ex att man spelar STARTREK samtidigt som skrivaren levererar 100 sidor.

PROT.CMD Används för att ändra och skriva ut eventuella "delete"- eller "write protect"-markeringar på en fil.

QCHECK.CMD Administrerar den kö av utskrifter som kan finnas då man utnyttjar printer spoolingen som FLEX tillåter.

RENAME.CMD Används för om döpnings av filer och suffix.

SAVE.CMD Flyttar minnesinnehåll upp på en diskfil. Man specificerar filnamn, start/stopp adress samt en sk transferadress (option). Transferadressen används för automatisk exekveringsstart av ett program. Alla kommandorutiner har en sådan.

SAVE.LOW Samma som SAVE.CMD utom att den ligger på lägre minnesadress. Medger undanlagring av program i själva arean för hjälprutiner.

STARTUP.TXT Detta är inte en hjälprutin utan en finess i FLEX. Tidigare har vi beskrivit hur FLEX, efter initieringen söker i biblioteket efter en fil med detta namn. Finns den så kommer dess enda tillåtna rad att tas för ett kommando och utföras.

TTYSET.CMD Med detta kommando kan man sätta en rad parametrar som kontrollerar egenskaperna hos terminalen, t ex att specificera antalet rader/sida, tecken/rad, delete tecken, antal nulls vid CR m m. Rutinen bör inkallas automatiskt genom en STARTUP.

VERIFY.CMD Används för att visa och/eller ändra verify flaggan. Med flaggan satt kommer alla de sektorer som skrivs till skivan att automatiskt läsas och jämföras med indata.

VERSION.CMD Skriver ut versionsnamn för ett program. Fordrar att man föjer en programmeringsregel och i tredje byten i programmet lägger ett versionsnummer.

XOUT.CMD Tar bort de filer av typ OUT som innehåller lagrade utskrifter med undantag av skyddade filer.

Överigt i katalogen

I listen i *fig 1* finns ytterligare några filer. Dessa är ej utility program.

ERRORS.SYS Innehåller felmeddelanden av förutbestämd längd. Filen är av typ "random access". Det gör att en systemrutin som vill ha en felutskrift gjord och har ett felnummer i handen, kan beräkna en sektoradress och direkt gå in och hämta en textsträng. Rutinen sparar primärminnesutrymme och medger texter på olika språk som enkel förlängning.

FLEX2.SYS Diskoperativsystemet. Komplet med I/O rutiner. Finns även i versionen **FLEX.COR** utan dessa rutiner. Dokumentatio-

nen visar hur man för ett eget system skräddarsyr sådana om de som finns inte passar.

Egna program som nyttjar FLEX

Medföljande *FLEX Programmer's Manual* innehåller all dokumentation för att man skall kunna skriva egna program som använder de möjligheter FLEX ger.

Först finns en förteckning med de minnesceller inom DOS-delen av FLEX vilka kan vara intressanta. Det är bl a alla parametrar som styr utmatnings- och inmatningsformaten. Vidare de pekare associerade med hanteringen av kommandobuffert, transferadress, feltyp och en massa andra dataceller som skulle ta för lång tid att räkna upp.

Sedan kommer en lista över de av användare anropbara rutinerna i systemet. Det är bl a kalla och varma startpunkter i DOS, teckenrutiner av alla slag för in/utmatning mot terminal och radbuffert, rutin för att extrahera filspezifikationer ur inbufferten och och lagra dessa i ett FCB, filkontrollblock. Dessutom rutin för laddning av binärfil och felmeddelande rapportering, talkonvertering m m. Vidare finns möjlighet att anropa DOS som en subrutin.

Hur man lägger upp egna kommandon visas i manualens efterföljande stycke. De kan vara både minnesresidenta och diskresidenta. Variabel längd tillåts.

Filhanteringssystemet, FMS

Filhanteringssystemet är en länk mellan DOS och hårdvaran för diskarna. FMS utför all allokering av filerna på disketten, dvs bestämmer var filerna skall ligga. Likaså tar den bort filer och frigör använt utrymme. Det sköts dynamiskt vilket gör att man som användare inte behöver bry sig om var en fil rent fysiskt ligger.

Kommunikationen med FMS sker genom filkontrollblock som betecknas FCB. Dessa innehåller nödvändig information om filen. In- och utmatningen på disken ut-

för filhanteringssystemet med ett tecken i taget. Ett program behöver därför bara sända eller begära ett tecken i taget. Likheten med arbete mot en vanlig terminal är därmed slående.

Filhanteringssystemet FMS är ett kommandospråk som styrs med hjälp av olika funktionskoder. Varje kod informerar filhanteringssystemet vad den skall göra. Det kan vara att öppna en fil för läsning eller att ta bort en fil. I praktiken är det rätt enkelt. Indexregistret får peka på det filkontrollblock som skall användas. Funktionskoden lagras i första ordet i filkontrollblocket och sedan anropas filhanteringssystemet som en subrutin. Efteråt kan man testa på en returkod för att se om allt gick rätt. Man behöver således aldrig bekymra sig om vart all information hämtas. Det utför filhanteringssystemet helt på egen hand.

Ett obestämt antal filer kan hållas i luften samtidigt. Det krävs bara att varje fil har ett eget filkontrollblock.

Funktionskoder

Koden består av en byte som lagras först i ett filkontrollblock. För närvarande finns 20 st använda. Att se dem listade ger bra grepp om vad man kan göra:

- 00 Read/Write Next Byte/Char
- 01 Open for Read.
- 02 Open for Write.
- 03 Open for Update.
- 04 Close File.
- 05 Rewind File.
- 06 Open Directory.
- 07 Get Information Record.
- 08 Put Information Record.
- 09 Read Single Sector.
- 10 Write Single Sector.
- 11 -
- 12 Delete File.
- 13 Rename File.
- 14 -
- 15 Next Sequential Sector.
- 16 Open System Information Rec.
- 17 Get Random Byte from Sector.
- 18 Put Random Byte in Sector.
- 19 -
- 20 Find Next Drive.
- 21 Position to Record N.
- 22 Backup One Record.

Filer: Random, Binär, Text

Flex 2.0 medger bruk av *random access* filer. Med detta menas att man kan nå information på ett godtyckligt ställe i filen utan att behöva läsa den från början i sekvensiell mod. För att få denna finess måste man offra utrymme. En random fil måste skapas slumpmässigt. Det är en egenskap som ligger i filens struktur. Systemet skapar en *File Sector Map* dvs en minnestabla, som sammanförs med filen. Minnestabla har alltid längden två sektorer. I minnestabla kommer nu systemet att lagra den fysiska adressen för varje logisk sektor.

Med logisk sektor skall förstås ett ordningsnummer. En fil består ju av flera sektorer som kan vara utspridda på skivan. De logiska numren är dock alltid desamma, t ex 0, 1, 2 . . n. Genom att man vet antalet användbara bytes inom en sektor (252) och längden på den informationsenhet man processar, kan den önskade logiska sektorns nummer framräknas. Att få fram rätt information kräver sedan två filaccesser.

En *binärfil* kan innehålla allt som data. En binärfil består av en eller flera inspelade binära poster. En sådan post kan se ut som följer:

- | | |
|----------|--|
| Byte 0 | Indikering av inspelningens start (\$02, ASCII STX). |
| Byte 1 | Mest signifikant byte i laddadressen. |
| Byte 2 | Minst signifikant byte i laddadressen. |
| Byte 3 | Antal bytes data i rec. |
| Byte 4-n | Binära data bytes. |

En binärfil kan även innehålla ett "transfer adress record". Det används för att man i en förbestämd adress skall kunna starta en automatisk exekvering. Format:

- | | |
|--------|---|
| Byte 0 | Transfer address indikator (\$16, ASCII ACK). |
| Byte 1 | Mest signifikant byte i transferadressen. |
| Byte 2 | Minst signifikant byte i transferadressen. |

En *textfil* innehåller bara de ASCII-tecken som kan skrivas på skärm eller papper. Den kan även innehålla specialtecken:

\$0D (ASCII retur) är ett tecken som markerar radslut.

\$09 (ASCII HT) är en flagga som säger att en sträng av mellanslag har tagits bort och komprimerats i filen. Det ord som följer omedelbart efter flaggan anger antalet borttagna blanka tecken (2-127). När man har öppnat en text fil för läsning eller skrivning, kommer filhanteringssystemet att på egen hand lägga till resp ta bort dessa blanka. Detta märker man inget av.

Summering

Att ta fram ett diskoperativsystem är förmodligen ett ganska omfattande, svårt och tidsödande arbete. Vad som framkommit vid körning av FLEX, studier av manual och erfarenheter av andra system förstärker intrycket att FLEX är ett mycket kraftfullt diskoperativsystem.

Text Processing System

är ett kraftfullt ordbehandlingsprogram med egenskaper som man vanligtvis bara finner i större installationer. Låt oss nämna några egenskaper: Justering av vänstermarginal, enbart justering av högermarginal, justering av vänster och höger marginal, valbart radmellanrum, indrag, sidkontroll, centrerung, numrering av sidor, rubriker, kontroll av rad och sidlängd m m. Vidare medger det sammanställning av flera kommandon till s k makrofunktioner, numeriska register, villkorlig kommandoexekvering, omflyttning av text, återupprepning, interaktivitet mot terminaloperatören etc. Valbara exekveringspunkter för makron tillåter en ändlös variation av ordbehandling såsom fotnoter och formulärbrev. Externa datafiler kan innehålla exempelvis namn och adressregister för standardbrev. Textprocessorn arbetar på en textfil som tidigare gjorts upp med editorn.

Basic

TSC:s flyttalsbasic är en av de snabbaste tolkarna för 8-bit mikrodatorer. Den är helt orienterad mot

FLEX med sekventiell och direktadresserad (random access) filhantering. Det senare kan vara antingen poster för in/utfunktioner eller virtuella fält. Matematiska operationer utförs binärt med 6 siffrors noggrannhet vid utskrift och det interna dynamiska området är 10 upphöjt till $+ - 37$. Alla standardiserade Basicoperationer ingår liksom finesser som dynamiska obegränsade strängvariabler, **HEXadecimal** funktion, **IF/THEN/ELSE** struktur, **ON ERROR GOTO**, **TRACE** och tvådimensionella fält. Tolken kräver 16k arbetsminne.

Extended Basic

Den här tolken innehåller alla egenskaper från TSC:s standardbasic. Dessutom har den funktioner som gör den lämplig i affärsrättsliga tillämpningar. Den har 17 siffrors noggrannhet vid matematiska operationer. En efterlängtat variabeltyp är integer variabeln, dvs heltalsaritmetik. Alla pekare, index etc som består av hela tal kan anges som integer. Det ger extremt snabb exekveringstid jämfört med flyttalsberäkningar. För att enklare få elegant formattering av utskrifter finns nu **PRINT USING**. Där finns **INSTR** för sökning i substrängar, **DATE\$** ger datum som en färdig sträng, **SWAP** snabbar upp sorteringar genom att flytta om strängarnas pekare i stället för alla tecknen. Datorns minne kan nu direkt manipuleras även på 16 bit nivå med **DPEEK** och **DPOKE**. Med **INCH\$** kan enstaka tecken inmatas från valfri kanal (terminal, fil etc). Trots sina många nya egenskaper och utökade noggrannhet är Extended Basic en av de snabbare tolkarna. Den är perfekt för seriös affärsrättslig och/eller vetenskapligt bruk. Programmet behöver 24k RAM för att köras.

Sort/Merge

Med detta professionella sorteringspaket kan innehållet i valfri typ av fil, med obegränsad storlek, sorteras enligt givna parametrar. Dessa kan ges via kommandoraden, via en interaktiv parametereditor eller via en existerande pa-

rameterfil. Paketet är skrivet i assembler och är extremt snabbt. Filer som är för stora för att rymmas i primärminnet bryts upp i flera temporära arbetsfiler. De sorteras individuellt och slås därefter helt automatiskt samman till en enda fil igen. Programmet accepterar records med både fix och variabel längd. Upp till 20 stigande eller fallande sorteringsnycklar kan ges. Vidare finns icke ASCII sorterande sekvenser och mycket mera.

Debug package

är ett extremt kraftfullt verktyg för testning och felsökning i program på assembler nivå. Paketet är kapabelt att stimulera alla funktioner hos mikroprocessorn, inklusive avbrott och I/O. Det medger multipla villkorliga brytpunkter, histogram för profilering av programavverkningen, spårning, stegning av instruktioner, spårning bakåt för att man skall kunna se de sista 256 instruktionerna, minnesskydd i block (skrivning, exekvering, minnes referens, eller simulerings-skydd), inbyggd miniassembler och disassembler, kommandon för undersökning av minne, hex kalkylator, maskinstatus-räknare, stackskydd och mycket mer. Kräver 9k RAM från \$3C00.

Additional FLEX Utilities

är ett stort urval av hjälpprogram för flex-skive systemet. Inkluderande minnesdump, skriv diagnostik/testning, utökad filkatalog utskrift, analys av segment i object fil, fil dumpare m m. Innehåller även källtexten till dessa program.

Basic Precompiler

Detta program finns i olika versioner för standard och extended basic. Med hjälp av det kan man skriva basicprogram i ett okonventionellt format. Variablers namn kan vara av obegränsat antal tecken. Likaså kan programlägen bestå av alfanumeriska tecken. De intetsäggande radnumren kan alltså utgå eftersom de inte behövs. Kommentarrader kan sättas in var som helst. Det här gör att basicprogrammet blir mer lättläst och enk-

lare att förstå. Resultatet blir det "kompilerade" internformatet hos TSC's Basic.

68000 Cross assembler

Det här är en full assembler för den nya 16 bitars mikroprocessorn 68000. Assemblern körs på ett 6800/09 system och producerar kod för 68000. Den understödjer makron, villkorlig assemblering, full 32 bits aritmetik samt flera moderna byggstenar för strukturerad assemblerkod. Källkodssyntaxen är kompatibel med Motorolas definierade 68000 syntax.

Övrig mjukvara

Även andra leverantörer än TSC finns naturligtvis. *Software Dynamics* har både editor/assembler samt en kompillerande Basic som går under FLEX. Basic:en har både heltal och flyttal. Diskhantering finns också. Satserna är i sin mångfald liknande dem som finns i TSC:s extended basic. Det kompillerade programmet kräver ett s k run-time-paket.

En basiskompilator finns också från *Microware Systems Corp*, både i kassett- och FLEX-version. Den har ej flyttalsaritmetik utan endast 16 bits heltalsaritmetik och inget "run-time-paket". Kompilatorn adderar själv till nödvändiga systemrutiner från ett internt bibliotek.

Uppräkningen ovan kan naturligtvis inte bli fullständig. Ögnar man igenom specialtidsskriften för 6800/09, 68 *Micro Journal*, finner man förvånansvärt många annonser. De flesta omtalade språken verkar ha åtminstone en försäljare. Man hittar **Pascal, Forth, Lisp, Fortran** etc. Se kapitlet högnivåspråk. Vi har vidare utelämnat massor av specialprogram som exempelvis ASCII/Morse/ASCII konverterare för telegrafi, lager- och bokföringsprogram för den lilla firman, spelprogram i mängder m m. Den intresserade bör ta kontakt med de som säljer 6800/09 system eller studera tidskrifter i ämnet. Personliga kontakter i en privatdataklubb kan också innebära många tips. ■

Olika högnivåspråk för smådatorer

AV GUNNAR LILLIESKÖLD

Avancerad programmering kräver högnivåspråk eftersom assembler- och maskinspråksprogrammering tar alltför lång tid att genomföra.

För datoramatörer är basic ett lämpligt högnivåspråk, men även andra språk kan vara lämpliga att använda i vissa sammanhang. Basic var också det första högnivåspråket för mikrodatorer, men allt fler finns att välja mellan.

■ Små och medelstora program kan man med fördel skriva i assembler. Det leder till program som på ett effektivare sätt utnyttjar maskinens minnesutrymme. Dessutom ger det som regel snabbast möjliga program. I assemblerspråket (eller om man arbetar direkt i maskinkod) har man en bra kontroll över hur lång tid varje moment tar.

När man skall göra större program är det bättre att satsa på ett högnivåspråk. Man kan då få enklare programmeringsförfarande och kortare tid för programutvecklingen.

Mikroprocessorerna var från början inte avsedda för högnivåspråk. Jämfört med en minidator är de ganska primitiva. Instruktionerna är inte lika kraftfulla och adresseringsutrymmet är mindre. Senare generationens mikroprocessorer är dock på ett annat sätt anpassade till högnivåspråk än de första mikroprocessorerna. Särskilt gäller det 16-bitarsprocessorerna. Faktiskt har man trots mikrodatorns begränsningar lyckats implemente-

ra ett flertal högnivåspråk på dem. Mikroprocessorernas begränsningar gör att det är relativt svårt att skriva kompilatorer för dem, medan däremot interpretatorerna lämpar sig väl. De senare tolkar högnivåspråket direkt. För en kompilator skriver man först ett *källprogram*. Det kompileras och man får som resultat en *objektkod*, dvs maskinspråk. Ett interpretativt språk upplevs av de flesta som lättare att arbeta med eftersom man direkt ser resultatet av det man skriver in. Gör man fel brukar datorn svara med ett felmeddelande. Interpretativa språk är som regel *basic*, *APL* och *Lisp* medan *fortran*, *algol*, *cobol* och *PL/I* med dess varianter för mikrodatorer (*PL/M*, *MPL*, *PL/65*, *PL/Z* m fl) normalt är kompillerande språk.

När högnivåspråken introducerades . . .

Låt oss se när de olika högnivåspråken introducerades:

<i>Fortran</i>	1957
<i>Cobol</i>	1960
<i>Algol</i>	1960
<i>Lisp</i>	1961
<i>Snobol</i>	1962
<i>Basic</i>	1965
<i>PL/I</i>	1965
<i>APL</i>	1967
<i>Pascal</i>	1971

Årtalen avser alltså introduktionen av en specificerad kompilator. Det skedde i många fall upp till 5 år efter det att den första försöksversionen specificerades. Språken har ursprungligen tagits fram för stor- och minidatorer. Under hand har flertalet av dessa dykt upp i versioner avsedda för olika mikrodatorer. Andra högnivåspråk som förekommer i mikrodatorsammanhang är: *Forth* och dess variant *urth*, *pilot*, *smalltalk*, *mumps*, *SAM 76*, *PL/M* (och varianterna *PL/Z*, *PL/65*, *MPL* m fl), *GPM*, *trac* och många fler. Varje språk har sina fördelar och sina begränsningar. Som nybörjare bör man välja ett allmänt accepterat och lättillgängligt språk som t ex *basic*.

● **Basic** blev det första högnivåspråket för mikrodatorer. Programspråket medger interpretato-

rer som översättare och därför var det relativt enkelt att implementera språket på de processorer som fanns i mitten av 70-talet då smådatorutvecklingen började. Språket blev populärt inte minst för att det är så enkelt att lära sig. Interpretatorn behövde inte heller vara så särskilt stor. Det finns basic-interpretatorer som bara tar upp 4 kbyte minnesarea (i ROM eller RAM) men de är å andra sidan ganska begränsade (kallas vanligen tiny basic). Normalt brukar en basic-interpretator vara 8 k byte stor men det förekommer interpretatorer upp till 24 kbyte.

En interpretator passar bra bara för en typ av processor. För en annan processor måste hela interpretatorprogrammet skrivas om. För att lösa det problemet har man börjat skriva översättaren i språket självt (!) för vissa högnivåspråk. Det gäller t ex för språket *pascal*.

• **Pascal**, som fått sitt namn efter matematikern *Blaise Pascal*, är ett språk på frammarsch. De flesta mikrodatorverkare har idag pascal för sina processorer. Med en s k p-kompilator överförs pascal till p-kod (pseudokod) som är universell för alla maskiner. Sedan behövs en enkel interpretator som översätter från p-kod till processorns maskinspråk. Mikrodatorn är här stack-orienterad vilket förklarar interpretatorn. Man är ju här inte lika beroende av vad den specifika mikroprocessorn har i form av register eftersom man i stället sparar information om ad-

resser o dyl under programexekveringen i stacken, dvs en viss del av RAM-arean.

Pascal innehåller satser som är avpassade för strukturerad programmering. Till skillnad från radorienterade språk som basic kan man i pascal lämna utrymme för luft i programtexten, tabuleringar m m. En stor fördel med *pascal* är att man lätt kan felsöka i programmet (debugging). Men först måste man förstås ha kunnat genomföra en kompilering som då innebär att programmet språkligt sett är riktigt formulerat.

• **Algol** (Algoritmisk Language) är ett betydligt äldre språk än pascal. De påminner om varandra. *Algol* är relativt ovanligt i mikrodatorsammanhang medan det förekommer ofta i stordatorer. Det ursprungliga syftet med *algol* var att åstadkomma ett universalspråk som lämpade sig för logisk och effektiv programmering av algoritmer. Språket som togs fram i Europa håller på att retirera till förmån för pascal. Det introducerade sådana finesser som rekursiva anrop (rutiner som kan anropa sig själv) blockstrukturering och dynamisk minnesuppdelning. Originaldefinitionen innefattade ej in- och utkommandon vilket hör till språkets svagheter.

• **Fortran** (*FORMula TRANslation*) fanns långt före basic. *Basic* gjordes faktiskt med utgångspunkt i *fortran* och likheterna är många. Tidigare var *fortran* mycket kraftfullare och lämpar sig särskilt väl i

beräkningssammanhang men det saknar stränghantering av olika slag som finns med i moderna *basic*-tolkar. På tekniska högskolor är det förmodligen fortfarande det mest använda språket idag, vilket många beklagar. Det är ett komplicerande språk till skillnad från *basic* som arbetar interpretativt (Men det finns faktiskt kraftfulla *basic*-dialekter som arbetar med kompilator). Den vanligaste versionen är *fortran IV*.

Om vi jämför *fortran* med *basic* finner vi att *fortran*programmen exekveras snabbare men att programmodifieringarna tar längre tid att utföra. I *fortran* kan vi ha en eller två bytes långa heltal, fyra eller åtta byte långa flyttal med upp till sexton siffrors noggrannhet. De flesta *basic*-tolkar har bara flyttalsaritmetik men det förekommer även heltalsaritmetik vilket ger snabbare programexekveringar. Medan flertalet *basic*-dialekter bara tillåter decimala värden och ASCII-kod kan man i *fortran* var som helst i programmet lägga binära, oktala, decimala och hexadecimala tal samt ASCII-tecken.

• **Cobol** är ett språk som är framtaget för administrativ databehandling (*Common Business Oriented Language*). Främst används det i stordatorer men finns även för minidatorer och en del mikrodatorer. Dess främsta fördel är att det troligen är det mest använda och standardiserade datorspråket av alla förekommande. Språket är anpassat för datahantering i databaser, register m m.

• **Lisp** har framför allt fått användning i behandling av listor. Ser man på program skrivna i *lisp* finner man en rad parenteser. Någon har hävdat att *lisp* är en förkortning av *Lots och Infernal Stupid Parentheses* vilket kanske säger en del . . . Språket består av en extern syntax kallad *M-lisp* och en intern syntax kallad *S-lisp*. Den senare kan sägas vara språkets assemblerdel.

• **Forth** uppfanns av en man vid namn *Charles Moore*. Det är ett kompilerande språk, men det består inte bara av en kompilator utan även av en interpretator, mac-

```
*p u n a c ÷ ω ↑ € | ^ \ > □ ; [ ° ∇ ' Δ
.._ < ≠ ≤ ≥ = > ) ∇ : ⊥ _ ( ↓ T → 1 ~ 0 ? [ -
123456789]9.BF[UN←ITOQD+
PRVCAZ×WYEMO/XL,SJGKH
```

Fig 28:1. Så här ser tecknen ut i APL-språket. Det kan ta sin tid att lära teckenrepertoaren och språket, men lönen för mödan blir extremt korta källprogram.

roassembler, operativsystem och mycket mer. Kompilatorn alstrar en *p*-kod, precis som i *pascal*. Det mesta i *forth*-systemet är skrivet i *forth*. För att flytta det från en mikroprocessor till en annan behöver man bara göra smärre ändringar, i storleksordningen 1 k byte. Språket är mycket flexibelt tack vare att man själv kan skapa makroinstruktioner i det. De kan skrivas i assemblerkod eller i ett annat högnivåspråk. En variant av språket heter *urth*.

- **APL** brukar betecknas som ett vetenskapligt eller matematiskt språk men har också funnit andra tillämpningar. Det är interpreterande därför interaktivt. Språket har mycket lustiga symboler. Se *fig 28:1*. Det är utvecklat av IBM. Språket är uppbyggt av matriser och notationen, kallad *Iverson*-notation efter språkets upphovsman, är speciell. APL bjuder på extremt korta källprogram, jämfört med andra högnivåspråk och det används allt flitigare. Däremot brukar objekt-koden bli lång och APL kräver relativt stora minnen.

- **PL/M**. Namnet är inregistrerat på Intel som tog fram språket år 1973 med utgångspunkt i stordatorspråket PL/1. Andra firmors varianter heter t ex MPL (*Motorola*), PL/65 (*Rockwell*) m fl. PL/Z från *Zilog* är dock utvecklat ur *pascal*. Språken har ungefär samma användningsområde som assembler med speciell inriktning på styrtillämpningar, men det ger kortare och enklare program som i sin uppställning mycket liknar *pascal*.

- **ADA** är ett försök av amerikanska försvaret att skapa ett språk som skall vara generellt användbart. Tidigare har man arbetat med hundratals olika språk, men år 1975 bestämde man sig för att satsa på det som skulle bli ADA. Språket togs fram av ett europeiskt team. Det är baserat på *pascal*.

- **Pilot** är utvecklat med tanke på undervisningsändamål. Det hjälper lärarna att utveckla frågor och svar och har bl a grafiska möjligheter. Det är även tillämpligt i CAD-sammanhang dvs vid konstruktionsarbeten utförda av datorer. ■

Ordlista

A

Akkumulator. Ett eller flera register som tillfälligt lagrar ett resultat i mikroprocessorn.

ACIA. Asynchronous Communications Interface Adapter. En krets i 6800 systemet som omvandlar 8 parallella bitar från datorn till serie. Kretsen kan arbeta både som sändare och mottagare.

A/D-omvandling. Omvandlar analoga signaler till digitala.

Adressbuss. Ett antal parallella ledningar som binärt indikerar den aktuella adress som datorn arbetar med.

Adressregister. Datorregister för lagring av en adress.

Algol. Se kapitlet högnivåspråk.

ALU. Aritmetisk logisk enhet.

ANSI. Förkortning av American National Standards Institute. En amerikansk organisation för industristandard.

Arbetsminne. Den del av primärminnet som kan användas för tillfällig lagring av data.

ASCII. En ANSI-standard för tecken. Överföring sker med 7 bitar mellan terminal och dator och till andra kringenheter. Den kompletta ASCII-koden har 128 tecken. En svensk variant förekommer med å, ä och ö i stället för], [och /. Kodtabellen finns angiven i bokens appendix.

Assembler. Ett programspråk där man i stället för maskinkod skriver memosymboler (mnemonics) för datorns instruktioner. Språket anses ligga över maskinspråk men under de s k högnivåspråken.

Asynkron överföring. Den är vanligen seriell och överföringen klockas ej synkront med andra signaler (t ex med datorns klocka). I stället innehåller orden i sig information som talar om när nästa ord skall registreras av mottagarsidan.

Avbrott eller eng *interrupt*. Rutin som innebär att datorprogrammet tillfälligt avstannar p g a en yttre händelse.

B

Basic. (*Beginners All-purpose Symbolic Instruction Code*). Lägst i rangordningen bland högnivåspråk. Enkelt för nybörjare och internationellt accepterat. Språket förklaras närmare i boken.

Baud. Ett mått på hastigheten vid dataöverföring. 300 baud betyder 300 bits/sek.

Baudot-koden. En fembitarskod som används vid RTTY, dvs radio-teletype.

BCD-kod. Förkortning av binärkodad decimal. Se kapitlet talsystem.

Binära tal. Talsystem med 2 som bas. Representeras av siffrorna 0 och 1. Används i datorer och annan digital utrustning.

Bit. En siffra i det binära systemet.

Bit-rate = Baud.

Boolesk algebra. Uppfunnen av fransmannen George Boole. Regler för logiska uttryck som ofta tillämpas i digitaltekniken.

Buss. Parallella ledare som förbinder flera enheter.

Byte. Omfattar digitalt ord om 8 bitar. Se även nibble.

C

Cobol. Ett probleminriktat högnivåspråk för administrativa tillämpningar.

CPU. Central Processing Unit, dvs datorns processor.

CRT. Cathode ray tube. Betyder egentligen katodstrålerör, men används i datorsammanhang för att beteckna en bildskärmsterminal.

Cursor. En markör på bildskärmen som visar var nästa tecken skrivs.

Cykeltid. Minsta tidintervall för på varandra följande skriv- eller läs-förlopp i ett minne.

D

D/A-omvandlare. Digital /analog-omvandlare.

Daisy wheel printer. En skrivare med typhjul.

Databas. Den totala massan tillgängliga data som en dator kan använda för sina bearbetningar och utskrifter.

Databuss. Ett buss-system för da-

torns olika enheter för dataöverföring.

Datorord. Ord som är lagrat i minnescell i en dator och där kan behandlas som en enhet. Se byte och nibble.

Debug. På svenska "avlusa", vilket innebär felsökning i program och borttagande av felen.

Decoder. Avkodare. Omvandlar en kod till en annan.

Diskett. Annat namn för flexskiva eller floppy disc.

DMA. Direkt minnesåtkomst. Innebär att man direkt kan nå ett minne utan att datorns processor är inblandad. Det kan ske samtidigt som annan verksamhet pågår kring datorns processor.

DOS. Disc Operating System. Ett program som sköter om rutinerna för flexskiveenheten eller skivminnet så att man kan hämta och lämna data, söka m m.

Dynamiskt minne. Ett minne, vanligen kapacitivt, som ständigt måste återladdas (refresh) för att inte tappa informationen. Fördelarna jämfört med statiska minnen är billigare pris/minnescell, ofta snabbare och mindre effektutveckling. Till nackdelarna hör att "refresh"-kretsar måste finnas och en omständligare felsökningsprocedur än vad de statiska minnena kräver.

E

EBCDIC. En förkortning av Extended Binary Coded Decimal Interchange. Används av IBM i stället för ASCII-kod.

Editor. Ett program som används för redigering av de program man själv arbetar med.

Emulator. En funktionsenhet som består av maskin- och programvara. Med dess hjälp kan man i ett datorsystem exekvera datorprogram som är avsedda för ett annat datorsystem.

EPROM. Ett programmerbart läsminne som även kan raderas och återprogrammeras. Raderingen sker med uv-ljus till skillnad från EAROM som kan raderas elektriskt. PROM kan enbart programmeras.

F

FET. Fälteffekttransistor.

Fil. En stor kvantitet sammanhørende data som behandlas som en enhet.

Flexskiva. Magnetskiva för lagring av data (Floppy disc, diskette). Finns i storlekarna 5 1/4" och 8". Skivorna är indelade i sektorer enligt principen hård- eller mjuksektorering.

Flyttalsaritmetik. Decimalkommat tillåts "flyta", dvs kan skiftas vänster eller höger vid multiplikation resp division.

Fortran. Formula translator. Probleminriktat högnivåspråk för matematiska beräkningar.

Fusible link. En teknik som används i PROM. Läsminnet programmeras genom att de inbyggda "länkarna" för de aktuella minnescellerna bränns av. Därför kan ett PROM inte raderas och återprogrammeras.

Fält. Avgränsat område som används för en speciell kategori data.

G

Generell dator. En dator som kan behandla all slags problem inom dess begränsningar av tid, hastighet och minnesstorlek.

H

Halvduplexöverföring. Dataöverföring över dataledningar i två riktningar, dock ej samtidigt.

Hexadecimal. Talsystem med 16 som bas. Se kapitlet talsystem.

Hållremsa. En remsa i vilken hål är instansade. Hålens lägen är standardiserade och deras sammansättning svara mot en kod.

Högnivåspråk. Ett problemorienterat språk som inte är avpassat till viss dator eller viss klass av datorer.

I

Index. Något som kan tjäna som pekare eller indikator, t ex innehållet i processorns indexregister.

Indexerad adress. En adress som datorn automatiskt räknar ut med

utgångspunkt i innehållet i indexregistret.

Indirekt adressering. En metod för adressering där man låter en minnescell innehålla adress-koden för en instruktion eller ett dataord.

Instruktion. Ett uttryck som specificerar vad datorn skall utföra.

Intelligent terminal. En terminalenhet, t ex en bildskärm, som i sig kan utföra en process utan att ta datorn till hjälp.

Interaktiv rutin. En dialog mellan dator människa: Datorn väntar på inmatning från mannen vid terminalen, går vidare i programmet och gör nya uppehåll för inhämtning av data.

Interface. Anpassning säger vi på svenska. Man menar här en enhet som anpassar signaler och ev koder mellan två enheter.

Interpretator. En översättare, eller tolk, från ett språk till datorns maskinspråk. Till skillnad från en kompilator arbetar interpretatorn direkt i ett steg.

Interrupt. Avbrott i programmet. Används t ex vid asynkrona signaler från t ex ett tangentbord kräver svar eller aktion från datorn.

Interrupt request (IRQ). En typ av programavbrott som datorn gör efter "önskemål" från en periferienhet. Ett tangentbord sänder t ex IRQ varje gång man trycker ned en tangent för att visa för datorn att data finns att hämta.

I/O. In- /utenhet.

J

Jämn paritet. Vid paritetskontroll skall antalet bitar som kontrolleras, inkl paritetsbit, med värdet 1 vara jämnt.

K

Kilobit. Ettusen bits vid litet k. Betecknar 1024 bits (2^{10}) vid stort K.

Kilobyte. Ettusen bites. Betecknar 1024 bytes (2^{10}) vid stort K.

Klocka. En pulsgenerator som ger periodiska signaler för att synkronisera signaler vid t ex överföring av pulståg eller att tidsmässigt påverka olika funktioner i en dator.

Komparator. Kallas även jämföra-

re. Den jämför två signaler och ger ett resultat som indikerar om signalerna är lika eller ej.

Kompilator. Omvandlar källspråk till objektкод, dvs det problemriktade högnivåspråket till maskinspråk.

Komplement. Se avsnittet talsystem.

Kärnminne. Ett minne där varje minnescell består av en ringformig magnetiserbar kärna.

L

Länkare. Kan sammanlänka var för sig kompillerade programmoduler.

Läspenna. Läser optiskt av program som kodats grafiskt.

M

Maskinkod. Den kod eller det språk processorn arbetar med.

Makroinstruktioner. En makroinstruktion innehåller en sekvens av flera mikroinstruktioner.

Makroassembler. En assembler med makroinstruktioner.

Mikroinstruktion. Instruktion i mikroprogram.

Minne. Enhet där data kan placeras för att lagras och sedan hämtas.

Mnemonic, eller på svenska memosymboler, är en symbol som stöd för minnet. I datorer används de för att beteckna datorinstruktioner, t ex LDA som betyder ladda. I assembler skriver man dessa memosymboler i stället för att skriva hex- eller binärkod för en viss instruktion.

Modem. En enhet som modulerar och demodulerar datasignaler. Med t ex ett telefonmodem kan man på telefonnätet analogt överföra datasignaler.

Monitorprogram. Ett dataprogram som övervakar och styr olika operationer i ett datasystem, t ex sökning i minnesceller, in- och utmatning av data till kassettbandspelar, presentation av data m m.

Multiprocesskörning. Fler program kan exekveras parallellt i en multiprocessor.

Multiprocessor. Flera centralenheter har här ett gemensamt minne.

N

NAND-grind. Inverterande och grind.

Nanosekund. 10^{-9} sekunder.

Nibble. Ett dataord bestående av 4 bitar.

NIBL är beteckningen på ett Basicliknande språk för styrändamål som National Semiconductor tagit fram för sin dator SC/MP.

NOR-grind. Negativ eller-grind.

NOT-element. Negerande element. Samma som inverterare.

O

Obeständigt minne. Minne som tappas sin information då strömförsörjningen avbryts. Kallas även flyktigt minne (volatile storage).

Objektкод. Den kod som datorn kan arbeta med och som är ett resultat av en kompilering eller assemblering.

OCR. Optical Character Recognition. Optisk teckenidentifiering.

Oktal. Talsystem med 8 som bas. Se avsnittet talsystem.

Ordlängd. Antalet bitar i ett ord.

Operativsystem. Ett program som sköter.

P

Packad BCD. I ett system med 8 bitars ordlängd läser datorn de högsta 4 bitarna som ett BCD-värde och de lägsta som ett BCD-värde.

Paritetsbit. En bit i ett tal som används för kontroll. Bitarna i talet summeras och resultatet skall alltid vara udda eller jämnt.

Paritetsfel. Fel i t ex en dataöverföring indikeras i det att paritetskontrollen ger fel värde.

Periferiutrustning. Kringutrustning till datorn, t ex skrivare, hållremsläsare, OCR-enheter, flexskivesystem m m.

Permanent minne. Ett minne vars innehåll inte kan modifieras, t ex ett ROM.

Pekare. Eng pointer. Ett register som pekar ut en adress.

PIA. Peripheral Interface Adapter. En krets som medger parallell överföring av data till och från pe-

riferienhet.

PLA. Programmed Logic Array. En krets bestående av grindar vars sammankoppling efter ett visst mönster kan programmeras. Ersätter i vissa applikationer PROM.

PL/M. Ett högnivåspråk framtaget av Intel mikroprocessorer. Det är baserat på IBM-språket PL/I som är utvecklat för stordatorer.

Polling. En metod där datorn sänder ut en fråga om någon av periferienheterna har begärt att få komma i förbindelse med datorn. Den kontrollerar statusregistret som indikerar "Interrupt request".

Port. En elektrisk in- eller (och) utgång i datasystemet. Det kan vara en enkel tråd eller parallella utgångar för överföring av ett helt ord. Se PIA och ACIA.

Primärminne. Ett minne som kan adresseras i ett datorprogram och som man kan hämta information ur för bearbetning.

Programräknare. Det register i en mikroprocessor som innehåller den adress som exekveras. Normalt stegas adressen fram 1 till 4 steg då en instruktion har blivit utförd, eftersom instruktionen kan vara 1 till 4 bytes lång. Om den instruktion som just har exekverats är en hoppinstruktion, laddas programräknaren med den adress till vilken hopp skall ske.

R

RAM. Random Access Memory. Skriv/läsminne.

Realtidsystem. Ett datasystem som arbetar med en process och som styrs av tidkrav från en yttre process.

Relativ adressering. En adresseringsmod där instruktionens adressdel innehåller en relativadress.

Relokerbar adress. Adress som kan förskjutas.

Resident programvara. Ett program som är laddat i en given dator för att generera eller avlusa program för samma dator. Ett monitorprogram eller ett operativsystem (hela eller delar av det) är residenta program.

ROM. Read Only Memory. Läsminne.

S

Sekundärminne. Ett massminne utanför datorn, t ex en flexskiveenhet. Dess kapacitet är större än datorns primärminne, men åtkomsttiden är betydligt större.

Skivminne. En roterande skiva på vars yta data är lagrat magnetiskt. Skivminnen har hårda skivor till skillnad mot flexskivor som är tunna och böjliga.

Stack. Innehåll i minnet som lagras i form av en lista så att den information som lades sist in i stacken, eller traven, även tas ut först.

Subrutin. Ett underprogram som kan återopas av huvudprogrammet. Subrutinprogrammet kan bestå av ett antal instruktioner som utför en speciell rutin.

Synkron överföring. Serieöverföring där sändare och mottagare är synkroniserade med en gemensam klocka.

T

Terminal. En enhet från vilken människan kan kommunicera med datorn.

Time sharing. En metod som innebär att två eller flera enheter använder en dator medels tidsdelning.

U

UART. Universal Asynchronous Receiver Transmitter. Används för anpassning mellan parallella och seriella snitt.

Udda paritet. Vid paritetskontroll skall antalet bitar som kontrolleras, inkl paritetsbit, med värdet 1 vara udda.

USART. Samma som UART.

V

Villkorligt hopp. Hopp i programmet som äger rum bara då vissa villkor uppfylls.

A

Åtkomsttid. Den tid som förflyter från det att datorn har anropat en enhet (t ex ett minne) och till dess data är levererade.

Klubbar för dator-intresserade

Förr eller senare brukar problem ge sig till känna vare sig det gäller maskin- eller programvara. Då kan det vara stor hjälp att vara med i en datorklubb för att få tips från andra som kanske har råkat ut för samma missöden. Datorn förutsätter också program vilka brukar distribueras mellan medlemmarna i klubben. Här visar vi en sammanställning över landets dataklubbar.

ABC-KLUBBEN

kom till under januari 1980, men hade ett halvår senare hela 800 medlemmar! Verksamheten är helt inriktad på den svenska datorn ABC 80. Lokalfrågan var inte löst när detta skrevs så man träffas sporadiskt. Programdistributionen sker i listform i tidningen som kallas ABC-bladet.

Hela 15 intressegrupper har bildats som bl a sysslar med administrationsprogram, datakommunikation, spel, grafik, musik, styr- och mätteknik, textbehandling m m.

För framtiden söker man initiera lokalavdelningar och skaffa egen lokal med dator.

Adress till ABC-klubben:

*Box 1201
171 23 Solna*

COSMAC DIGITAL GROUP

startades i augusti 1978 och hade hösten 1980 240 medlemmar. Den är helt inriktad på användandet av 1802-datorer, dvs CMOS-processorn från RCA. De datorer som används är i huvudsak Telmac 1800, men här förekommer också Oscom Nano, VIP, ELF, Super ELF samt några egna byggen.

Styrelsen håller till i Norrtälje medan medlemmarna är spridda över Skandinavien. Man håller inga klubbträffar men medlemmarna knyts samman med den tidning som distribueras med 4 nr/år.

Programmen distribueras som listor i tidningen. Några egentliga träffar anordnas som sagt inte, men hos klubbens "primus motor", Conny Winroth, Grindvägen 34b i Norrtälje kan man få titta in i ett litet rum, sammanställt för klubben, där olika datormateriel ställs ut och kan provas.

För framtiden planerar man att inom klubben ta fram ett flexskivsystem. Än så länge är man alltså hänvisad till kassetter på gott och ont.

Adress till klubben:

*Box 300
761 00 Norrtälje*

Chalmers Datorförening (CD)

Sedan starten 1974 har aktiviteten varit mycket hög på CD. Från början var verksamheten helt inriktad på stordatorer, och föreningen har för närvarande en Siemens 4004-anläggning, och några olika mikro-datorer i sina lokaler. Föreningen har idag ca 200 medlemmar. Formella möten hålls 2-6 gånger/år, men det pågår nästan alltid någon verksamhet i lokalen. Tidningen CD-nytt sänds till alla medlemmar. Ett organiserat programutbyte till självkostnadspris sker via programbanken. En lista på tillgängliga program publiceras i tidningen Editor, som även innehåller artiklar om datorer i allmänhet och programmering i synnerhet.

I stordatoranläggningen ingår tre centralenheter, den största (en 4004/135) har medlemmarna själva byggt ut från 128 Kbyte kärnminne med halvledarminne till 512 Kbyte. Bland kringutrustningen kan nämnas 5 st 29 Mbyte skivminnen, 6 st bandstationer samt radskrivare och hålkortsutrustning. Med hjälp av en på CD konstruerad kommunikationsdator kan flera terminaler anslutas till systemet. Följande programmeringsspråk finns tillgängliga: *Algol, cobol, PL/I, RPGII, PL/360, fortran* och *as-*

sembler (basic är under utveckling).

Föreningen förfogar också över ett antal mikrodatorsystem nämligen: **Zilog MCZ** med två floppy, **Mikroset 8080** och **CD990** med två floppy. **CD9900** systemet är utvecklat vid föreningen och uppbyggt kring **Texas TMS9900** 16-bitars mikrodator. Ytterligare ett **CD9900**-system är under byggnad. Till dessa system finns följande mjukvara: Operativsystem **UCSD-pascal**, **RIO** och **TDL** till **MCZ**-systemet och ett eget **DOS** till **CD9900**-systemet. Implementerade programmeringsspråk är: **MCZ: basic, pascal, PL/Z, pilot och assembler**. **Mikroset: Enbart maskinkod för 8080. CD9900: C, lisp, basic och assembler.**

Bland aktiviteterna kan nämnas sk samköp. Medlemmarna köper då in komponenter tillsammans för att kunna få lägre priser. På detta vis har en **CD**-konstruerad bildskärmsterminal, **CD9900**-datorer och flexskivenheter anskaffats. En del seminarier om div olika datorämnen liksom kurser har genomförts och planeras även för framtiden. Verksamheten omfattar också bland annat konstruktion av datorer, grafiska bildskärmar m m samt framtagande av kompilatorer, operativsystem, texthanteringsprogram och naturligtvis en uppsjö av spelprogram . . .

I den närmaste framtiden (1980) kommer ett nytt terminalsystem (trol. **UNIX**-baserat) att skrivas till **4004**-systemet och **UCSD-Pascal** att införas på **CD9900**.

Adress till **Chalmers Datorförening**:

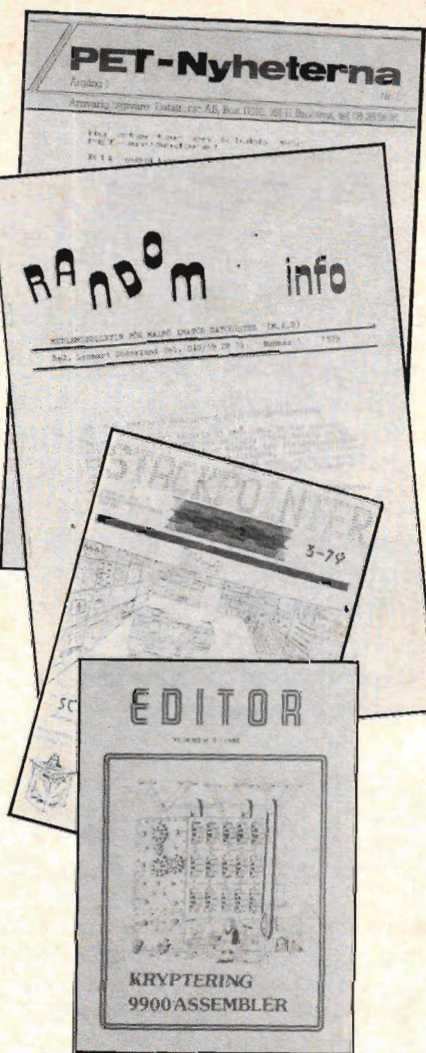
412 96 Göteborg

Gatuadress:

Rönnevägen 8 (Chalmersområdet)

Lysator

bildades redan 1963 och är landets äldsta datorförening och bland de äldsta i Europa. Idag finns omkring 100 medlemmar och föreningen har en egen lokal i Linköping. Det hela startade i och med att **Datsa** skänkte en **D21:a** till klubben. Senare tog man fram en 16-bitars hobbydator som några år produce-



rades av **ATEW** i Flen, men tillverkningen lades ned av olika anledningar.

De datorer som används av klubbmedlemmarna idag är **ABC 80, Apple II**, båda i assembler- och basic-språk, **Datsa** **D5/30** i **fortran, basic** och **assembler**, **D21** och **D22** i **Algol-genius, assembler, LYS 16** i **basic** och **assembler, Univac 9200-II, Varian 520** i **assembler**. Dessutom förekommer andra datortyper bland medlemmarna.

Tidningen **Garb** utkommer när det finns material och man har föreningsmöte ungefär var 6:e vecka under terminerna men nästan alltid är någon verksam i klubblokaler. Någon distribution av programverksamhet sker inte, kanske p g a att man använder så olika

och i amatördatorsammanhang udda datorer.

Bland de övriga aktiviteterna kan nämnas utarbetande av rutiner för anslutning av plotter till **D21**, mikroprogrammerbara bit-slice-processorer för t ex pascalmaskiner, interaktiv terminalhantering för **D22**. För framtiden satsar man på mjukvara för **D22** bl a Adress till klubben:

Datorföreningen Lysator
c/o ISY

Universitetet i Linköping
581 83 Linköping

MAD - Malmö Datorförening

Klubben startade i augusti 1977 och hade ca 60 medlemmar sommaren 1980. Man har inte specialiserat sig på någon speciell dator. Inom klubben förekommer **IMSAI 8080, ABC 80, 6802**-datorer, **Sorcerer, SC/MP**-datorer m fl. Man träffas andra torsdagen i månaden med uppehåll i juni och juli. Programdistributionen sker spontant medlemmarna emellan. Programmen listas dock i tidningen "Random Info". Programmen lagras på 8" flexskiva enl CPM och många för över dem till 5 1/4 flexskiva. Till **ABC 80** förekommer program både på skiva och på kassett. **6802**-användarna använder **Kansas City** standard på kassetterna. Man använder mest **basic**, men **fortran** och **pascal** förekommer inom klubben.

Övriga aktiviteter är föredrag av olika slag. De har t ex handlat om olika programspråk som **lisp, Pascal, cobol** m fl eller t ex referat från **USA**-besök. En hårdvarugrupp har byggprojekt som vänder sig direkt till nybörjare. En annan grupp håller på att utveckla en robot, ytterligare en har konverterat en **pascal**-kompilator så att den passar datorer med **CPM** hanteringssystem för flexskivor.

Adress till **MAD**:

MAD c/o Lennart Söderlund
Södra Gulsparvsgatan 42,
214 61 Malmö

Privatdatorklubben PD68

Klubben kom till hösten 1977 och har idag drygt 200 medlemmar.

Den är specialiserad på mikroprocessorer av typen 6800/6809 och kanske kommer även 68000 att användas. I begynnelsen arbetade de flesta med små enkortsdatorer från Motorola av typ D2 och D1. Idag har flertalet medlemmar mera kapabla datorer som arbetar med bl a basic. Många är byggda efter RT:s beskrivningar. De flesta använder kassetband som massminne men allt fler går över till den snabbare flexskivan.

Sista torsdagen varje månad, utom juni och juli, är det klubbmöte med föredrag, demonstrationer och diskussioner.

Klubbtidningen heter MPU-laren. Den brukar innehålla artiklar om program av olika slag, t ex användar-, monitor- och systemprogram. Programmen listas ofta i tidningen men inom klubben distribuerar man också kassetter och flexskivor med basic- och assemblerprogram. Maskinvara, som t ex anpassningskretsar mellan datorer och skrivare med tillhörande program, tar man även upp i spalterna. Tidningen utkommer när det finns material som räcker.

För framtiden planerar man att eventuellt skaffa en klubb dator. Den skall kunna användas direkt eller nås via modem-förbindelse, dvs över telefonnätet. Kanske blir datakommunikation över tråd eller med radio en ny och spännande specialgren av datorhobbyn?

PD68 har adressen:

Box 98

122 21 Enskede 1

PET användarklubb

Klubben kom till i maj 1979 och medlemsantalet under hösten året efter var uppe i 250. Vid den tiden hade två nummer kommit ut av klubbens tidning, men man räknar med 6 nr per år. Till medlemmarna skall man dessutom distribuera engelska och tyska nyhetsbrev från motsvarande klubbar.

En programbyttarklubb finns med i verksamheten vilket innebär att man skickar in ett egenhändigt framtaget program och får så ett annat program i byte. På så sätt räknar man med ett aktivt delta-



gande från medlemmarna med en ständigt ökande programbank att ösa ur.

Eventuellt kommer man att anordna klubbträffar och man planerar att driva kurser i klubbens regi.

Aktiviteterna är, som namnet på klubben antyder, helt knutna till datorn PET i dess olika versioner.

Adress:

PET Användarklubb

Fack

104 40 Stockholm

Stacken

Är en klubb som har sitt ursprung den 18:e maj 1978. Låter det inte historiskt så säg? Då bildades styrelsen och man tog fram stadgar. Så småningom kom det igång klubbträffar, som hölls i en lokal som ställdes till förfogande av

KTH i Stockholm. När boken gjordes hade man blivit av med den lokalen och arbetade för att få fram en ny. Träffas gör man dock. Närmare bestämt första torsdagen i varje månad i sal E7 på KTH och dit kan alltså de som ev är intresserade vända sig.

Många medlemmar har byggt sina datorer som mestadels är bestyckade med Z80-processor. Systemen storlek varierar: Från små maskinspråksdatorer upp till basic- och pascaldatorer med flexskivsystem. Man har också börjat titta på 16-bitarsmaskiner med Z8000-processor, ev MC68000. Klubben har en egen Seven-S dator som är byggd kring 4680-kort.

Tidningen heter Stackpointer och kommer ut när det finns material för den.

Programvara distribueras inte centralt men utbyte sker mellan medlemmarna och vidare har man samarbete med Chalmers datorförning.

Kurser är en aktivitet man har satsat på och en speciell grupp är bildad för 16-bitarsdatorer.

Stackens adress är:

Box 5074

141 05 Huddinge

Sällskapet

6502-användare

Det här är en direkt efterföljare till KIM-klubben som det bara fanns rester av då vi tog kontakt med dem. Sällskapet kommer att omfatta användare av datorer med 6502-processor, t ex KIM, SYM, Apple, PET, Ohio Superboard, AIM 65 m fl.

Idéerna till det nya sällskapet är i stort sett följande:

■ Det skall finnas en lista över dem som använder 6502-bestyckade datorer.

■ Ett informationsblad skall ges ut med tips om maskin- och programvarat. Där skall även medlemmarna kunna annonsera utan kostnad.

■ Sällskapet skall stödja en kursverksamhet i samarbete med Studieförbundet.

Adress till 6502-sällskapet är:

Box 1135

163 12 Spånga

Böcker om mikrodatorer

Mikrodatorns område är stort. Allt kan inte rymmas inom den här bokens pärmar. Vi ger därför några tips på läsvärda böcker för dem som vill tränga vidare i ämnet.

1) *Osborne, Adam: An introduction to microcomputers, vol O, The beginners computer.* Titeln är kanske något förbryllande. Boken kom till efter det att vol 1 fanns ute och vad var naturligare än att starta med siffrorna noll för en bok som var mera elementär än vol 1. Den vänder sig till rena noviser. Generellt sett gäller för Osbornes böcker att de är *mycket* pedagogiska med en nära nog oöverträffbar framställning. I varje kapitel finns ett antal avsnitt med fet stil. Om man vill kan man utan att tappa tråden läsa bara denna och fördjupa sig i de avsnitt som intresserar extra mycket. Bättre nybörjarsbok finns inte.

Adam Osborne har även givit ut andra böcker om mikrodatorer:

2) *An introduction to microcomputers, vol 1, Basic concepts.*

3) *An introduction to microcomputers, vol 2, some real products.*

4) *8080 programming for logic design.*

5) *6800 programming to logic design.*

6) *8080A/8085 assembly language programming.*

7) *Some common basic programs.*

8) *Payroll with cost accounting in basic.*

9) *6800 assembly language programming.*

10) *Markesjö, Gunnar, Bli bekant med en minidator.* En av de första svenskspråkiga böcker om datorn, hur den fungerar och programmeras. Det är en lärobok som mycket väl lämpar sig för självstudier tack vare en framställning som är lätt att följa.

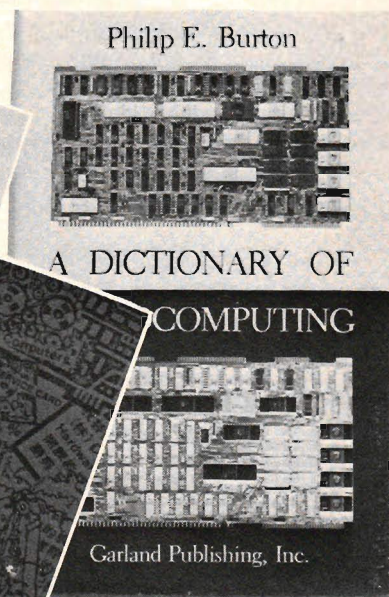
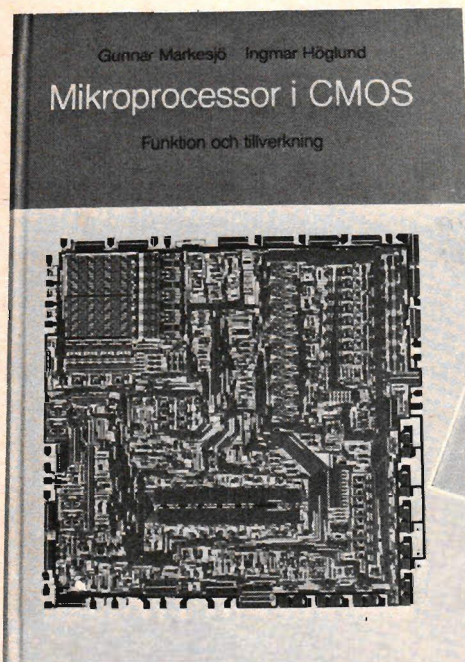
11) *Zaks, Rodnay: Microprocessors: From chips to systems.* En mycket omfattande bok om mikro-

datorer på hög teknisk nivå.

12) *Zaks, Rodnay: Microprocessors interfacing techniques.* Datorns anpassning till omvärlden är ett svårt kapitel. Boken ger uttömmande svar på hur problemen skall lösas.

13) *Bergström, Lennart: Mikrodatorn.* ISBN 91-970296-0-2. En mycket bra bok på svenska. Här får man förklaring på hur mikroprocessorn, minnen och periferikretsar fungerar, hur mikrodatorn är uppbyggd, vad den används till och hur den programmeras. Varje kapitel börjar på en elementär nivå och svårighetsgraden ökar därefter. Som komplement till boken finns ett arbetshäfte med övningsexempel.

14) *Hemert, Lars-Hugo: Grundläggande mikrodator teknik. Studentlitteratur.* ISBN 91-44-15161-6. En skolebok som även kan användas för vidareutbildning inom industrin och för självstudier. Framställningen är baserad på processorn 8080 som detaljerat genomgås vad avser uppbyggnad, funktion,



instruktionsrepertoar och anpassning till omgivningen. Styrning exemplifieras med en konstruktions-skiss till ett litet programmerbart styrsystem. Översiktligt orienteras om periferikretsar, programspråk, översättare, hjälpprogram och utvecklingssystem för mikrodatare. Boken innehåller ett flertal övningsuppgifter med lösningar.

15) **Markesjö, G, Höglund, I: Mikroprocessor i CMOS.** Esselte studium. ISBN 91-24-27562-X. Boken handlar om mikroprocessorns inre uppbyggnad. Som exempel har man valt en mikroprocessor som tillverkas av Asea Hafo med beteckningen CMC8. Här får vi läsa om såväl CMOS-processen vid tillverkning av kretsarna och funktionen i detalj med instruktioner och kontrollfunktioner.

16) **Hallberg, Tord-Jöran: Hur datare fungerar.** Studentlitteratur. ISBN 91-44-16591-9. Titeln indikerar kanske att det är fråga om en bok för nybörjare, men texten är mycket faktaförtätad med nästan

matematisk, logisk uppställning och boken får snarast ses som ett komplement till tidigare läsning om datare. Här finns bl a genomgång av avbrottsrutiner, DMA (direkt minnes access) och en genomgång av speciella datare som ortogonala datare, enbitsdatare och bilddatare.

17) **Wennersten, B G: Privatdatare – din egen datare.** Studentlitteratur. ISBN 91-44-14371-0. En orienterande bok som direkt vänder sig till nybörjare och datarehobbyister. Lättläst och översiktlig.

18) **Elementärt om mikrodatare.** Nordpress AB. ISBN 91-970242-0-1. En översättning av Motorola-boken "Understanding Microprocessors". En relativt elementär och lättförståelig introduktion för alla som vill bekanta sig med mikrodatare.

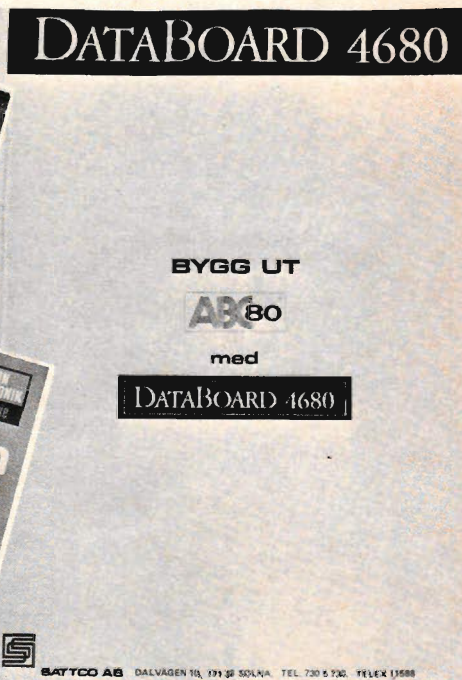
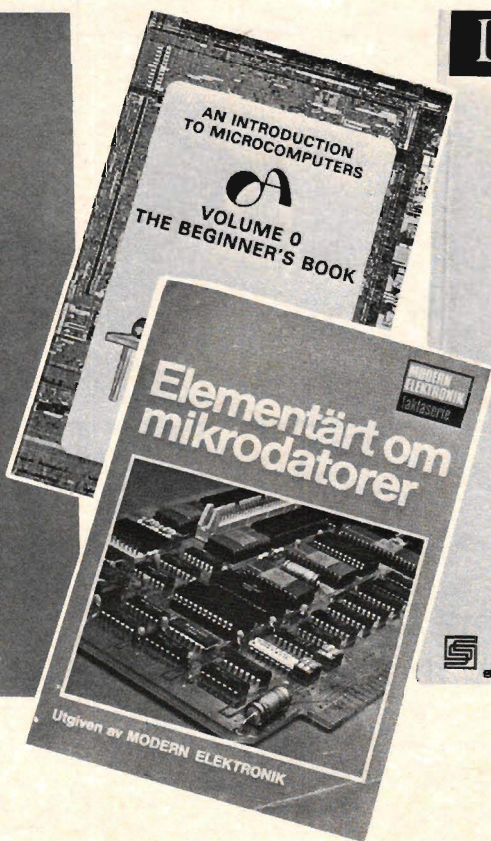
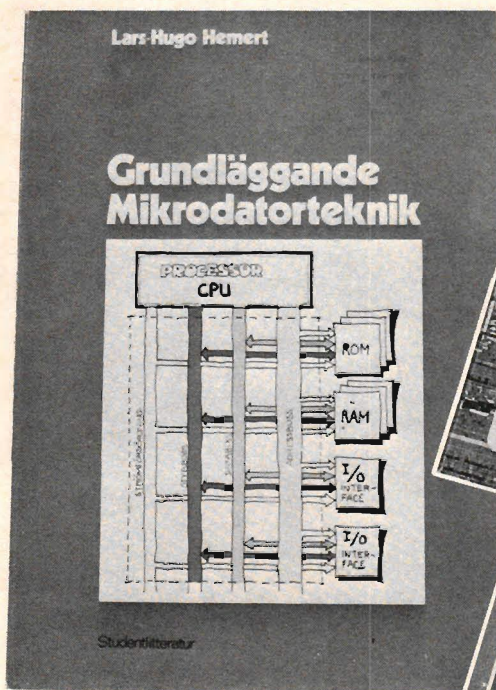
19) **Håkansson, Bertil: Den mångsidiga mikrodataren.** Ingenjöröförlaget. ISBN 91-7284-086-2. En elementär introduktion till mikrodatarens värld även för dem som inte

är så insatta i tekniken.

20) **Burton, Philip: A dictionary of microcomputing.** Garland Publishing, Inc. ISBN 0-8240-9930-3. Ett utmärkt lexikon som ger svar på det mesta om mikrodatare.

21) **Dataordboken.** SIS – Sverige. ISBN 91-7162-052-4. En ordbok för svenska dataord är verkligen en tillgång i den annars engelskspråkiga elektronikvärlden. Tyvärr är förklaringarna till orden ibland ganska kryptiska och intetsägande.

22) **Bygg ut ABC80 med DataBoard 4680.** Boken är utgiven av Sattco som här utförligt beskriver hur man kan använda deras 4680-kort i kombination med dataren ABC 80. Kortet kan inte bara utnyttjas för minnesexpansion utan även för kommunikation med omvärlden med de i/o-enheter som finns. Ett stort antal exempel med scheman och program visar hur det går till.



Appendix

På de följande sidorna återges kretskortsmönstren till de olika datorkorten i skala 1:1 eller 1:2. Dessa bilder är avsedda för de som vill göra kretskorten själva till sin egen dator. Det är inte tillåtet att kommersiellt tillverka kretskorten.

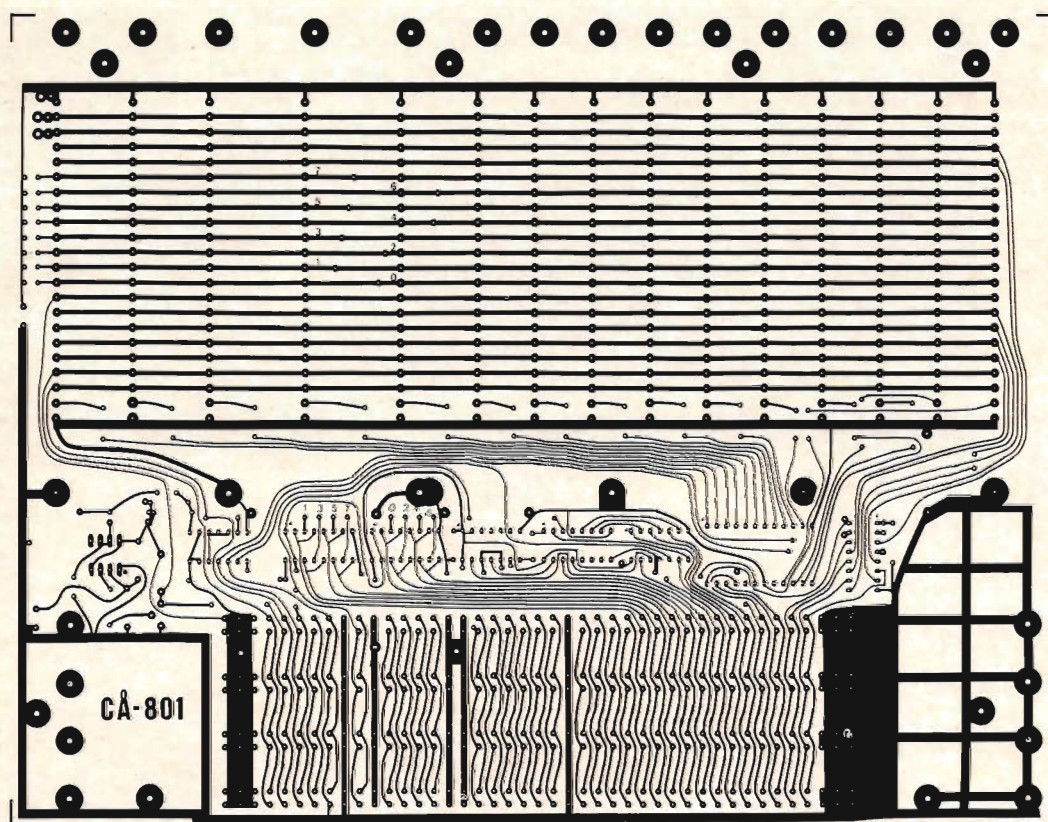


Fig 32:1. Kretskortbild för moderkort till RT-datorn. Skala 1:2.

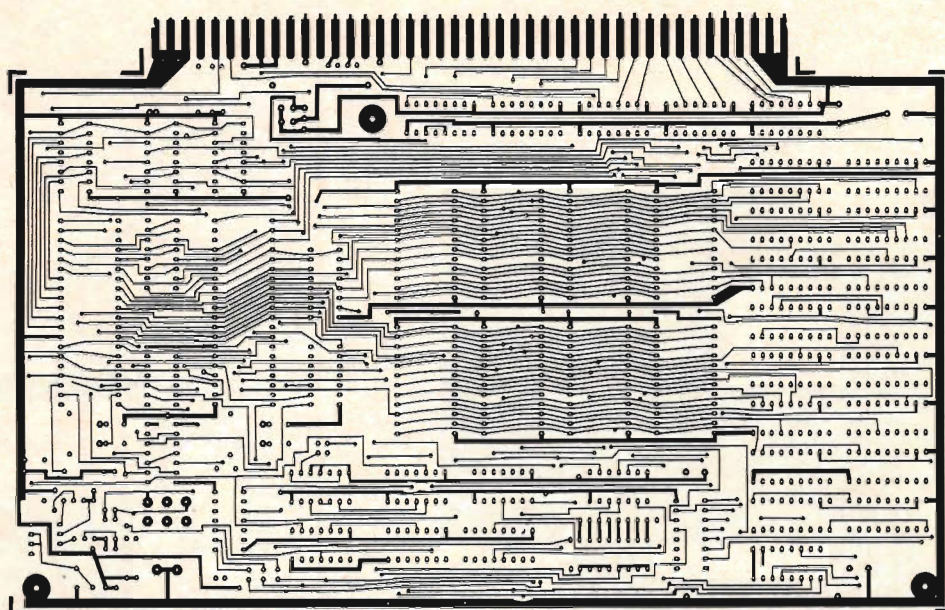


Fig 32:2. CPU-kortets lödsida. Skala 1:2.

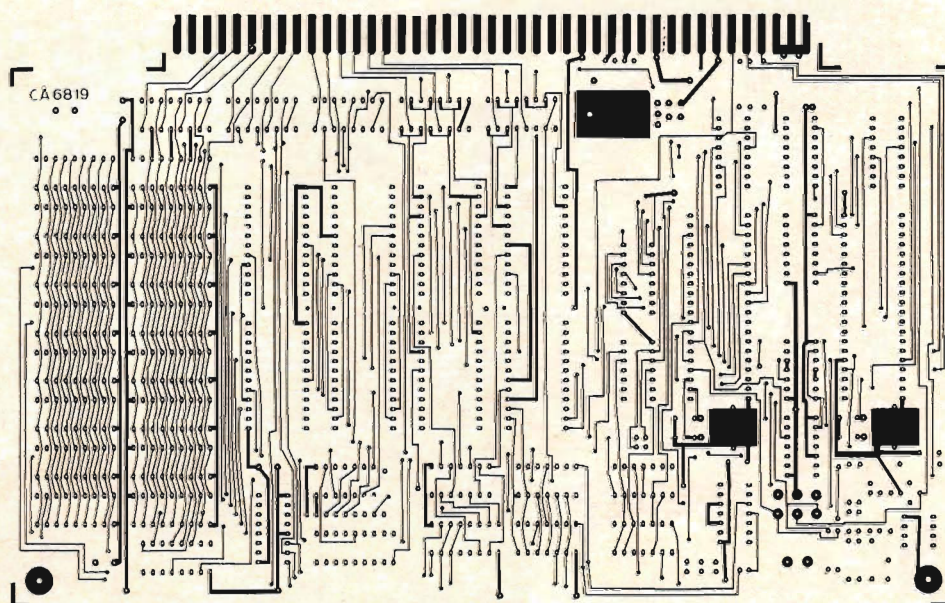


Fig 32:3. CPU-kortets komponentsida. Skala 1:2.

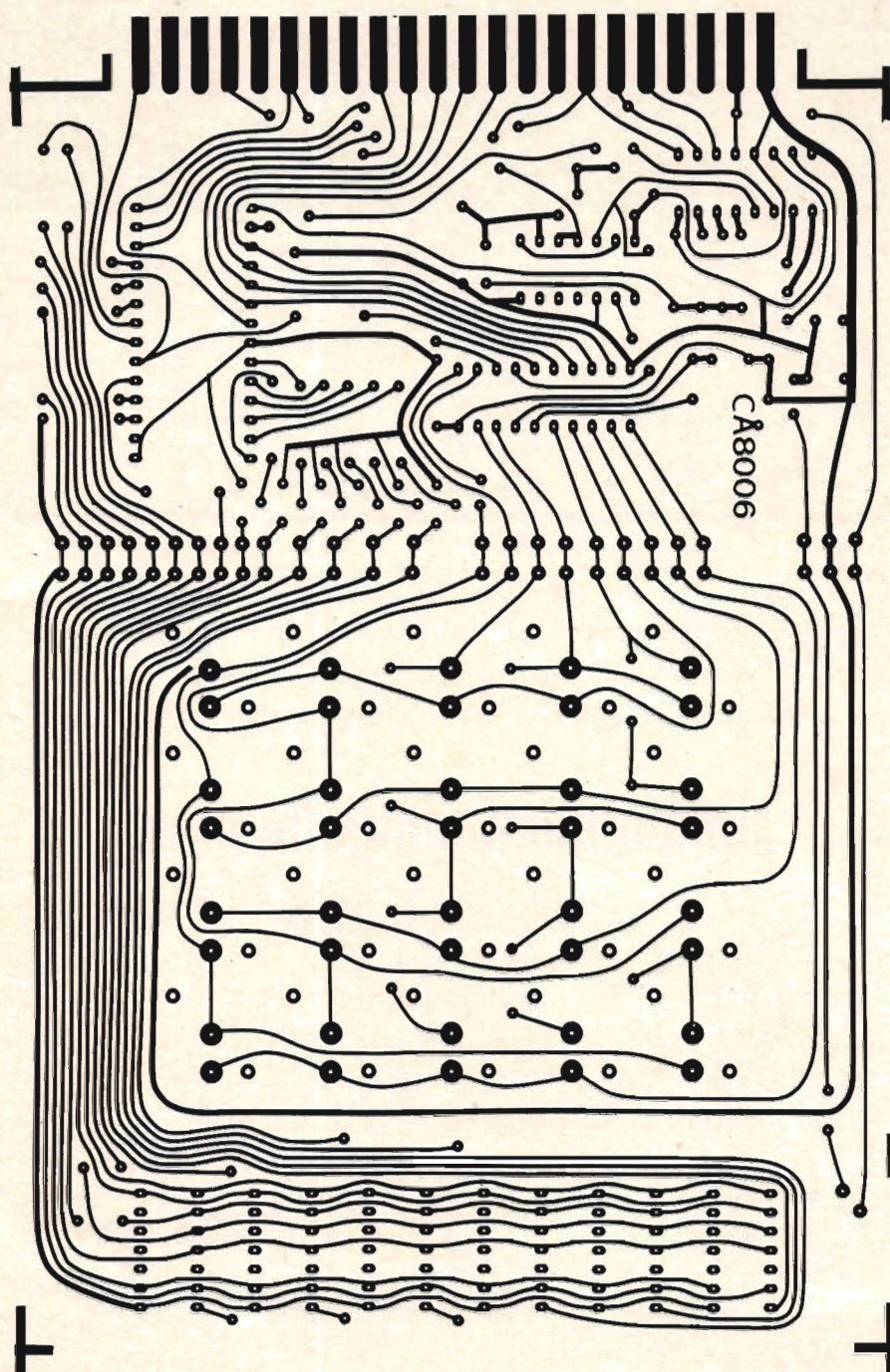


Fig 32:4. Det lilla terminalkortet. Skala 1:1.

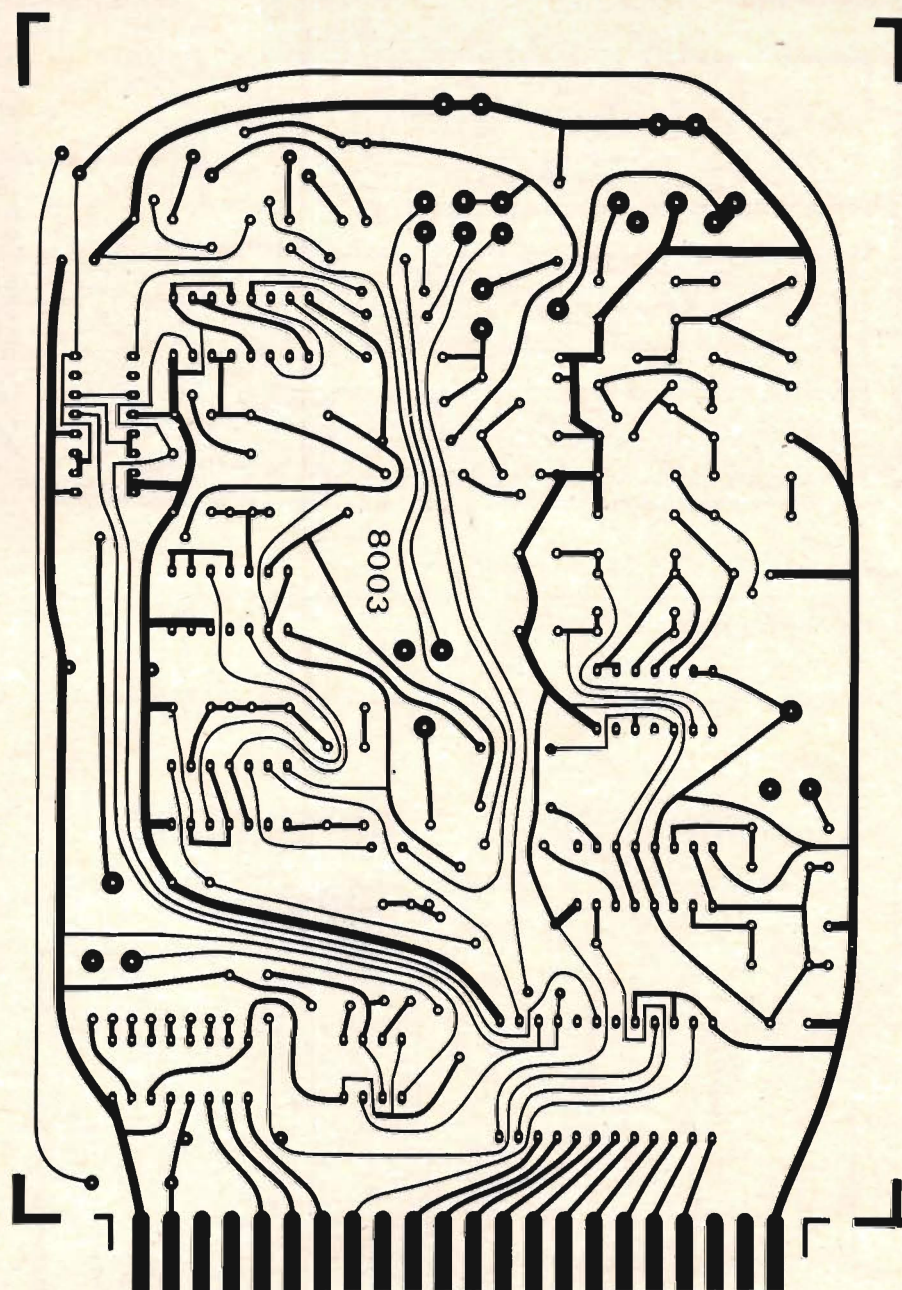


Fig 32:5. KCS-modemets kretskort. Skala 1:1.

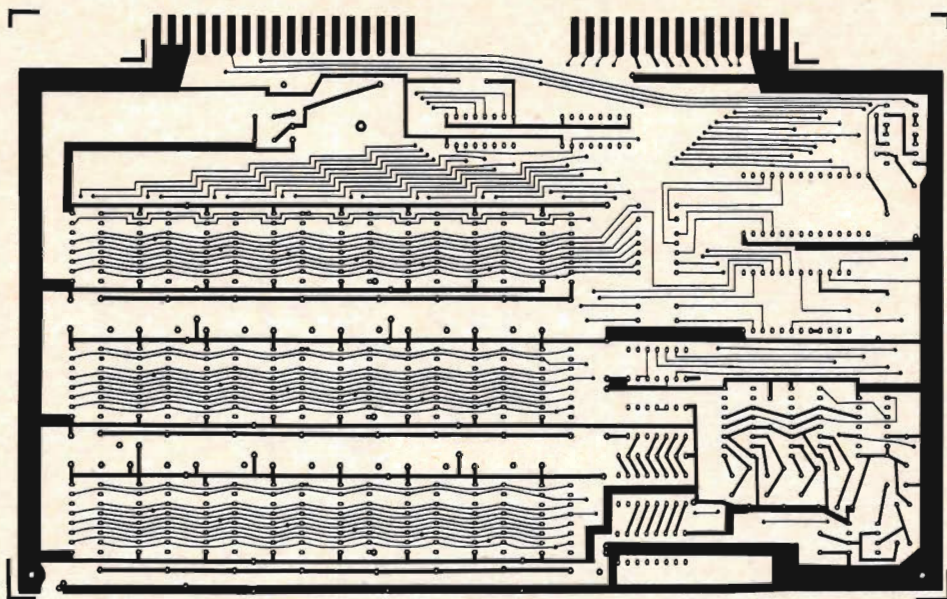


Fig 32:6. Minneskortets lödsida. Skala 1:2.

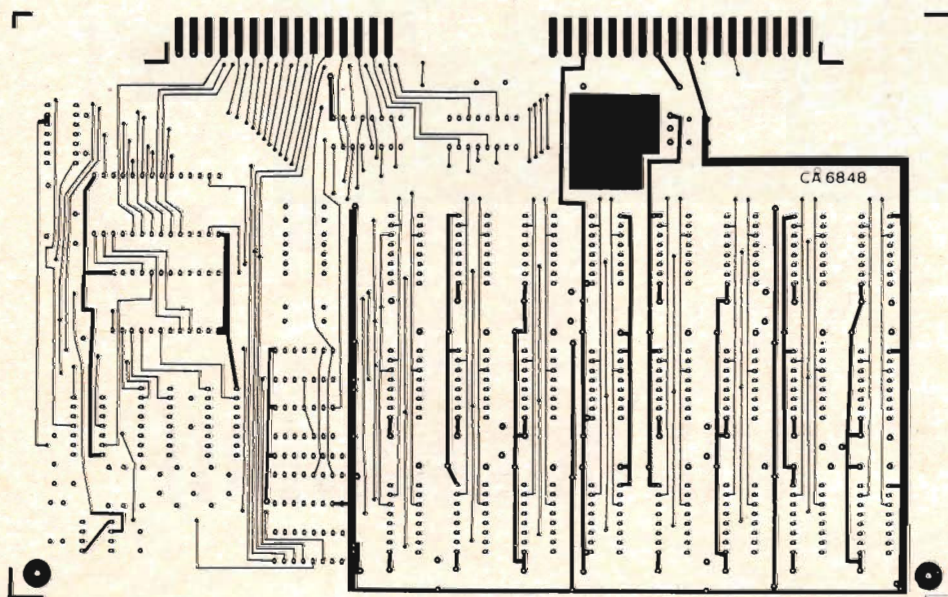


Fig 32:7. Minneskortets komponentsida. Skala 1:2.

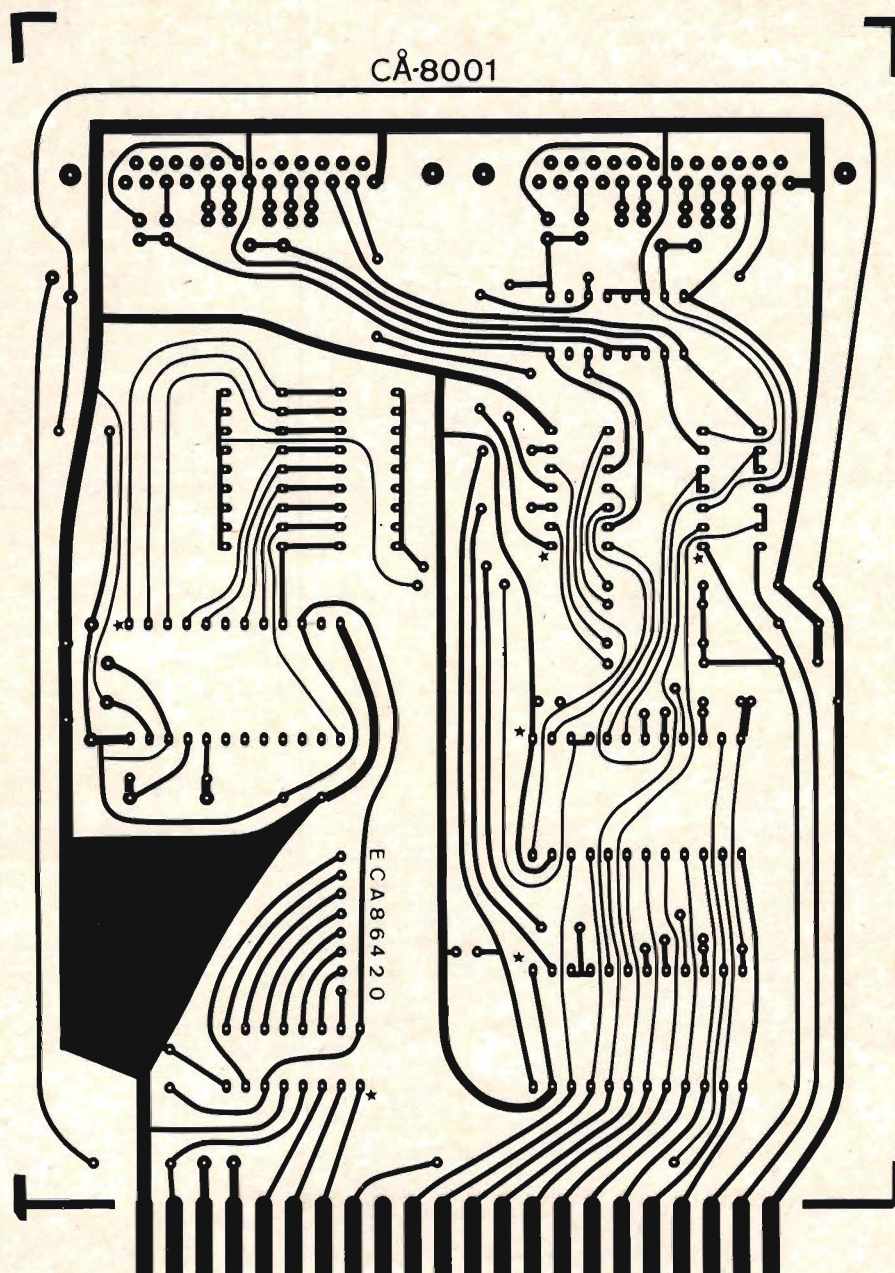


Fig 32:8. ACIA-kortets lödsida. Skala 1:1.

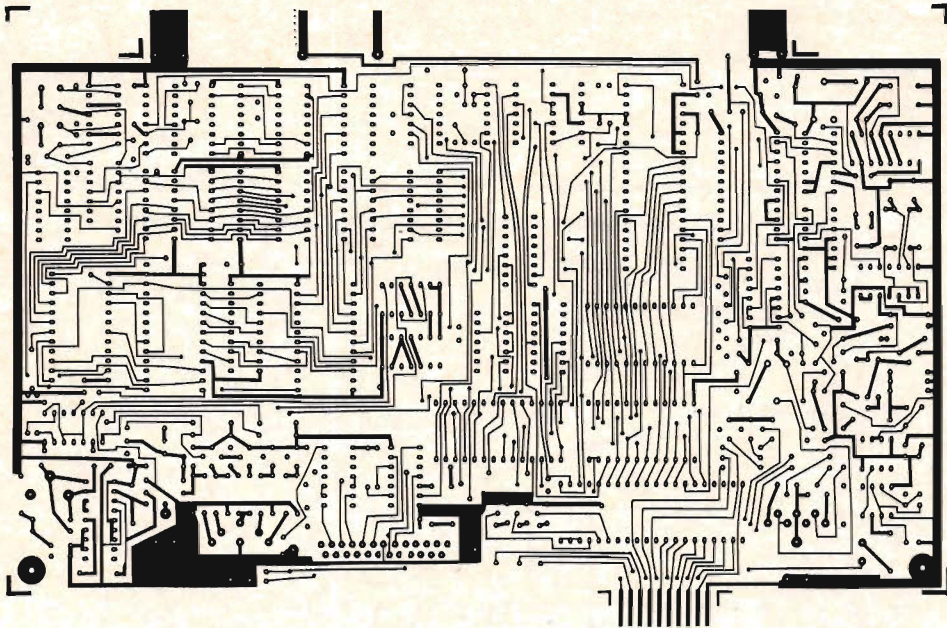


Fig 32:9. Videoterminalens lödsida. Skala 1:2.

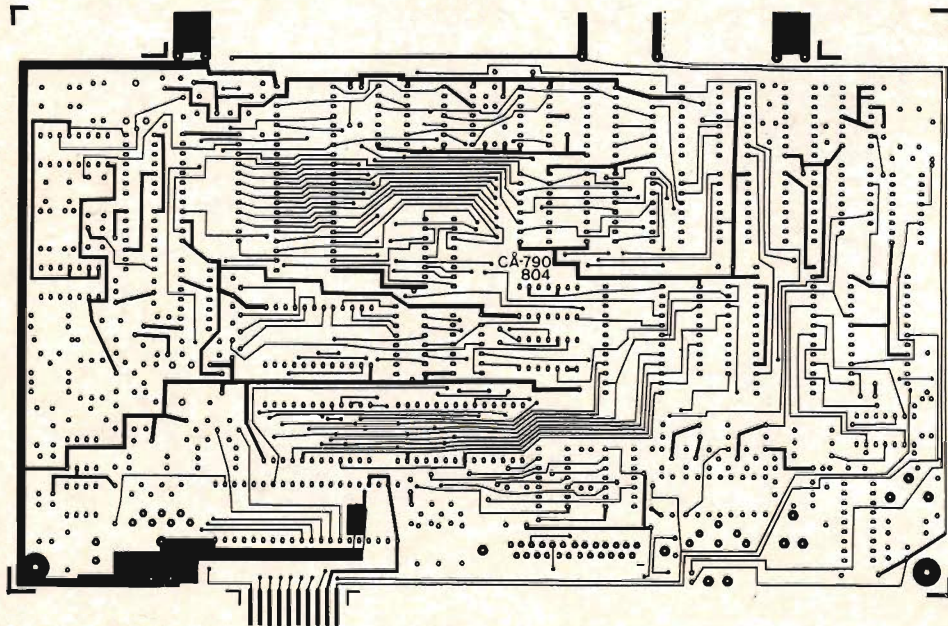


Fig 32:10. Videoterminalens komponentsida. Skala 1:2.

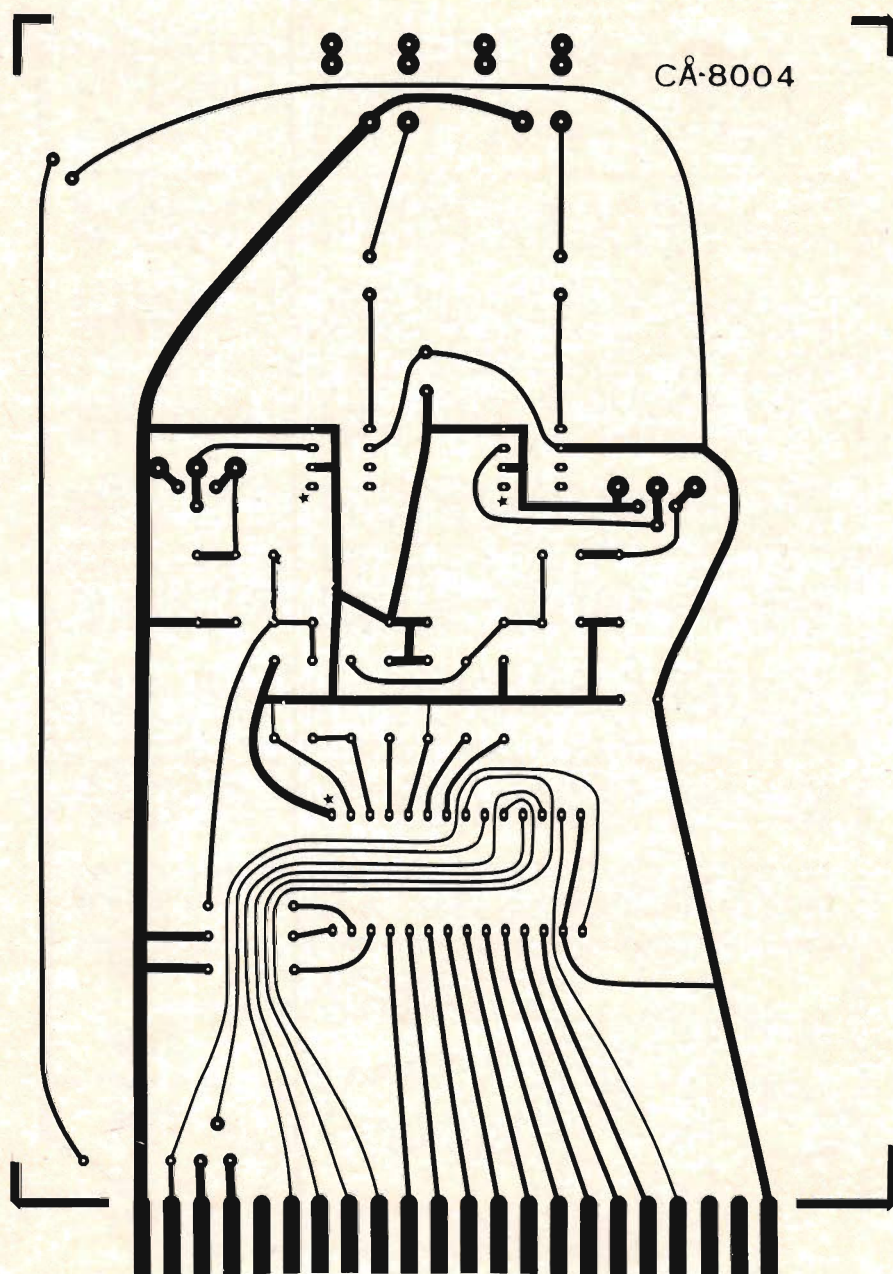


Fig 32:11. Timerkörtet. Skala 1:1.

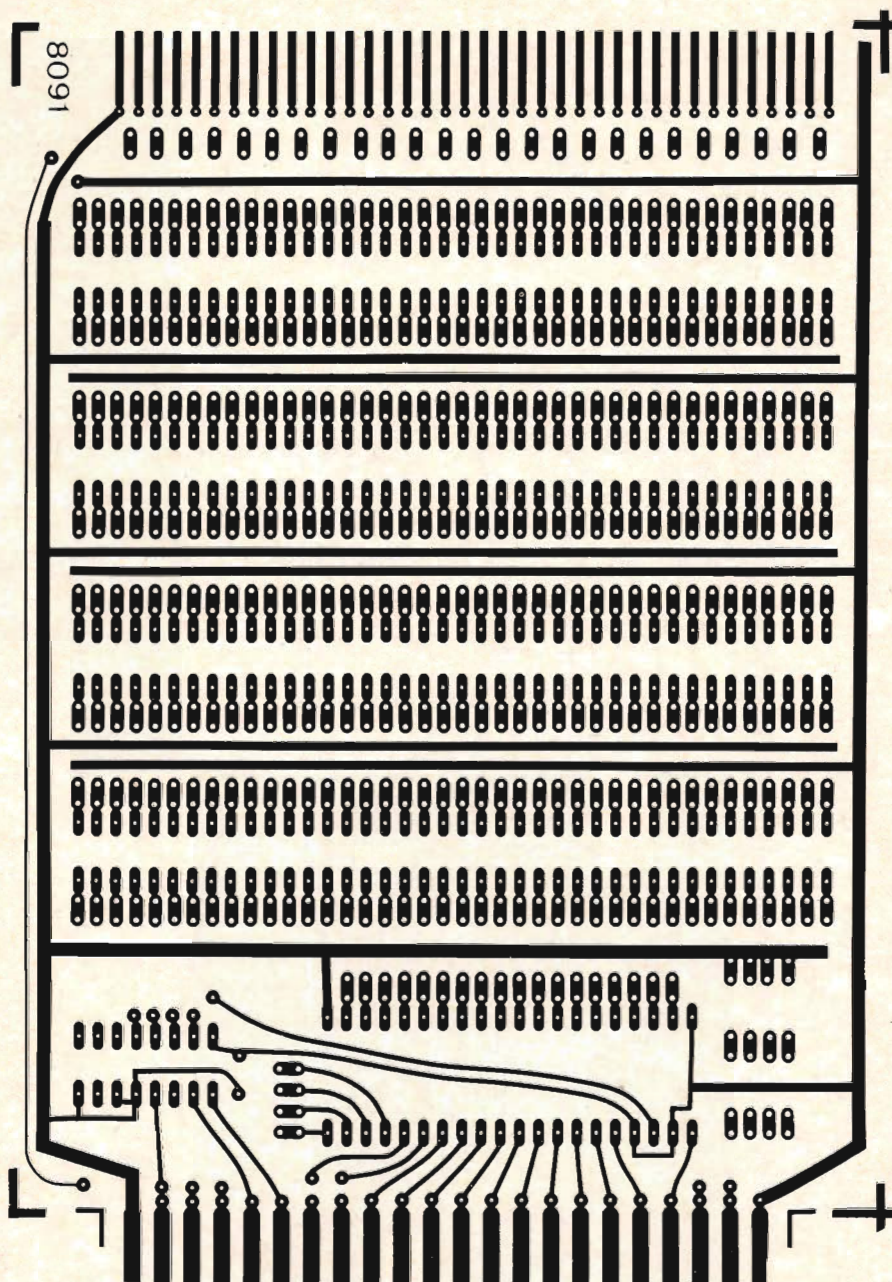


Fig 32:12. Labkort med PIA. Skala 1:1.

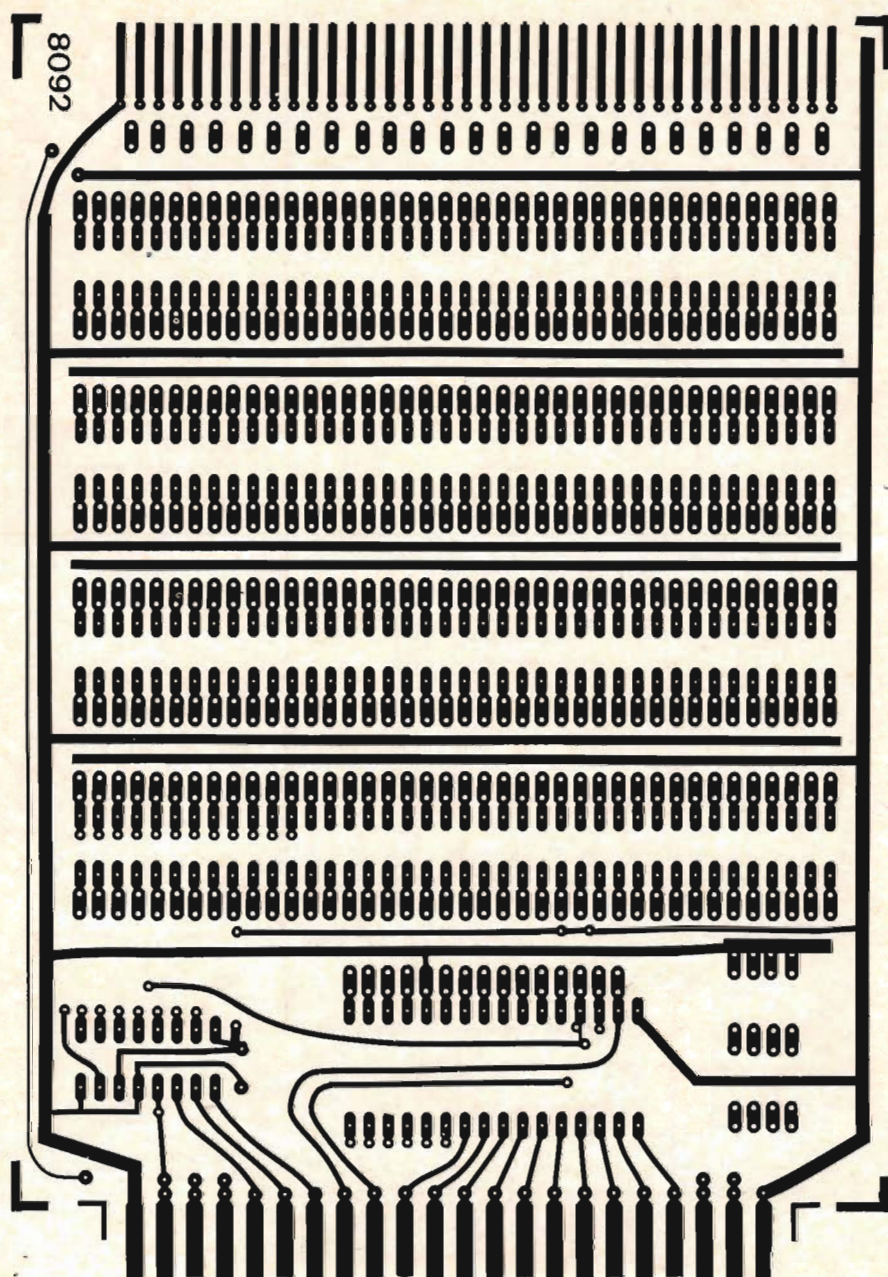


Fig 32:13. Labkort med ACIA. Skala 1:1.

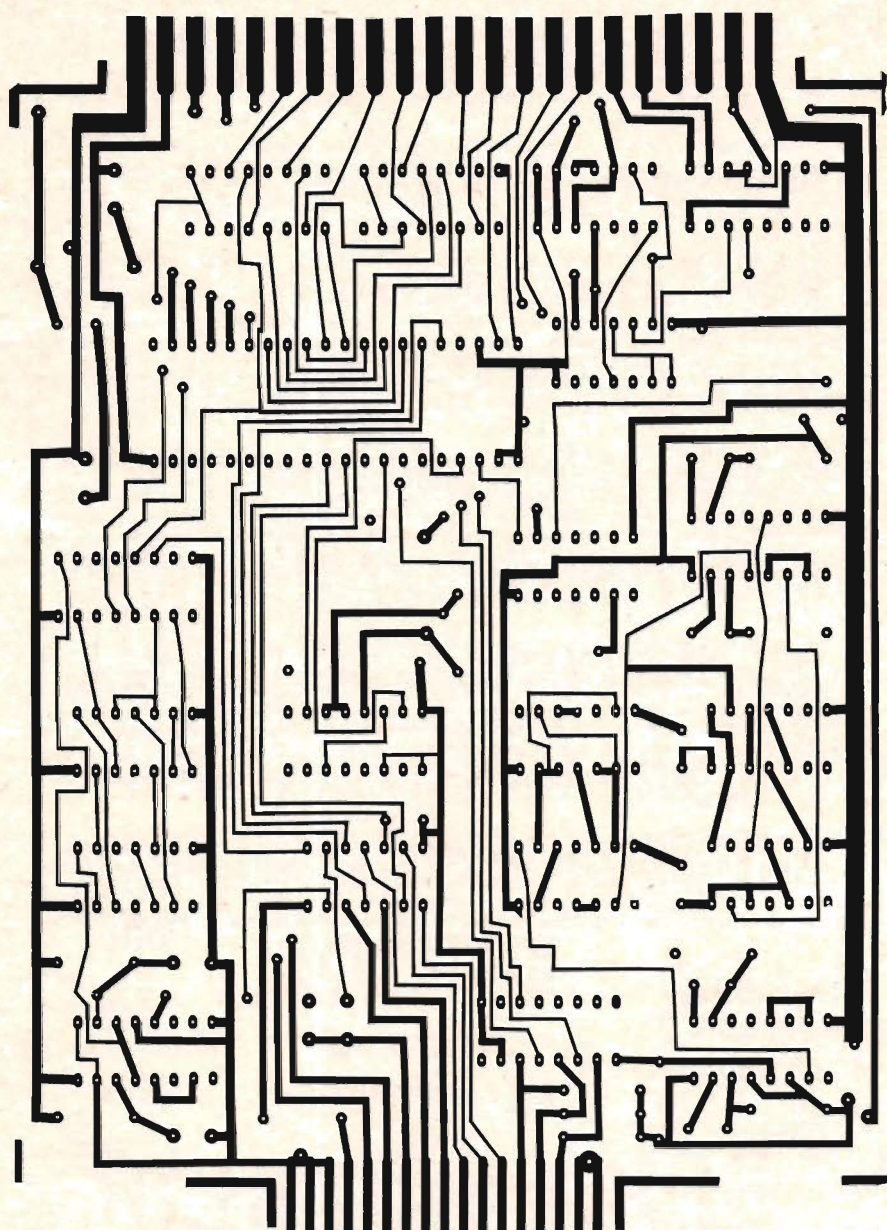
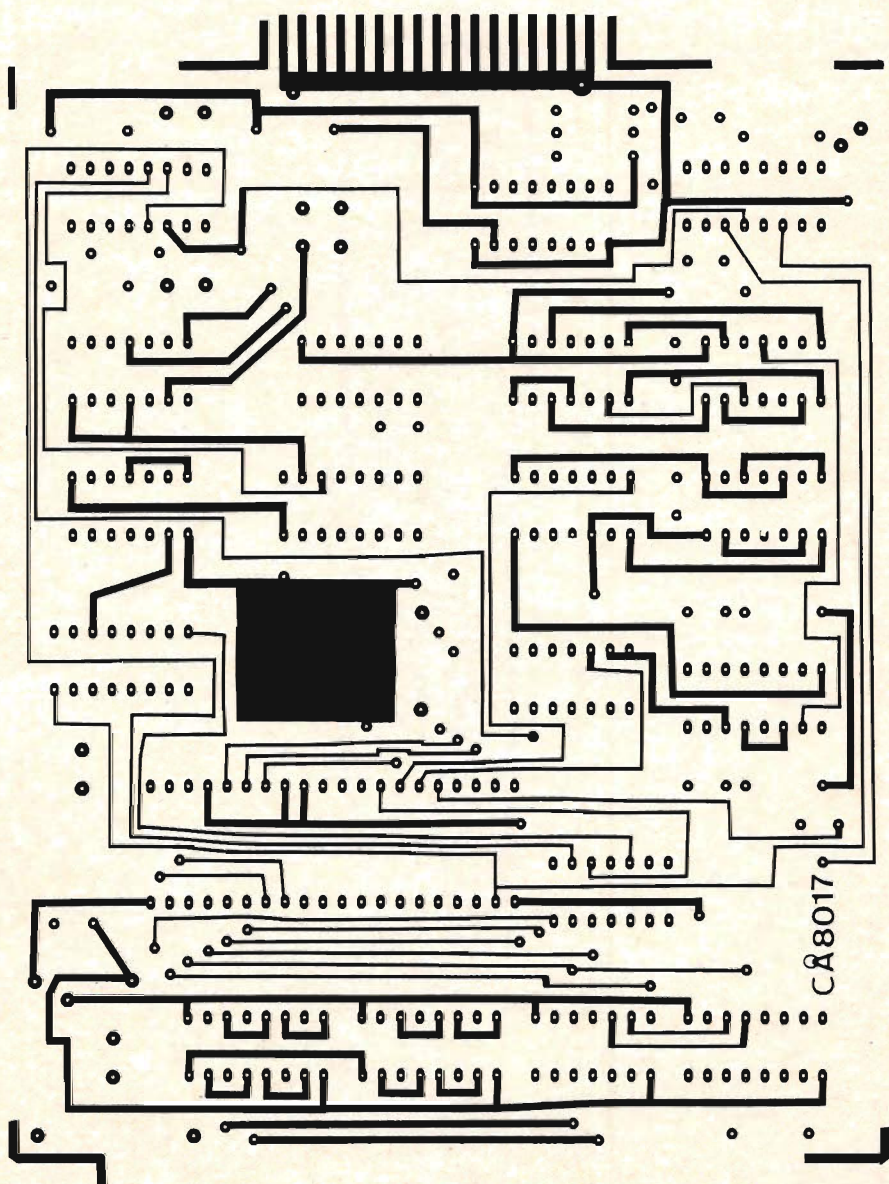


Fig 32:14. Flexkontrollkortets lödsida. Skala 1:1.



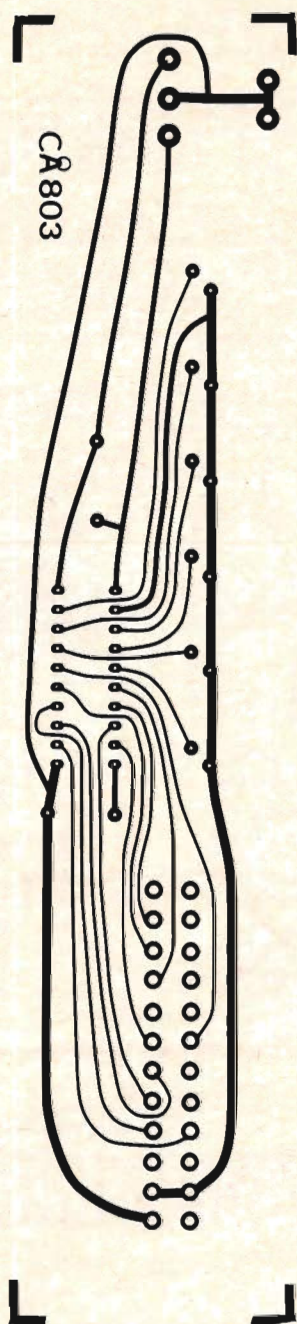


Fig 32:16. Kretskortet för tangentbordet. Skala 1:1.

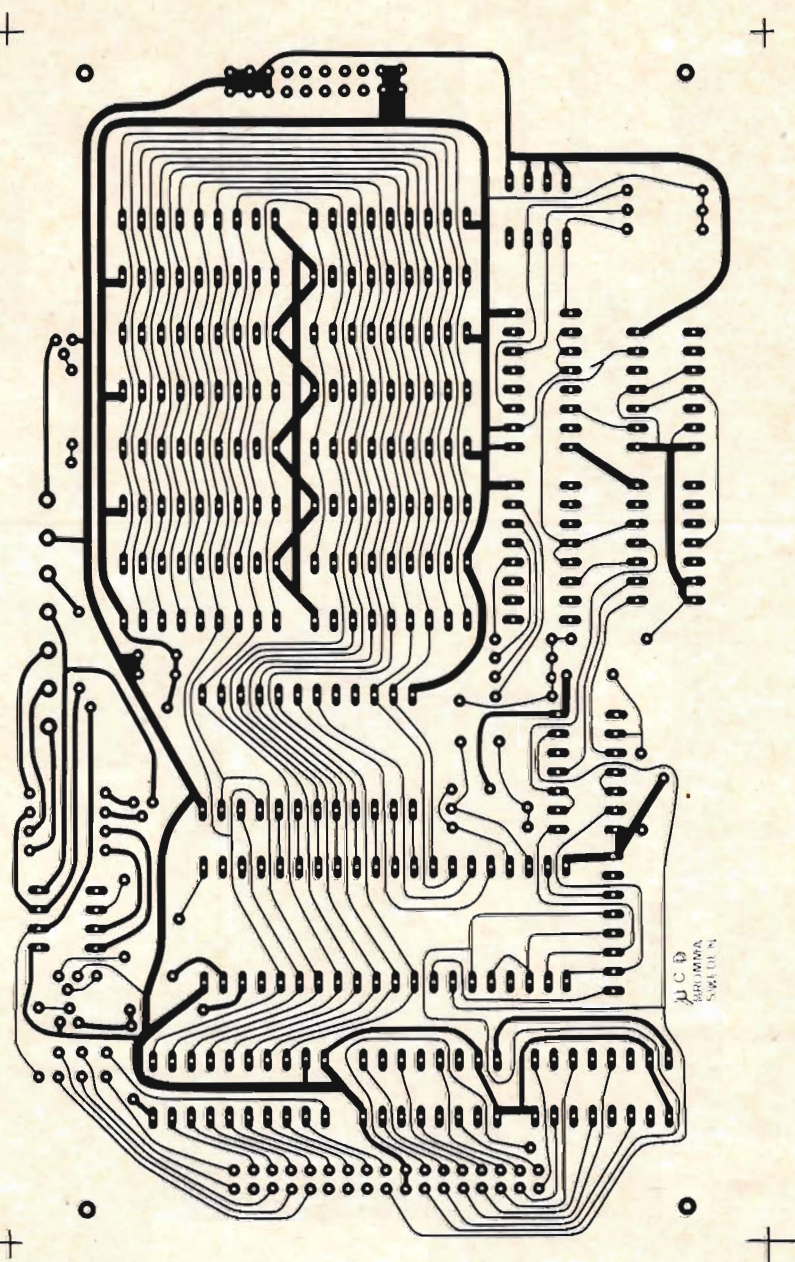


Fig 32:17. Basic-dator på ett kort. Undersidans mönster. Skala 1:1.

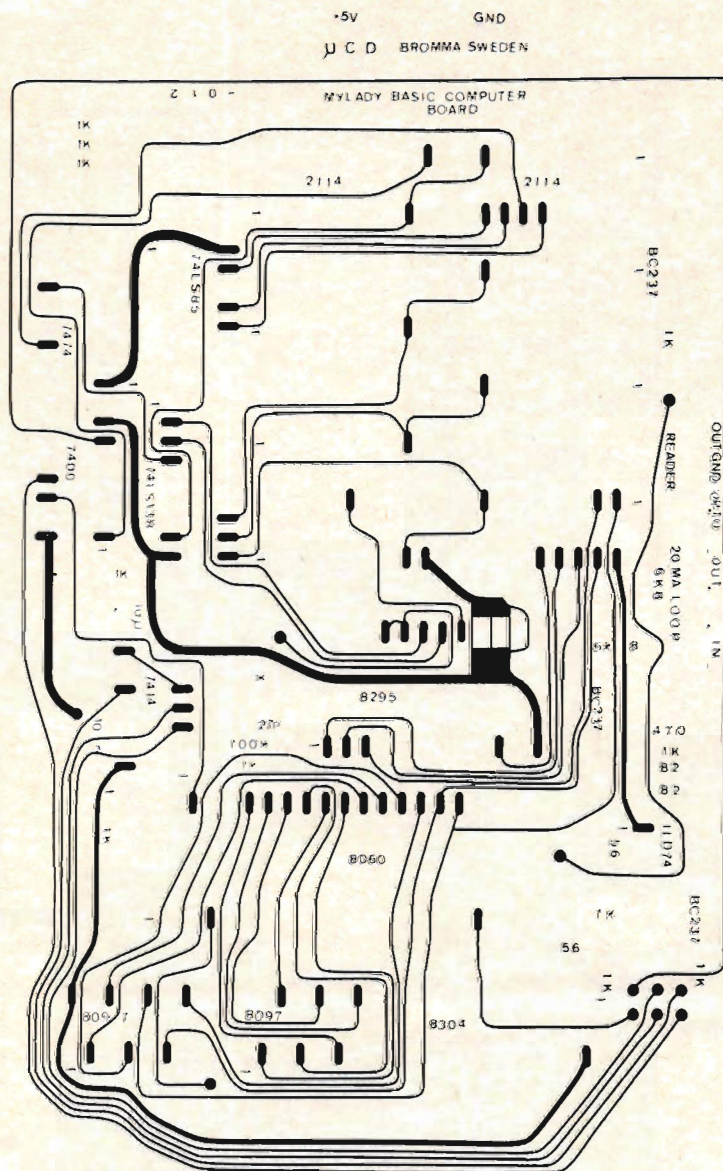


Fig 32:18. Basic-dator på ett kort. Översidans mönster. Skala 1:1.

Fig 32-19. Instruktioner för 6800-processorn (6808).

LSB MSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*	NOP (INH)	*	*	*	*	TAP (INH)	TPA (INH)	INX (INH)	DEX (INH)	CLV (INH)	SEV (INH)	CLC (INH)	SEC (INH)	CLI (INH)	SEI (INH)
1	SBA	CBA	*	*	*	*	TAB (INH)	TBA (INH)	*	DAA (INH)	*	ABA (INH)	*	*	*	*
2	BRA (REL)	*	BHI (REL)	BLS (REL)	BCC (REL)	BCS (REL)	BNE (REL)	BEQ (REL)	BVC (REL)	BVS (REL)	BPL (REL)	BMI (REL)	BGE (REL)	BLT (REL)	BGT (REL)	BLE (REL)
3	TSX (INH)	INS (INH)	PUL (A)	PUL (B)	DES (INH)	TXS (INH)	PSH (A)	PSH (B)	*	RTS (INH)	*	RTI (INH)	*	*	WAI (INH)	SWI (INH)
4	NEG (A)	*	*	COM (A)	LSR (A)	*	ROR (A)	ASR (A)	ASL (A)	ROL (A)	DEC (A)	*	INC (A)	TST (A)	*	CLR (A)
5	NEG (B)	*	*	COM (B)	LSR (B)	*	ROR (B)	ASR (B)	ASL (B)	ROL (B)	DEC (B)	*	INC (B)	TST (B)	*	CLR (B)
6	NEG (IND)	*	*	COM (IND)	LSR (IND)	*	ROR (IND)	ASR (IND)	ASL (IND)	ROL (IND)	DEC (IND)	*	INC (IND)	TST (IND)	JMP (IND)	CLR (IND)
7	NEG (EXT)	*	*	COM (EXT)	LSR (EXT)	*	ROR (EXT)	ASR (EXT)	ASL (EXT)	ROL (EXT)	DEC (EXT)	*	INC (EXT)	TST (EXT)	JMP (EXT)	CLR (EXT)
8	SUB (A) (IMM)	CMP (A) (IMM)	SBC (A) (IMM)	*	AND (A) (IMM)	BIT (A) (IMM)	LDA (A) (IMM)	*	EOR (A) (IMM)	ADC (A) (IMM)	ORA (A) (IMM)	ADD (A) (IMM)	CPX (A) (IMM)	BSR (REL)	LDS (IMM)	*
9	SUB (A) (DIR)	CMP (A) (DIR)	SBC (A) (DIR)	*	AND (A) (DIR)	BIT (A) (DIR)	LDA (A) (DIR)	STA (A) (DIR)	EOR (A) (DIR)	ADC (A) (DIR)	ORA (A) (DIR)	ADD (A) (DIR)	CPX (A) (DIR)	*	LDS (DIR)	STS (DIR)
A	SUB (A) (IND)	CMP (A) (IND)	SBC (A) (IND)	*	AND (A) (IND)	BIT (A) (IND)	LDA (A) (IND)	STA (A) (IND)	EOR (A) (IND)	ADC (A) (IND)	ORA (A) (IND)	ADD (A) (IND)	CPX (A) (IND)	JSR (IND)	LDS (IND)	STS (IND)
B	SUB (A) (EXT)	CMP (A) (EXT)	SBC (A) (EXT)	*	AND (A) (EXT)	BIT (A) (EXT)	LDA (A) (EXT)	STA (A) (EXT)	EOR (A) (EXT)	ADC (A) (EXT)	ORA (A) (EXT)	ADD (A) (EXT)	CPX (A) (EXT)	JSR (EXT)	LDS (EXT)	STS (EXT)
C	SUB (B) (IMM)	CMP (B) (IMM)	SBC (B) (IMM)	*	AND (B) (IMM)	BIT (B) (IMM)	LDA (B) (IMM)	*	EOR (B) (IMM)	ADC (B) (IMM)	ORA (B) (IMM)	ADD (B) (IMM)	*	*	LDX (IMM)	*
D	SUB (B) (DIR)	CMP (B) (DIR)	SBC (B) (DIR)	*	AND (B) (DIR)	BIT (B) (DIR)	LDA (B) (DIR)	STA (B) (DIR)	EOR (B) (DIR)	ADC (B) (DIR)	ORA (B) (DIR)	ADD (B) (DIR)	*	*	LDX (DIR) ^(B)	STX (DIR) ^(B)
E	SUB (B) (IND)	CMP (B) (IND)	SBC (B) (IND)	*	AND (B) (IND)	BIT (B) (IND)	LDA (B) (IND)	STA (B) (IND)	EOR (B) (IND)	ADC (B) (IND)	ORA (B) (IND)	ADD (B) (IND)	*	*	LDX (IND)	STX (IND)
F	SUB (B) (EXT)	CMP (B) (EXT)	SBC (B) (EXT)	*	AND (B) (EXT)	BIT (B) (EXT)	LDA (B) (EXT)	STA (B) (EXT)	EOR (B) (EXT)	ADC (B) (EXT)	ORA (B) (EXT)	ADD (B) (EXT)	*	*	LDX (EXT)	STX (EXT)

DIR = Direct Addressing Mode
EXT = Extended Addressing Mode
IMM = Immediate Addressing Mode

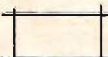
IND = Index Addressing Mode
INH = Inherent Addressing Mode
REL = Relative Addressing Mode

A = Accumulator A
B = Accumulator B

*Unimplemented Op Code

TABLE 1-4.1-1. M6800 Instruction Map

Hexkod					Binärkod					0	1	2	3	4	5	6	7
					b ₃	b ₂	b ₁	b ₀	b ₆	b ₅	b ₄						
									→0	→0	→0	0	0	0	0	1	1
												0	0	1	1	0	1
												1	1	0	1	0	1
0	0	0	0	0								NUL	DLE	SPACE	0	@	P
1	0	0	0	1								SOH	DC1	!	1	A	Q
2	0	0	1	0								STX	DC2	"	2	B	R
3	0	0	1	1								ETX	DC3	#	3	C	S
4	0	1	0	0								EOT	DC4	\$	4	D	T
5	0	1	0	1								ENQ	NAK	%	5	E	U
6	0	1	1	0								ACK	SYN	&	6	F	V
7	0	1	1	1								BEL	ETB	'	7	G	W
8	1	0	0	0								BS	CAN	(	8	H	X
9	1	0	0	1								HT	EM	)	9	I	Y
A	1	0	1	0								LF	SUB	*	:	J	Z
B	1	0	1	1								VT	ESC	+	;	K	[
C	1	1	0	0								FF	FS	,	<	L	\
D	1	1	0	1								CR	GS	-	=	M	]
E	1	1	1	0								SO	RS	.	>	N	^
F	1	1	1	1								SI	US	/	?	O	—
																	DEL

 Skrivbara tecken

 Kontrolltecken för yttre utrustning

 Kontrolltecken för skrivare

 Koder som genereras av tangentbord, men som ej skrivs ut

Fig 32:20. ASCII-koden. Den används vanligen i snittet mellan datorn och en alfanumerisk terminal.

FIGURE 14 – PROGRAMMING MODEL OF THE MICROPROCESSING UNIT

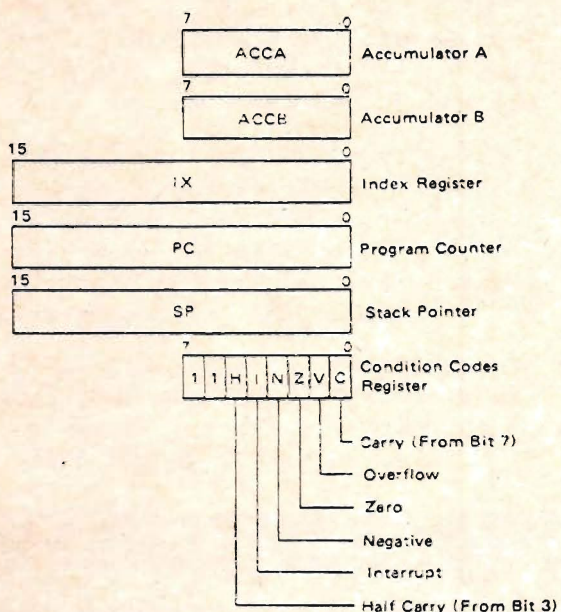


Fig 32:22. Programmeringsmodell för 6800.

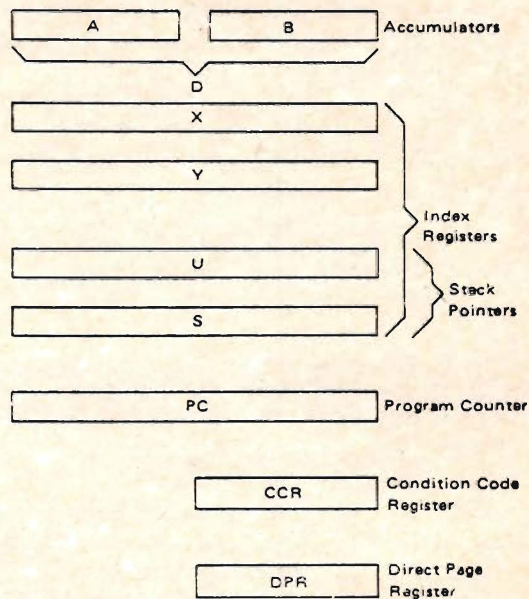


Fig 32:23. Programmeringsmodell för 6809.

Tabell 1
Sifferpresentationen i olika talsystem

Binär	oktal	decimal	hexadecimal	BCD
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	10	8	8	8
1001	11	9	9	9
1010	12	10	A	Ej tillåtet tal
1011	13	11	B	Ej tillåtet tal
1100	14	12	C	Ej tillåtet tal
1101	15	13	D	Ej tillåtet tal
1110	16	14	E	Ej tillåtet tal
1111	17	15	F	Ej tillåtet tal

Fig 32:21. Tabellen visar sifferpresentationen i olika talsystem.

MICROPOLIS™

När det gäller miniflexskivenheter av högsta kvalité och med mesta möjliga lagringskapacitet.

Finns bl.a. för ABC-80, S-100 bus och Multi-bus, 8 tums Winchesterdiskar med. 45 kontroller

45 M-byte lagringskapacitet (oforaterat) finns med kontroller för bl.a. S-100 bus.

HOBBY DATA

Europas Första Databutik

Box 2528
200 12 Malmö

Butik: Kronborgsvägen 8
Malmö
Tel: 040-910191, 913115
Telex: 32871

KOMPONENTERNA KÖPER DU FRÅN HN ELEKTRONIK.

KATALOG MOT 10:-

HN Elektronikkomponenter
Box 1004, 12610 Hägersten
Tfn 08/881600

**med nöjda kunder i
Sverige, Norge, Finland!**

EDITOR

EDITOR är en tidning för de som *ANVÄNDER* datorer. Det huvudsakliga innehållet handlar om mjukvara i alla former. *EDITOR* utkommer med 5 nummer per år.

I *EDITOR* presenteras de i *PROGRAMBANKEN (PB)* ingående programmen. Från *PB* kan man få program till självkostnadspris. Man kan också byta till sig nya program. *PB* är öppen för alla datorer.

Prenumeration på 5 nr. av *EDITOR* kostar endast 35 kr, som insättes på postgiro 1 44 71 - 7. Adressen är *EDITOR*, Fack, 412 96 GÖTEBORG.

PROGRAMBANKEN

Bli medlem i PD68

Om du använder mikrodator eller bara är intresserad av data, är Du välkommen att bli medlem i vår klubb.

Vi startade i januari 77 och har sedan dess ordnat med träffar där vi bjuder in olika föredragshållare, byter idéer och hjälper varandra med lösningar, samt tar del av de senaste nyheterna på området.

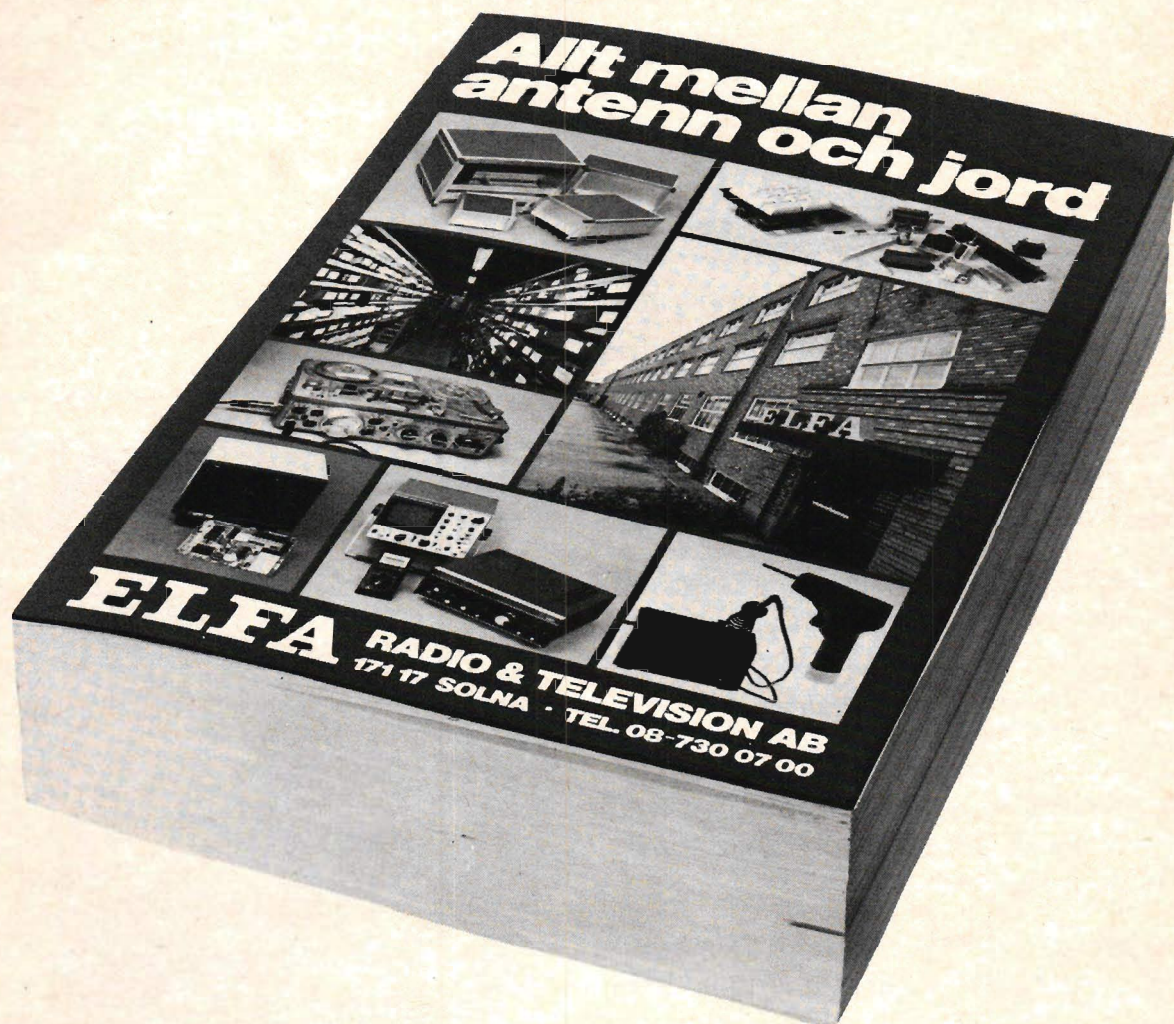
Vi ger dessutom ut en tidning som heter MPU-laren. Som medlem får du även vissa förmåner, exempelvis hjälp om Du kör fast, rabatt på vissa firmor och tips om nya kretsar, mässor o dyl.

För ytterligare information skriv till:

PD 68
Box 98
122 21 Enske 1

PD68 — klubben för er som sysslar med mikrodatorer.

Nu har **ELFA-katalogen** nr 29 kommit!



Elektronikbranschens oundgängliga uppslagsbok. Innehållsrikare än någonsin. Med mer än ett och ett halvt tusen sidor om mätinstrument, halvledare, kabel, mikrodatorer, strömställare, kontaktdon, chassystem, byggsatser, passiva komponenter, verktyg, fackböcker och mycket, mycket mer, totalt ca 17.000 artiklar.

ELFA-katalogen nr 29 distribueras gratis till industrin, institutioner och skolor. Trevlig läsning om Allt mellan antenn och jord!

ELFA
RADIO & TELEVISION AB
171 17 SOLNA
INDUSTRIVÄGEN 23 • 08/730 07 00

Du, elektronik- & bitpulare,

inte är det för inte, som
Du inte kan undvara
vårt katalogprogram!

Skriv eller ring (9-14)
idag och det föreligger
risk för att ett och annat
framledes kommer med
posten.

BHIAB electronics

Box 216 — 761 00 NORRTÄLJE

Tfn 0176/184 25

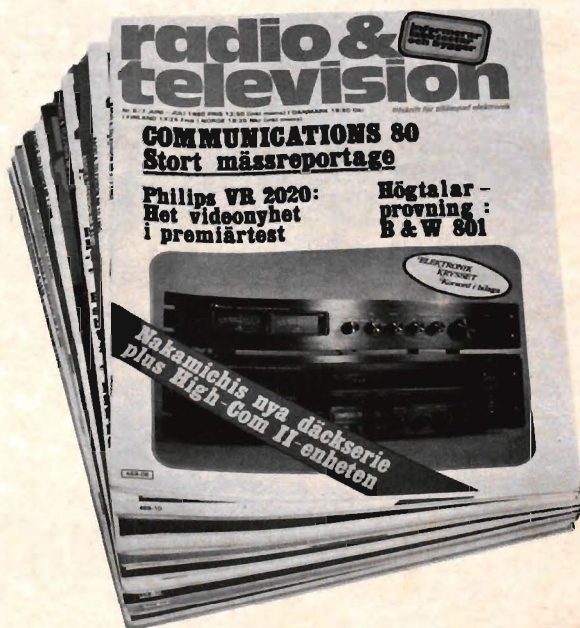
LÄS RADIO & TELEVISION!

Utvecklingen inom elektroniken går fort.
Om allt det nya, om genomgående och
grundliga tester, om självbyggen kan du
läsa i tidningen för tillämpad elektronik.
Och vad händer inom videon, musiktekniken
och på Hi-fi-sidan? I Radio & Television
hittar du allt nytt. Får mycket testat.
Testat på ett sätt som du har nytta av.
Tester av värde för dej.

Det lönar sig att läsa Radio & Television.
Och mest tjänar du på att prenumerera.

radio & television

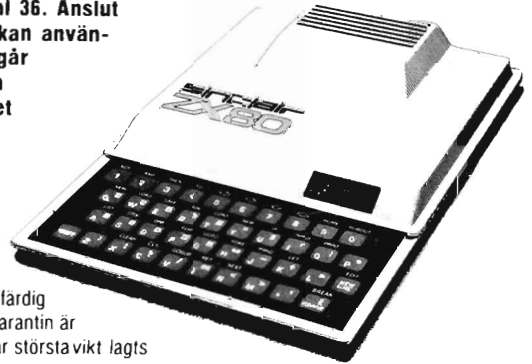
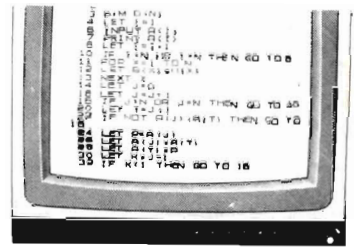
Ring 08/34 07 90 för prenumeration.



Nu har Din micro-dator kommit!

Nu har Du för första gången chansen att skaffa Dig en egen personlig dator och lära Dig programmering eller utnyttja Dina tidigare kunskaper. Din dator kan göra massor av saker. Lösa problem och mängder av uppgifter supersnabbt, rita diagram och figurer, spela spel, tippa tipsrader, lotto, V-65, etc. etc., men kanske det viktigaste: Utgöra Din inkörsport till datatekniken och ge dig viktiga kunskaper om framtiden.

Du ansluter datorn till Din TV:s antenningång och ställer in TV:n på kanal 36. Anslut nätadaptern i väggen och börja programmera. Din kassetbandspelare kan användas som programminne. Anslutningssladdar för TV och bandspelare ingår. Allt Du skriver och datorn tar fram redovisas med perfekt skärpa på Din TV-skärm. Dataspråket är BASIC — det vanligaste programmeringsspråket runt om i världen.



Sinclair ZX-80

Du minns säkert åren 1973–1974. De tidigare så exklusiva mini-räknarna blev med ens var mans egendom. Köpruschen var fenomenal på varuhusen. Sinclair var fabrikanter. Nu har dom gjort det igen — Sinclair ZX-80. ZX-80 kommer till Dig antingen färdig eller som enkel byggsats — Du spar några hundralappar och lär känna den från grunden. Garantin är densamma. Eftersom ZX-80 främst är tänkt för Dig som gör Din första datorerfarenhet, har störst värdigt lagts på att göra det enkelt för Dig.

BASICKURS, över 100 sidor med svensk översättning
DIREKTfunktion, endast en tangenttryckning för en hel instruktion som t.ex. RUN, PRINT, GOTO, CONTINUE. Alla instruktioner står skrivna på tangentbordet och skrivs i klartext på bildskärmen.

SYNTAXKONTROLL, ZX 80 kontrollerar att Dina instruktioner är riktiga och talar om när Du gjort fel.

EDITERING, Du kan var som helst i programmet ändra, byta ut eller lägga till instruktioner, bokstaver eller siffror.

DESSUTOM MINNESSÅL, genom direktfunktionen upptar instruktioner endast en enda byte i minnet, även om de har flera bokstaver som t.ex. CONTINUE.

SLUMPGENERATOR, för spel och tipsvarianter.
22 GRAFISKA SYMBOLER, för diagram och figurer.

Du kan beställa extra minnen och en ny ROM chip med flytande decimalkomma, 9 siffriga tal, trig- & log funktioner. Andra ytterligare tillbehör kommer successivt för Dig som samlar med utbildning är ZX 80 extra intressant. Kursverksamheten vid Umeå Universitet & Studieförbundet har skrivit en datorkurs byggd på ZX 80. Både elevhandbok & lärarhandledning finns. Idealt för skolor, studiecirklar & intern företagsutbildning samt självstudier.

HARD AND SOFTWARE

CPU: NEC780C—1, 3.25 MHz. Memory, 1K expandable to 16K. Screen, 24 lines x 32 characters. Bus: Edge connector, 44 lines. 37 från CPU, 0V, 5V, 9V. Clock: Ext. memory, indicator, två jord. Software, 4K ROM for BASIC, Editor & Operating system.

• Up to 26 string variables of any length. All can undergo relational tests. • Up to 26 single dimension arrays. • FOR/NEXT loops nested up to 26. • PEEK/POKE machine code instructions. • USA jump to machine language sub routine. • Key words occupy only a single byte.

Så tyckte tidningarna:

Radio & Television 8/80:

"Med andra ord är den ett intressant objekt för envar som vill lära sig BASIC på ett inspirerande sätt."

Teknik för Alla 8/80:

"Sinclair 8/80 är trots sitt pris en mycket kunnig dator. Att den kan programmeras i BASIC gör den lämplig som utbildningshjälpmedel, antingen för självstudier eller i skolor."

Godkänd av Televerket

Komplett byggsats ZX-80
inkl. S-märkt adapter &
kurs i BASIC-program-
mering. Inkl. moms.

1.395:-

monterad 1.595:-

Acorn Atom

"This is a very important machine" — omdömet gäller Acorn Atom och uttalades av den engelska datorutbildningstidningen "Educational Computing" efter test i nr 6/80. Atom är utvecklad av forskare knutna till det välkända Cambridge universitetet i England, för undervisning och hobbybruk.

ATOM är alltså tänkt för Dig som vill ha en avancerad dator för hobby och utbildning till ett rimligt pris, men utan att ge avkall på programmerings- och expansionsmöjligheter. Du kan programmera både i BASIC och ASSEMBLER. För avancerad utbildning och extrem snabbhet är detta mycket värdefullt. Mycket utförlig dokumentation medföljer — bl. a. en kombinerad manual och kurs i BASIC på över 200 sidor. ATOM använder Din vanliga TV som bildskärm men har även videoutgång för monitor. Grafiken har mycket hög upplösning och kan kompletteras med färgmodul. ATOM klarar file-hantering på Din kassetbandspelare. Timer-Funktionen, Rörlig Grafik och Ljudgenerering ger Dig möjlighet till roliga diagram, figurer och spel.

ATOM kan utrustas med RINGINTERFACE, med detta kan flera ATOMER kopplas ihop och utbyta data med varandra och spara på dyrbara tillbehör som floppy och skrivare. En unik egenskap för skolor och företag. (Jämför med time-sharing mot stordator).

ATOM är gjord så att Du kan börja med en basmodell (2K RAM—8KROM) och sedan komplettera med mera RAM- och ROM-minne. FÄRGMODUL, SKRIVARE, RINGINTERFACE, A/D—D/A OMVANDLARE M. M.

PRIS FRÅN
inkl. moms

2.875:-

Generalagent

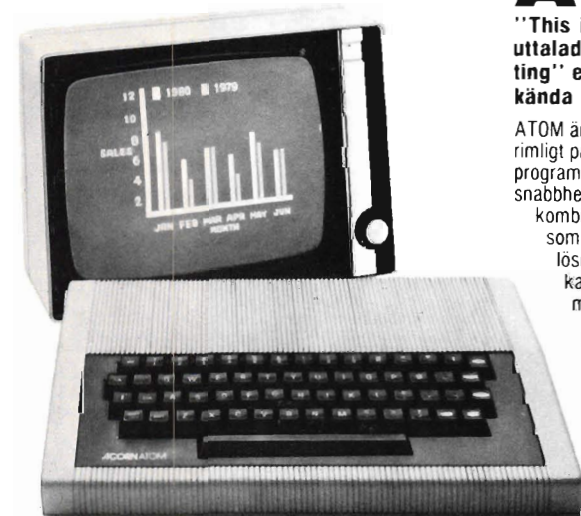
BECKMAN

Beckman Innovation AB

Telefon 08-39 04 00 Telex 10318

Gamla Dalarövägen 2 Box 7

S-122 21 Enskede SWEDEN



The ATOM software includes:

☆ 32 bit arithmetic ($\pm 2,000,000,000$) ☆ High speed execution ☆ 43 standard/extended BASIC commands ☆ Variable length strings (up to 256 characters) ☆ String manipulation functions ☆ 27 32-bit integer variables ☆ 27 additional arrays ☆ Random number function ☆ PUT and GET byte ☆ WAIT command for timing ☆ DO-UNTIL construction ☆ Logical operators (AND, OR, EX-OR) ☆ LINK to machine code routines ☆ Plot draw and move.

The ATOM hardware includes:

☆ Memory from 2K to 12K RAM on board (up to 35K in case) ☆ 8K to 16K ROM (two 4K additions) ☆ 6502 processor ☆ Video display allows high resolution (256 x 192) graphics and red, green and blue output ☆ Cassette interface — CUTS 300 baud ☆ Loudspeaker allows tone generation of any frequency ☆ Channel 35 UHF modulator output ☆ Bus output includes internal connections from Acorn Eurocard.

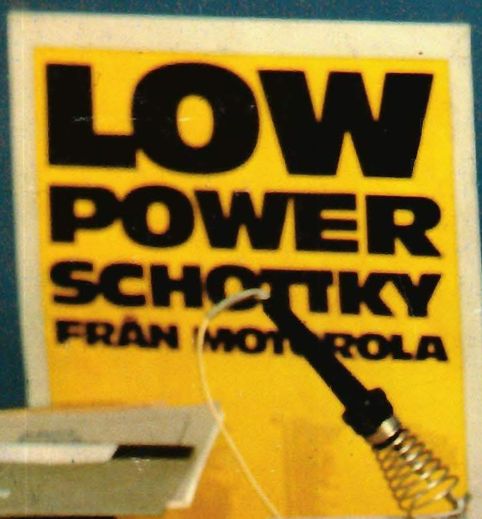
För dig som vill bygga datorer

har den här boken kommit till. Här finner du inte bara byggbeskrivningar för datorer och periferiutrustningar som dataterminaler, flexskivsystem. Du får dessutom veta hur en mikrodator fungerar, vad den arbetar i för språk och hur den kan programmeras. För att sedan komma vidare i din nya hobby får du tips om litteratur och dataklubbar. Väljer du i stället att köpa en dator har du ändå stor behållning av boken som bland annat presenterar ett urval av marknadens smådatorer, lämpliga för hobbybruk.

Cirka pris

38:—

(inkl. moms)



**radio &
television**

SPECIALTIDNINGSFÖRLAGET AB

ISBN 91-7274-108-2