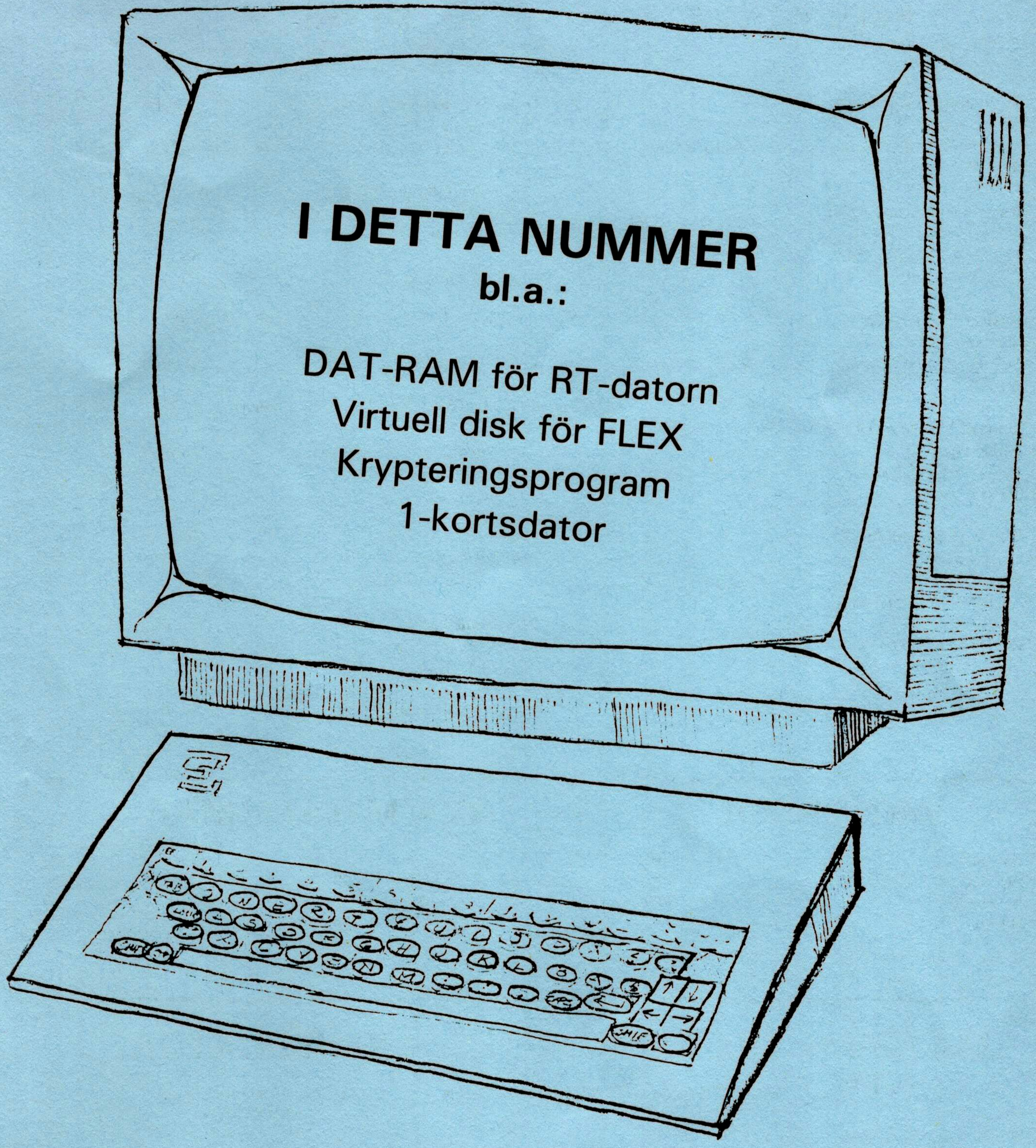
MPU-laren. Nr 2 1984

Tidskrift för Privatdataklubben PD68 Årgång 7

I DETTA NUMMER

bl.a.:

DAT-RAM för RT-datorn
Virtuell disk för FLEX
Krypteringsprogram
1-kortsdator



PRIVATDATAKLUBBEN

PD68

Box 1098, 122 21 ENSKEDE Pg 96 04 68-7

REDAKTION

Bernt Svensson (BS)
Östgötagatan 31 4 tr
116 25 STOCKHOLM ☎ 08-448864
(FLEX 2/9)

Roger Holmstrand (RH)
Rapsodivägen 168
142 00 TRÅNGSUND ☎ 08-7713852
(FLEX 9, OS9)

Gunnar Ejemoh (GE)
Altvägen 121
142 00 TRÅNGSUND ☎ 08-7715562
(FLEX 9)

Lennart Uusitalo (LUU)
Skrakvägen 20
141 72 HUDDINGE
(FLEX 9)

Sonny Taberman (SJT)
Ripstigen 5
171 73 SOLNA ☎ 08-853054
(FLEX 9, APPLE 2&3)

Lennart Holm (LH)
Ingelstavägen 11
710 15 VINTROSA
(FLEX2/9)

STYRELSE

Sven-Erik Ringström (SER) Ordf.
Högbyvägen 18
175 46 JÄRFÄLLA ☎ 0758-16228

Werner Gustafsson (WG) V.ordf.
Klövervallsvägen 63 C
183 44 TÄBY

Ingemar Skarpås (IS) Sekr.
Töjnavägen 42 A
191 44 SOLLENTUNA ☎ 08-353459

Leif Ulfström (LUL) Kassör
Fornhöjdsvägen 70
151 58 SÖDERTÄLJE ☎ 0755-18772

Tore Hall V.sekr.
Sävstaholmsvägen 209
125 36 ÄLVSJÖ

Tommy Bladh (TBL) Suppl.
Norrholmsvägen 106
132 00 SALTSJÖ-BOO

Åke Holm (ÅH) Suppl.
Långsjövägen 15 B
135 54 TYRESÖ

Roger Holmstrand (RH) Suppl.
Rapsodivägen 168
142 00 TRÅNGSUND

ANNONSER

Helsida	**500:-**
Halvsida	**300:-**
Kvartssida	**200:-**
Bilaga	**Enl. ö.k.**

Priserna gäller tryckfärdigt material. Medlemmar får sätta in privata radannonser gratis.

NYA MEDLEMMAR

Du som vill bli medlem i PD68 !
Sätt in 100:- på postgiro 96 04 68-7.
Glöm ej skriva namn, adress och tel.

MATERIAL TILL TIDNINGEN.

Alla bidrag STORA, små, korta eller **lååånga** är välkomna.
Om möjligt, skriv spalten med 40 kolumner !
Ni som har möjlighet bör skicka materialet på flexskiva.
Format: Enkel/dubbelsidigt, enkel packningstäthet och 35/40 spår.
Annat format enl. ö.k. Flexskivan returneras

KLUBBDATOR

Tel. nr. till klubbdatorn:
08-998716
300 baud fdx
OBS! Kräver password.

Ordföranden har ordet.

Nu när "den blomstertid nu kommer" har klingat ut, har många av er kanske lagt datorn på hyllan. För att bidra till semesterlitteraturen försöker vi i alla fall få ut ett nummer av MPU-laren. Materialet strömmar inte in, därav ett något magert exemplar. Dem av er som räknar in er bland så kallade "hackers", har ni sett hur uttrycket börjat användas i platsannonser? "Överårig hacker sökes till.....", "för en hacker mellan 25-35 är det här drömjobbet...! Ett tredje exempel ur DN är, "Vi kan tänka oss antingen hackertypen eller dig med mera normal programmeringsbakgrund till.....!" Det är tydligen marknad för den som vill byta en hobby till ett yrke. Fast då blir det kanske inte lika roligt längre? Mycket av det som skrivits i MPU-laren den senaste tiden har handlat om datorn som färdig maskin med operativsystem, massminne mm. Jag tycker att det skulle vara roligt att se några exempel där ni använder mikrodatorn som komponent. Det kan vara någon nyttig eller onödig applikation, som mikrostyrd myggjagare eller energispargris mm. Fatta pennan, luren eller terminalen och berätta!! I övrigt önskar jag er en trevlig sommar, och ser fram emot en pularhöst med mycket nyheter på 68 fronten.

SER

Innehållsförteckning

68008	4	Klubbdatorn	22
68k-projektet	4	Baudrategenerator	27
Ändring av RT-datorn	6	Candela	27
VME-bussen	13	Division med precision	27
Krypteringsprogram	13	OS-9 assembler	28
Programinitiering PIA	16	Virtuell disk för FLEX	30
Styrmodul m. mikroprocessor	17	64k dynamiskt RAM	32
Annonser	21	FSK-modulator	34

68008

Allt mer hör vi talas om 16-bitsdatorerna som skall ersätta 8-bitarna.
Hur fungerar då en 16-bits dator och vad är det för speciellt med den ?

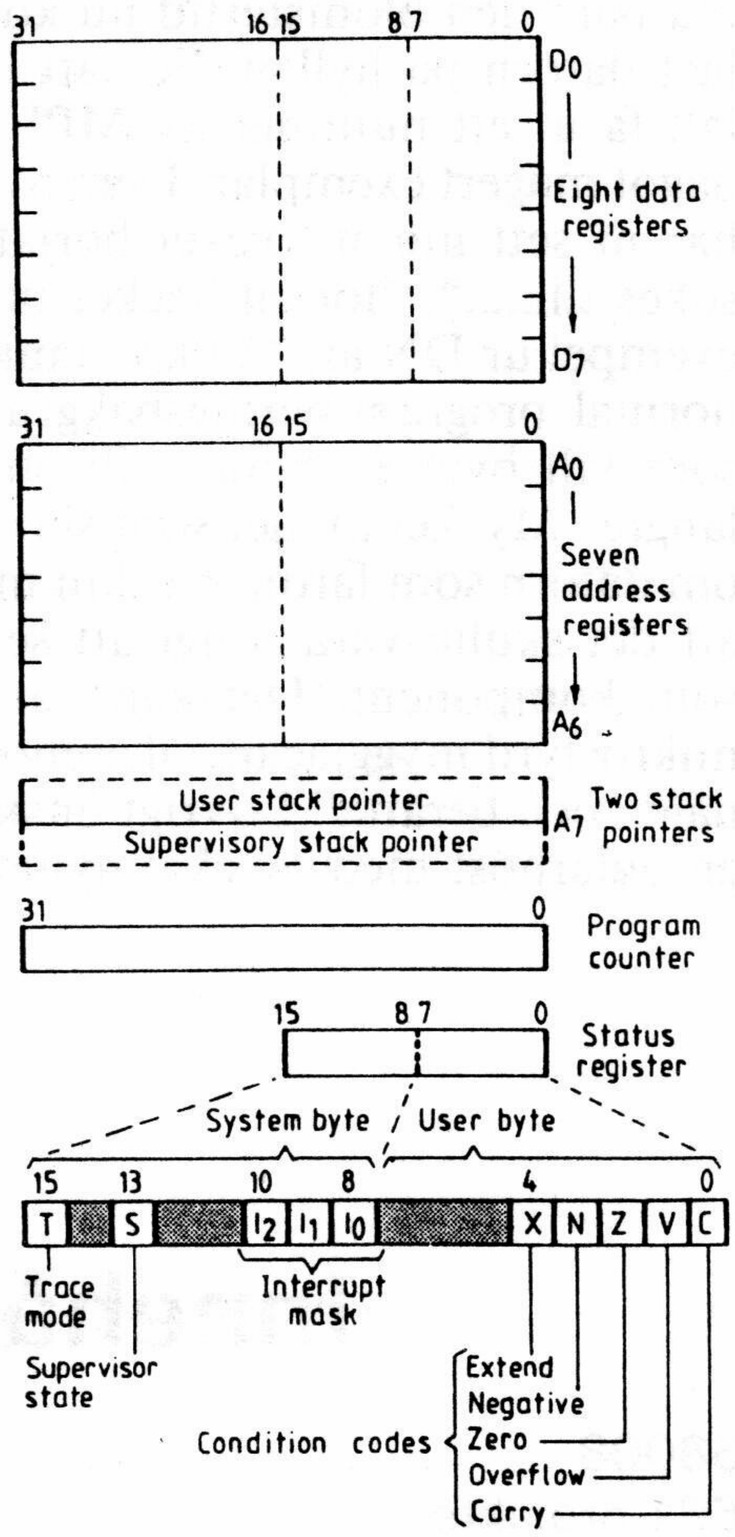
I den här artikeln skall jag ta upp den processor som troligtvis blir den första 68000-processor vi knåpar ihop en konstruktion med, nämligen 68008. Det är den processor som bl.a finns i Sinclairs nya dator Quantum Leap (QL). 68008 har bara 8 bitars databuss vilket gör den lämpad för en MPU-lare då man på ett enklare och mer ekonomiskt sätt kan konstruera ett minimi-system.
Jämfört med t.ex. 6800 har 68008 fyra extra adressledningar vilket ger möjlighet till direkt adressering av mer än en megabyte minne.
Visst finns det 6800 eller 6809-system som har mera minne än de 64 k som adressledningarna tillåter, men då har man bara möjlighet att välja in och manipulera ett 64 k block i taget vilket många gånger kan vara irriterande och tidsödande.

Genom att bl. a. reducera databussen till 8 bitar och eliminera adressbit A20-A23 har man lyckats klämma in innehållet i en 48-pinnars kapsel.
Ett viktigt plus för 68008 är, förutom att den är en ekonomisk 68000-variant, också att alla program skrivna för 68008 går att köra direkt i 68000,68010 och 68020 utan några som helst modifieringar.
Nåväl, men hur ser den då ut invändigt ?

68008 har ett stort antal allmänna 32-bitars register. (Det har varit lite diskussion om huruvida en dator med 32-bitars registerarkitektur skall kallas 16-bits eller 32-bits dator. Man har dock enats om att låta den falla inom kategorin 16-bits dator).

Den är utrustad med 8 st allmänna 32-bits dataregister,D0-D7, (ej att förväxla med databit D0-D7). Dessa kan användas för byte (8-bit), ord (16-bit), eller "långa ord" (32-bit) operationer.
Vidare finns sju st adressregister, A0-A6, och två stackpekarregister. Adressregistren kan också användas som stackpekare. Samtliga 17 register kan användas som indexregister.
Med 17 st 32-bitars register som kan manipuleras på detta sätt, samt i de flesta fall användas i byte, ord eller "långt ord"-mod, fås en väldig flexibilitet och snabbhet. Detta genom att många parametrar kan ligga kvar i ett register utan att kontinuerligt behöva skickas mellan CPU och externt minne.
Ett flertal instruktioner och adresseringsmoder kan användas tillsammans med programräknaren. (Även denna är av 32-bits typ).

Detta gör det möjligt att skriva positionsoberoende program som fungerar oberoende av var koden finns i minnet.



68008 har ett 16-bitars statusregister som består av två delar : en användardel och en systemdel, båda med åtta bitar.
Användardelen har de normala statusflaggorna som carry, overflow, etc. medan systemdelen har bitar som t.ex. interrupt mask, supervisor state och trace mode och används bl.a. vid "avlusning" av program.

68008 har 56 standardinstruktioner.
Till och med den 10 år äldre 6800 har fler, men pga. 68008:s flexibilitet blir antalet instruktionskombinationer flera tusen. Filosofin bakom 68000-serien har varit ett litet antal flexibla instruktioner, lätta att komma ihåg.
Man har också lyckats utrusta 68008 med 14 st kraftfulla adresseringsmoder, vilket resulterar i väldigt snabba, korta och kraftfulla program.

Om man har en kraftfull processor vill man kanske, förutom RAM-kretsar, kanske haka på en del kringkretsar. Då 68000-serien kan arbeta med klockperioder så korta som 80 nS kan det vara marigt (och dyrt) att hitta tillräckligt snabba kretsar för detta. För att klara detta problem har 68000-serien utrustats med s.k. asynkron buss.
Detta innebär att processorn har en signalingång som här benämns DTACK. DTACK är helt enkelt en ingång där den yttre kretsen signalerar när den är klar. Först därefter fortsätter processorn sin programexekvering. Härigenom kan processorn snurra med den maximala hastighet som den aktuella yttre kretsen har, vilket naturligtvis ökar den totala snabbheten hos systemet.

Gunnar Ejemoh

✦

Intressant läsning

Så här i sista stund vill jag nämna några intressanta artiklar ur BYTE June 1984.

An ADA Language Primer, Part 1. sid 131. Du som från början vill hålla dej informerad om det Pascal-liknande programmeringsspråket ADA får här en viss introduktion.

Build a Printer Buffer. sid 142. Beskrivning över ett bygge av en printerbuffer med 64K RAM. Ett virprojekt som använder sej av en Z80 som styrprocessor för en parallell dator-port till en parallell printer-port. 128 bytes programvara i PROM.

Kermit: A File Transfer Protocol for Universities, Part 1: Design Considerations and Specifications. sid 255. Beskrivning av ett kommunikationsprotokoll som fungerar bra på alla storlekar av datorer. Håller redan på att bli ett populärt protokoll världen över, och har många fördelar som pekar på att det åtminstone kan bli en defactostandard.

Another Look at CP/M-80 C Compilers. sid 303. En artikel av stort intresse för dom som är intresserade av klubbens 68000-projekt. Fastän artikel här beskriver några C kompilatorer för CPM, så är en av de resencerade kompilatorerna Whitesmiths C. Eftersom denna kompilator i IDRIS-utförande och för 68000, ingår som standard i 68K-projektet kan man åtminstone få en uppfattning om denna kompilators företräden.

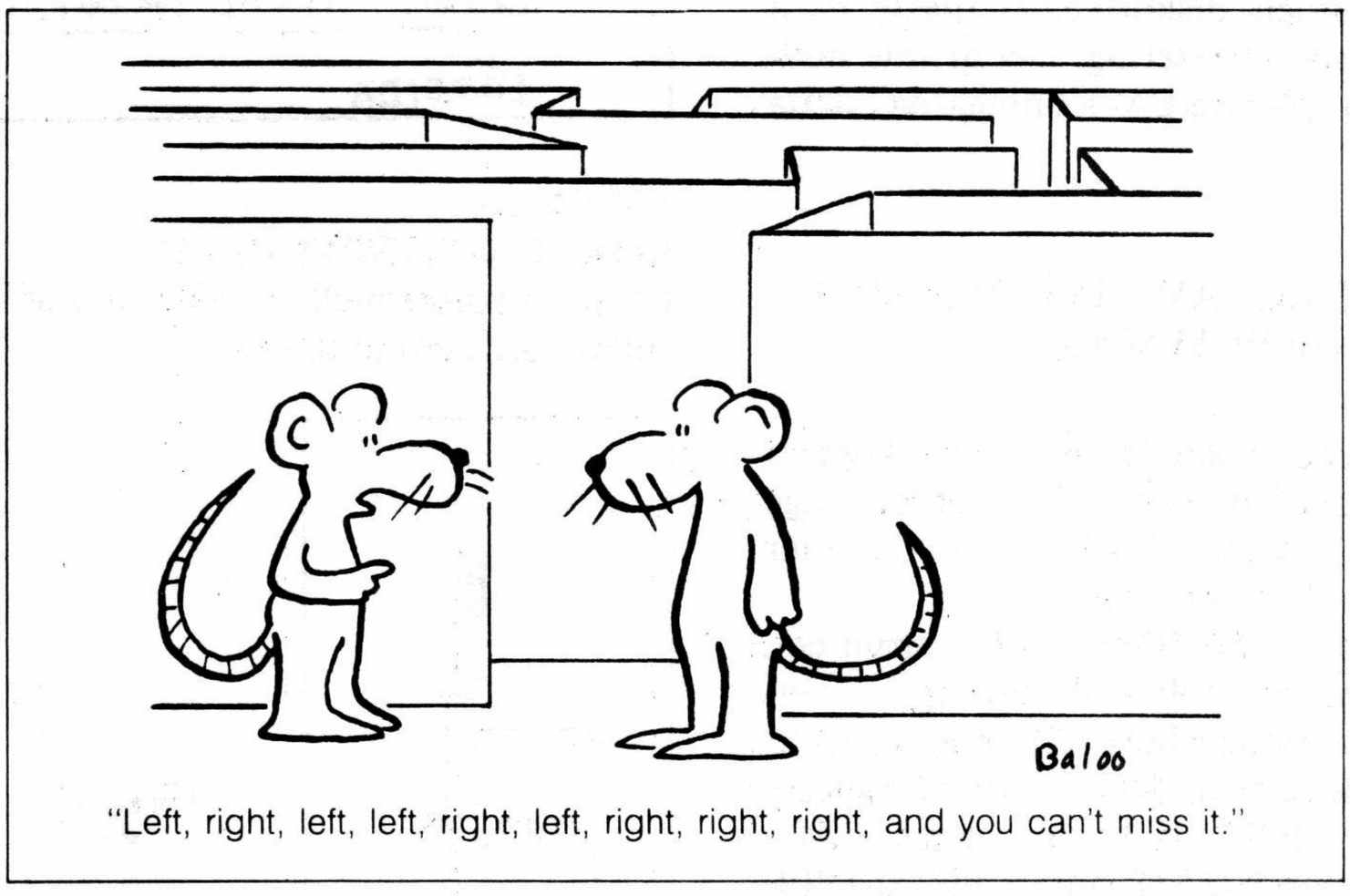
I.S.

✦

Vad händer med 68k-projektet.

Ledsamt nog drar det hela ut på tiden. Arbetet har enligt uppgift framskridit till första filmen på kortlayouten. Man beräknar att prov och revidering av denna ska kunna vara klar på sensommaren. Någon definitiv leveranstid har inte kunnat utlovas.
Håll ut, när det hela kommer är det en godsak.

SER



"Left, right, left, left, right, left, right, right, right, and you can't miss it."

DATAMATION

RT-DATORN
ÄNDRING AV CPU-KORTET
CA-6819

Ändring nr 2 och nr 3 gäller endast för 6809-läge.

1. 2716 EPROM

Modifiering av ledningsmönstret till Eprom-socklarna IC26-29, IC31-34 så att Eprom typ 2716 kan användas istället för 2708. Ram typ 6116 (2k x 8) som är pinkompatibelt med 2716 kan användas som IC32 el IC34. Då böjs pin 21 WE utanför sockeln och kopplas till R/W.

2. AVKODNING AV EPROM

Hela adressområdet $E100-$FFFF adress- eras inom CPU-kortet. Avkodningen är ändrad så att 6809 kan använda området $E100-$EFFF. IC45 byts ut till en 74LS- 138, några ben böjs utanför sockeln och en del ledningar löds in. Ändringen är utförd utan åverkan på mönsterkortet

3. EXTENDED ADRESSERING

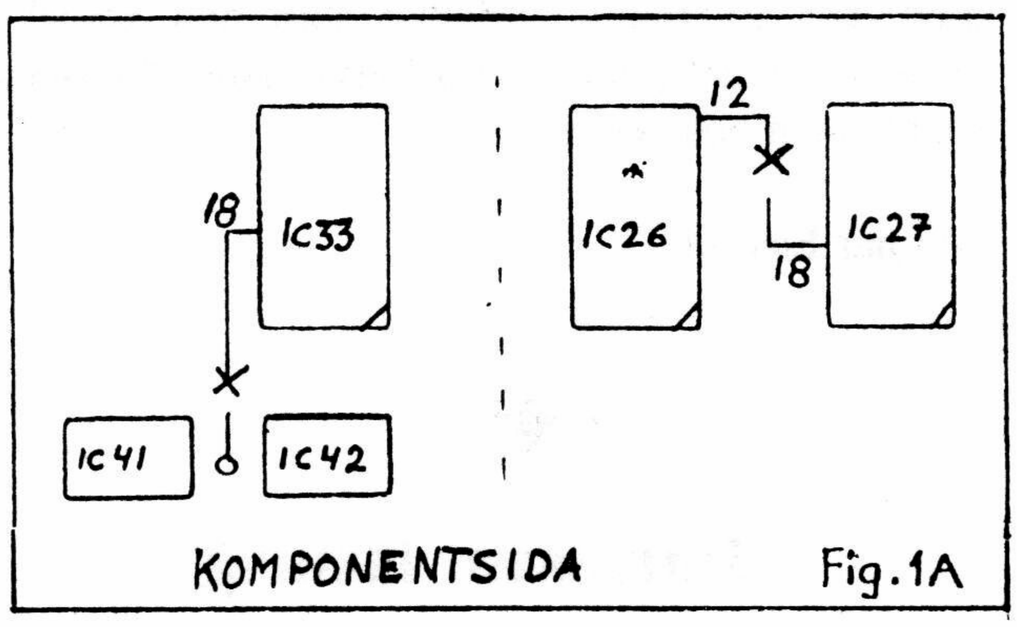
Fyra extra adressledningar ökar adress- området till 16 x 64k, ca 1 Mbyte. Detta är teoretiskt då allt minne som avkodas internt på CPU-kortet återfinns i varje 64k block.
För att få dessa fyra extra adress- ledningar låter man dom fyra MS adress- ledningarna dvs A12-15 adressera ett så kallat DAT-RAM. Data ut från DAT-RAM väljer vilket 64k block och 4k page inom respektive block som mappas in. DAT-RAM transformerar A12-15 från 6809 till A12' - A15' (page) ut på busskon- takten och till EA16 - EA19 (Extended Adress) som är kopplade till stift NOT, F,E,D,C respektive på busskontakten. En kort rutin som ingår i RESTART- rutinen (coldstart) sätter upp DAT-RAM så att extended adress är $F (EA16-19 = höga) och varje page är samma som 6809 adresserar.
Om man inte använder fler minneskort utöver ett 48k fungerar allt som förut. Om tex två 48k kort ska användas,det ena som användarminne det andra som en virtuell diskdrive så måste dom ändras för Extended Adressering. Det är inte nödvändigt att avkoda alla fyra EA- ledningarna i detta fall.

1.1 MODIFIERING AV LEDNINGSMÖNSTRET FÖR 2716 EPROM Se fig 1.

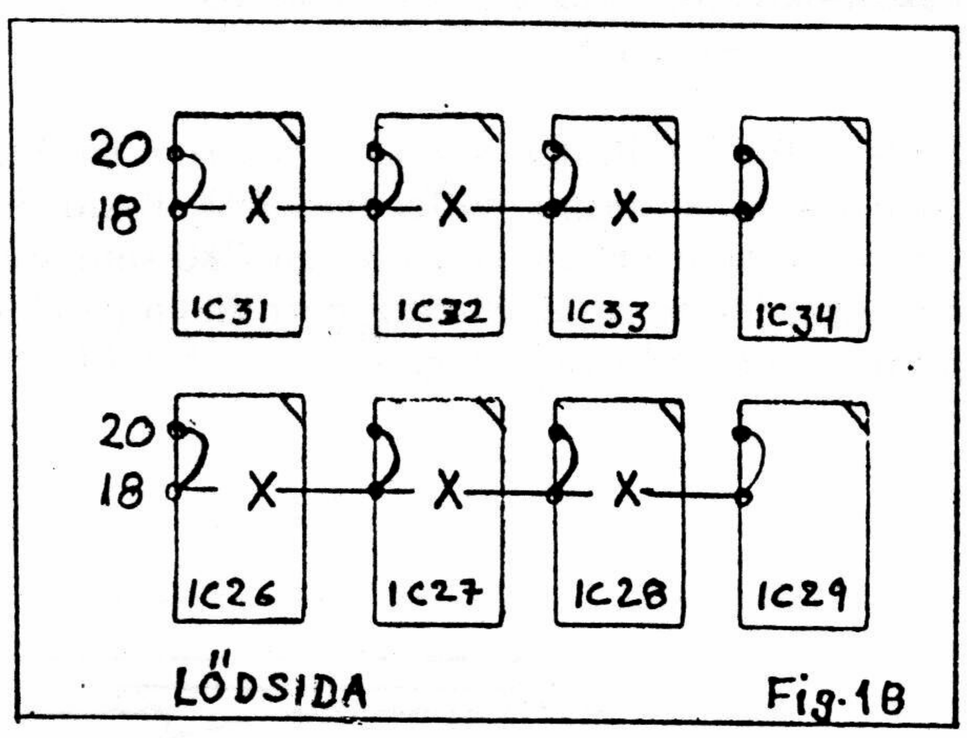
OBS, innan Du börjar ändra cpu-kortet kopiera över C-BUG och ev T-BUG till 2716 eprom. Lägg in patchen för DAT-RAM i C-BUG om Du tänker koppla in Extended Adressering.
Det går att använda 6808/09 med 2716 eprom och 6116 ram om man gör följande ändring av ledningsmönstret och kopplar bort A10 från IC43 pin 2,14 och kopplar dessa ben till O2. T-BUG placeras i IC 32, C-BUG i IC29. IC34 får adress $E800 -EFFF i 6808 läge och IC27 får adress $F000-F7FF i 6809 läge, övriga socklar är ej avkodade. Om ram används obs att pin 21 måste kopplas till R/W.

Pin	2716	2708	6116
18	CE	GND	A10
19	A10	+12V	A10
20	OE	CS	OE
21	+5V	-5V	WE

1.2 PIN 18.
KOMPONENTSIDA Fig 1A: Kapa mönstret mellan IC33 pin 18 och jord, mellan IC27 pin 18 och IC26 pin 12



LÖDSIDA Fig 1B:
Kapa mönstret som förbinder pin 18 under varje sockel IC26-29 IC31-34, 6 ställen. Bygla pin 18 till pin 20 vid samtliga socklar, 8 ställen.



1.3 PIN 19.
KOMPONENTSIDA Fig 1C:
Kapa mönstret mellan hålet nära IC29 pin 1 och stiftet på busskontakten.

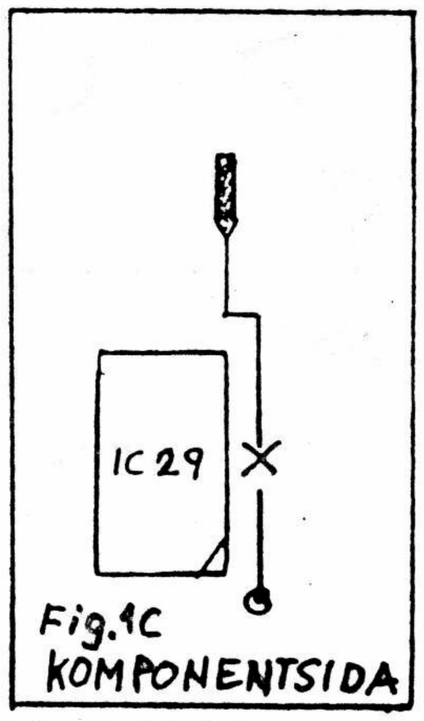


Fig. 1

MOD. AV LEDNINGSMÖNSTRET FÖR 2716 EPROM

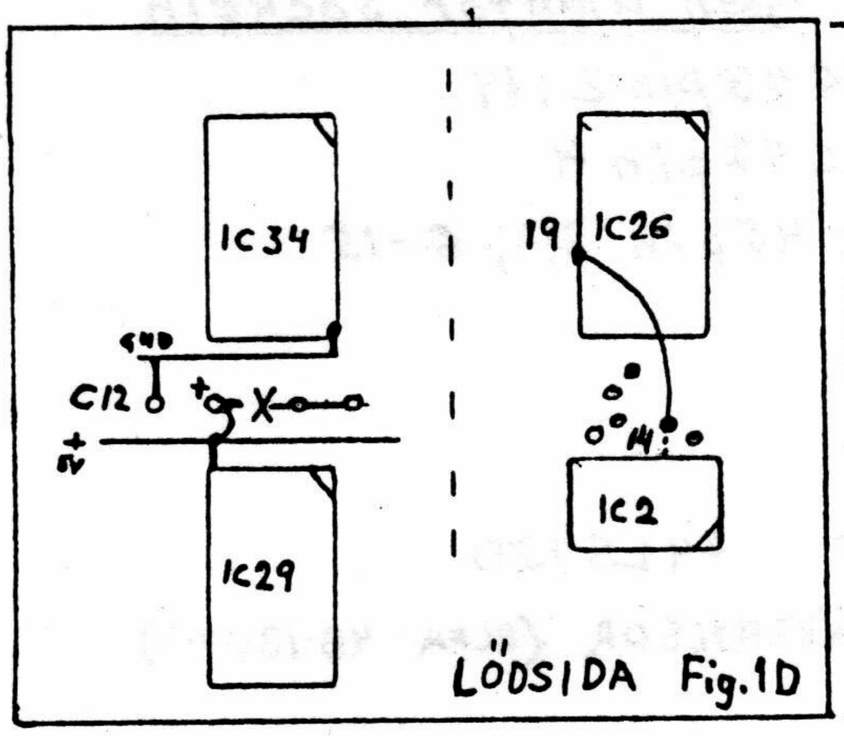
LÖDSIDA Fig 1D:

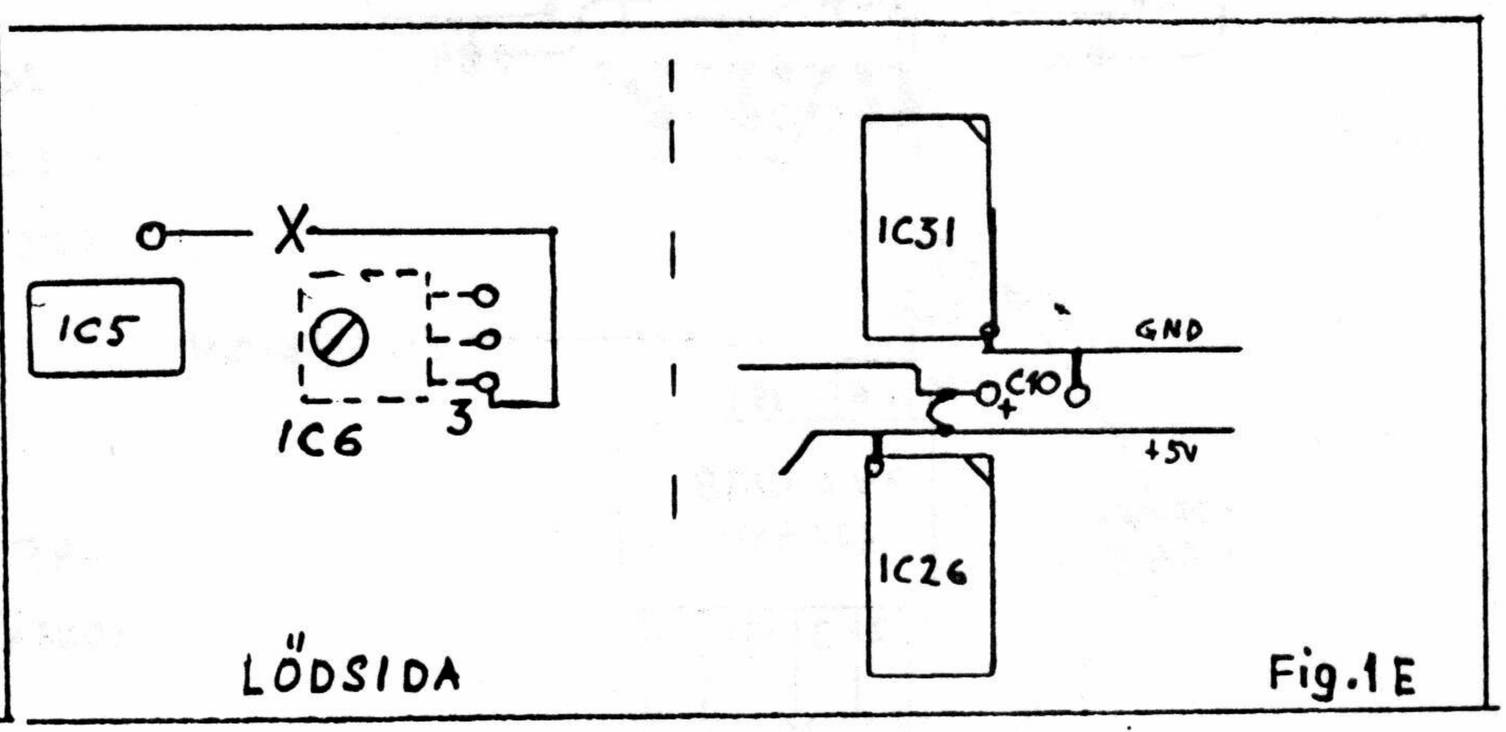
Kapa mönstret vid +benet på C12, bygla + på C12 till +5v. Förbind IC26 pin 19 till IC2 pin 14.

1.4 PIN 21.

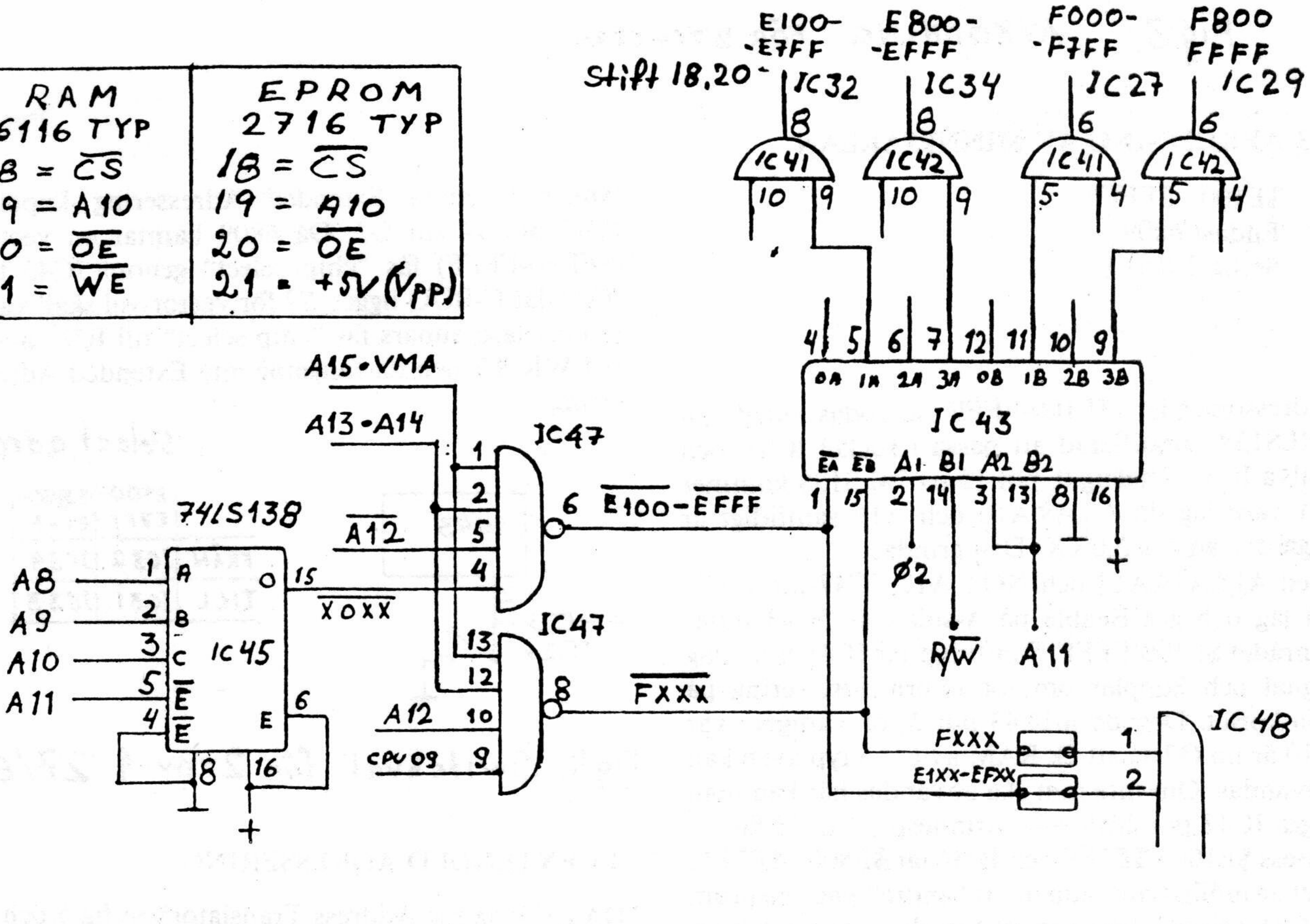
LÖDSIDA Fig 1E:

Kapa mönstret som går från IC6 ben 3 (-5v). Vänd på C10 och bygla +benet till +5 volt.





RAM 6116 TYP	EPROM 2716 TYP
18 = $\overline{CS}$	18 = $\overline{CS}$
19 = A10	19 = A10
20 = $\overline{OE}$	20 = $\overline{OE}$
21 = $\overline{WE}$	21 = +5V (V_{PP})

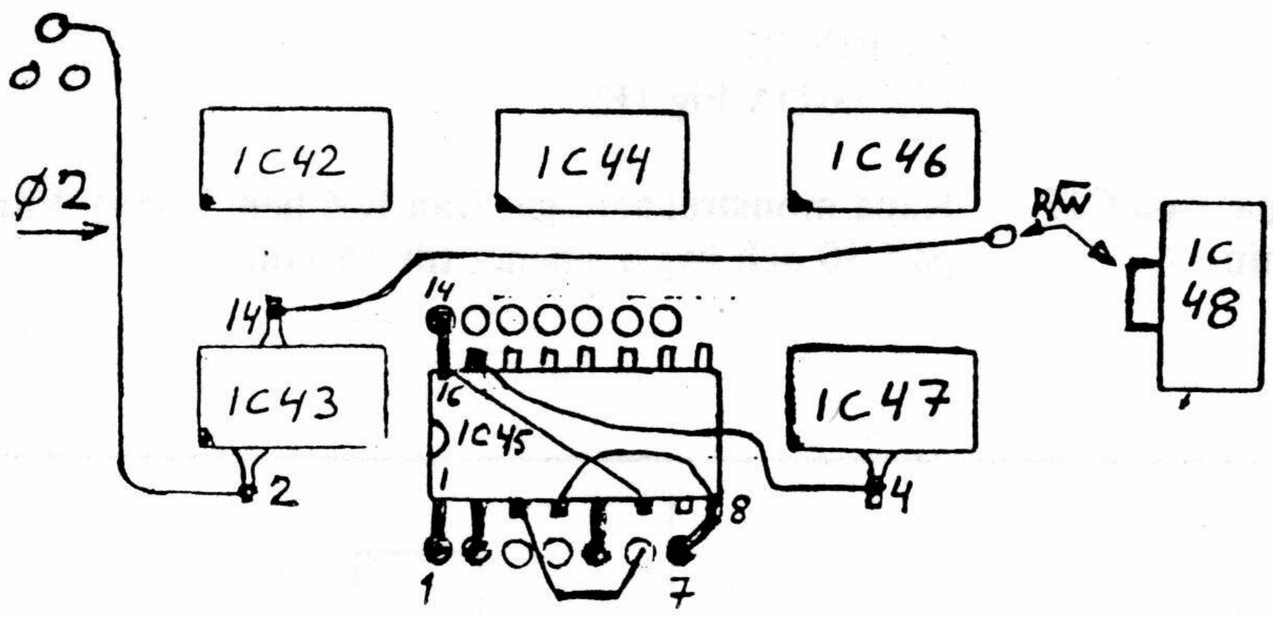


IC 32 - E100 , IC 34 - E800 , EPROM el. RAM (obs ben 21=RW)

IC 27 - F000 , IC 29 - F800 , EPROM

IC 31, IC 33, Se text. IC 26, IC 28 Ej avkodade

Fig 2 PRINCIPSCHEMA AVKODNING

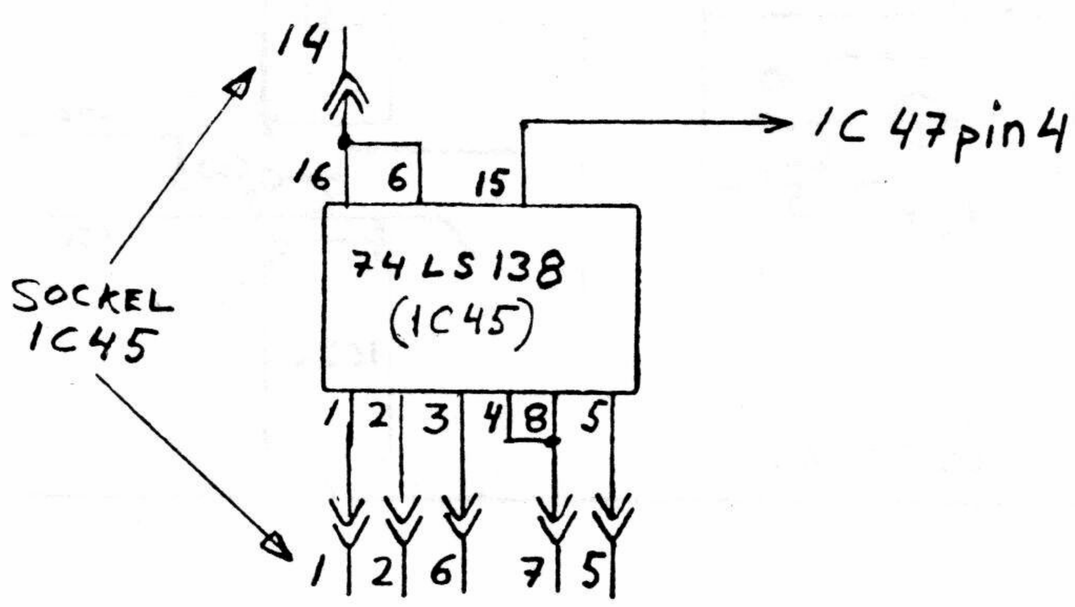


Ben utanför sockeln

IC 43 pin 2, 14

IC 47 pin 4

IC 45 pin 3, 4, 6-15



IC 45 74 LS 138

KONTAKTHYLSOR (ELFA 48-1307-7)

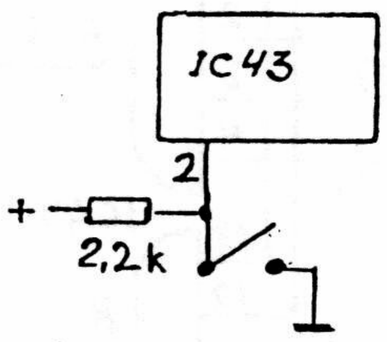
Fig 3 AVKODNING FÖR 2716 EPROM

2.1 AVKODNING AV MINNESAREAN

$E100 - $FFFF.
Endast 6809.
Se fig 2 och 3.

Adressområdet $E100-EFFF avkodas med en 74LS138, modifierad att passa i sockel IC45, och halva IC47. Utgång 0, pin 15 på 74LS138 kommer att vara låg då A8,A9,A10 och A11 samtidigt är låga dvs adress X0XX. Den grindas med A15,A14,A13 och NOT A12, IC47 pin 6 blir då låg och ger Enable på A-sidan av IC43 inom området $E100-EFFF. Samtidigt får IC48 pin 2 låg signal och kopplar om för intern adressering på cpu-kortet. Decodern IC43 pin 2 som tidigare var A10 är nu O2 så att 2k RAM av 6116 typ även kan användas. Om inte ram ska användas här kan man lägga IC43 pin 2 till +5v. Minnesplats IC32 får nu adress $E100 - $E7FF och IC34 får $E800 - $EFFF. Det är möjligt att välja ut ett "andra" par av eprom i IC31 och IC33 genom att koppla in en switch till IC43 pin 2 se fig 4. Det fungerar förstås bara på eprom då ram måste ha O2 med.

Adressområdet $F000-FFFF avkodas på liknande sätt i 2k block dvs A10 tas bort från IC43 pin 14. Där inkopplas R/W signalen. Det innebär att "chip select" till IC27 och IC29 är aktiv då 6809 utför READ (R/W = Hög). Vid WRITE kopplas IC27 och IC29 bort. Detta är till för att undvika dubbeladressering av eprom och DAT-RAM. DAT-RAM erhåller ju "chip select" på adress $FFF0-FFFF då R/W signalen är låg (Write Only). Om man inte tänker använda Extended Adressering kopplas IC43 pin 14 till O2. Då 6809 hämtar en vector ($FFF0-FFFF) fås "chip select" genom IC43 till IC29 där C-BUG ligger. S2 för vector-val skall vara i öppet läge annars fås "chip select" till IC29 även vid WRITE och då fungerar inte Extended Adressering.



Select adress

	E100-E7FF	E800-EFFF
FRÅN	IC32	IC34
TILL	IC31	IC33

Fig 4. Switchval för 2-av-4 27/6 Eprom

3.1 EXTENDED ADRESSERING

DAT (Dynamic Address Translator) Se fig 5 och 6

Som DAT-RAM används två 74S189 (64 bit 16x4 Bipolar Ram) med avkodningslogik kopplade på ett Veroboard. Kortet an- sluter till cpu-kortet i IC3's sockel där adressledningarna A12-A15, R/W, O2 hämtas. Från sockel IC30 tas A0-A3 och databussen. En select signal FFXX från IC44 pin 8 kopplas med en ledning till kortet. Anslutningen till IC3's sockel tillverkas av en 16 pins virsockel med långa ben. Virsockelns pinnar sticks genom passande hål på kortet och löds på undersidan. Pin 4,6,10,12 klipps av jäms med kortet, dom andra

pinnarna löds fast på en 16 pins komponent-adapter som sticks i sockeln IC3 på cpu-kortet. Kretsen IC3 monteras i vir- sockeln, kortet kommer då att ligga över en del kretsar på cpu-kortet men om det placeras i busskontakten närmast periferikorten så finns det utrymme.

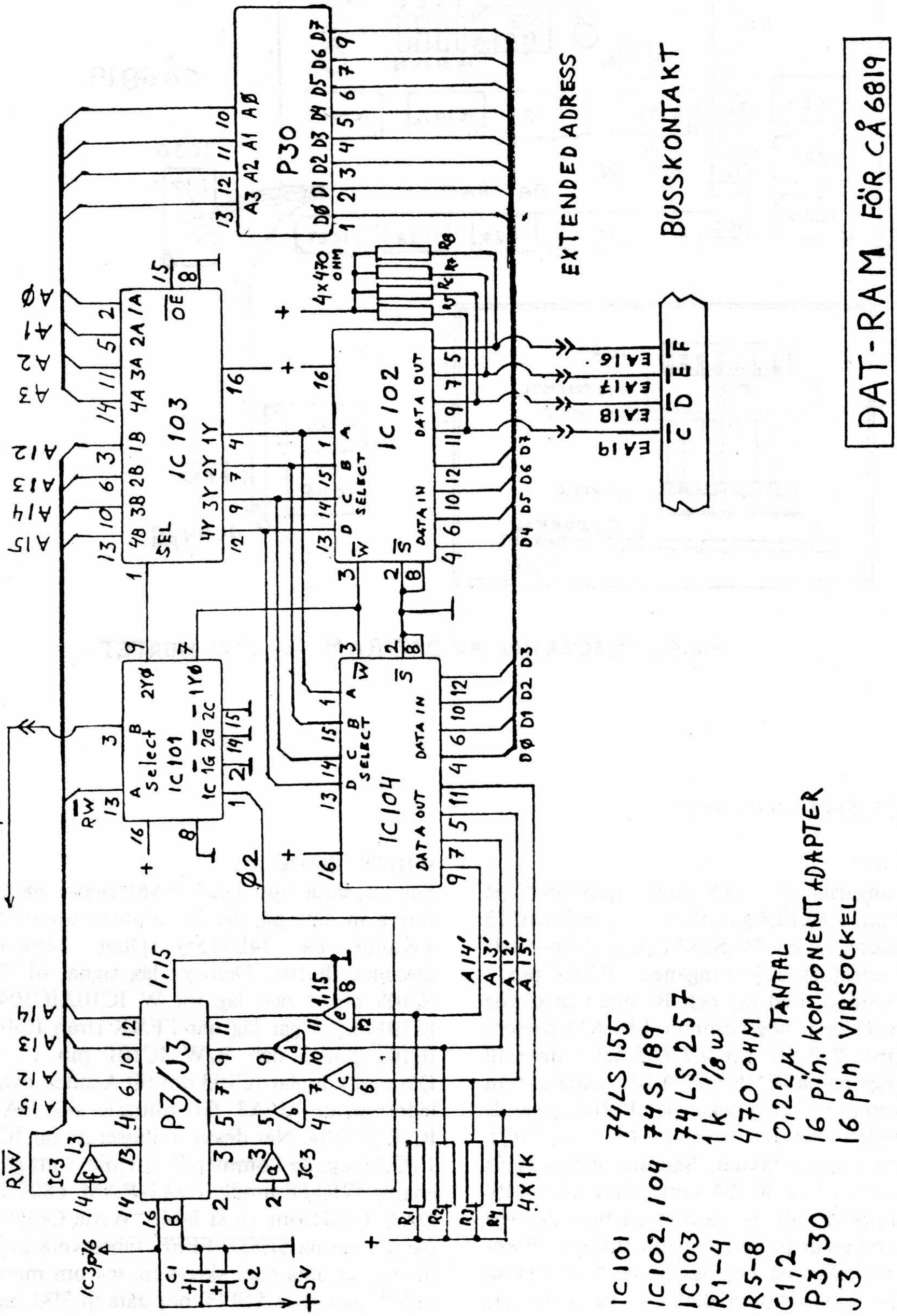


Fig 5.

Försening

Boken, An Assembly Language Course, är försenad från USA. De som beställt den får den så fort den kommer. Om någon ytterligare önskar beställa den går det bra, jag tog hem några extra exemplar.
I.S.

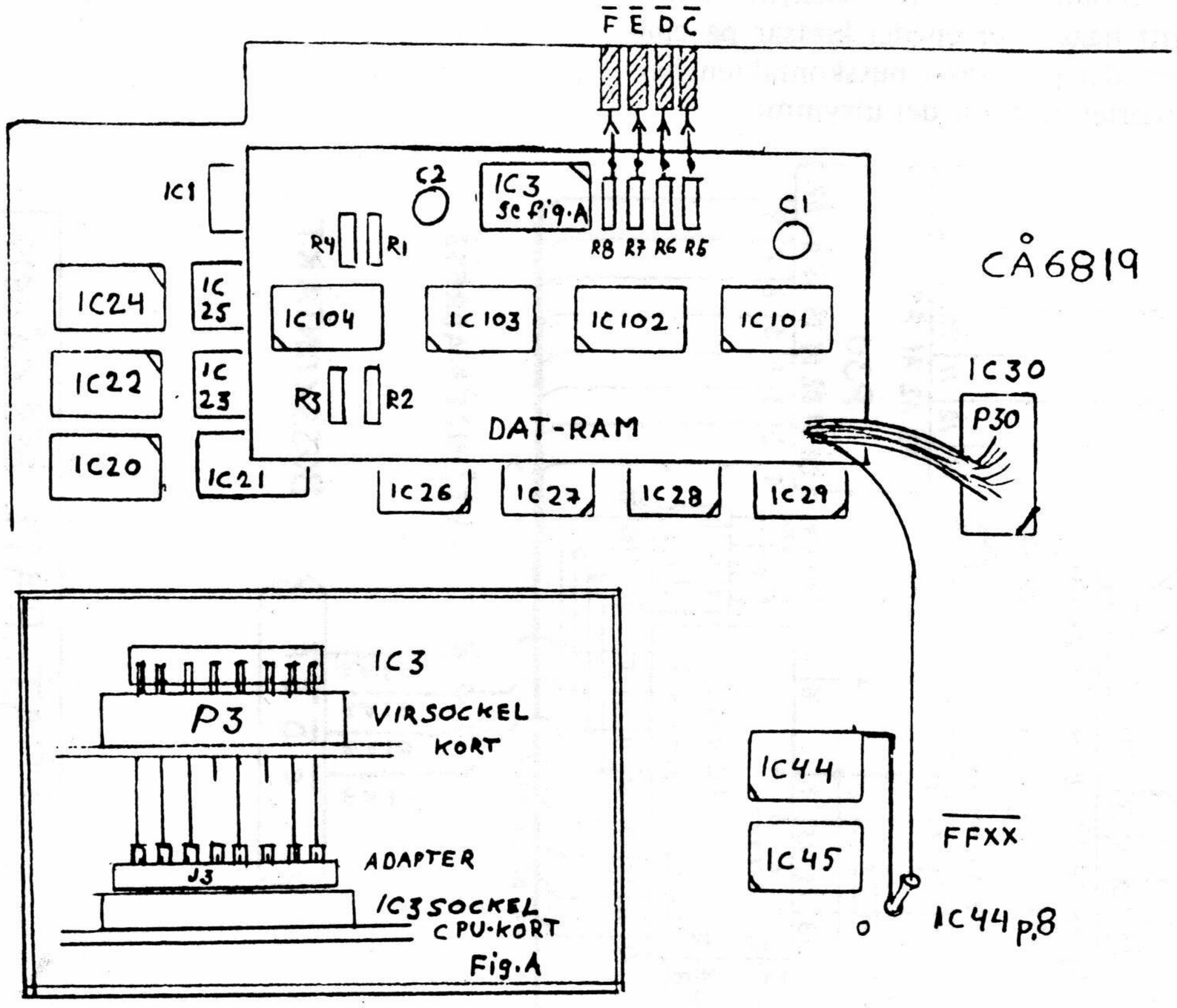


Fig 6. PLACERING AV DAT-RAM PÅ CPU-KORTET

3.2 ELEKTRISK FUNKTION

READ MODE

Adressledningarna A12-A15 direkt från 6809 på sockel IC3 pin 6,10,12,4 går till IC103 pin 3,6,10,13 som är B-sidan av en 74LS257 (Quad 2-line-to-1-line Data selector). Select-ingången IC103 pin 1 väljer ut A-sidan med låg och B- sidan med hög signal. Om select är hög kommer A12-A15 passera till utgångarna Y IC103 pin 4,7,9,12 och vidare till adressingångarna på IC102 och IC104. Data ut från den adress som A12-A15 pekar på i IC102 används som Extended Adress EA16-EA19 stift NOT F,E,D,C, på busskontakten. Samma adress A12--A15 ger data ut från IC104 som kallas A12'-A15' Dom är kopplade till ingångarna på bus- drivern IC3 och därefter ut på bussen som tidigare. Beroende av innehållet på en viss adress i IC104 så ändras alltså bussadressen därefter. På detta sätt mappas page i 4k block in med A12'-A15' Ett exempel: 6809 ska läsa adress $1000 A12-A15 är då 1. På adress 1 i DAT-RAM IC102 IC104 är data out = $E2. Extended Adress blir då $E som väljer ut det minneskort som är avkodat för EA16-EA19 = $E. Vidare kommer A12'-A15' på bussen att vara $2, 6809 erhåller då data från adress $2000 på minnes- kort 1 (NOT $E).

WRITE MODE

För att sätta upp DAT-RAM måste 6809 kunna skriva in data på alla 16 minnes- adresserna. Här används en 74LS155 (Dual 2-line-to-4-line Decoder) IC101. Den ger låg signal till SELECT IC103 pin 1 och låg till W IC102,IC104 pin 3. IC101 pin 9 går låg när FFXX (från IC44 pin 8) IC101 pin 3 och R/W IC101 pin 13 är låga. Därmed kopplar IC103 om till A-sidan och adressledningarna A0-A3 får tillträde till DAT-RAM IC102,IC104. När dessa stabiliserats går IC101 pin 1 (O2) hög, då kommer IC101 pin 7 att gå låg och enabla WRITE mode i DAT-RAM. 6809 ser alltså DAT-RAM som 16 st 8-bit "Write Only" register på adresserna $FFF0-FFFF. Observera att 74S189 inverterar data som skrivs in, tex om man skriver in $0F på adress $FFF0 blir data ut $F0, låga delen mappar då in page 0 när 6809 adresserar området $0000-0FFF, höga delen ger select till minneskort med Extended Adress $F. Därför avkodas minneskort 0,som normalt är User-ram, till Extended Adress $F.

AVKODNING AV MINNESKORT FÖR EXTENDED ADRESSERING

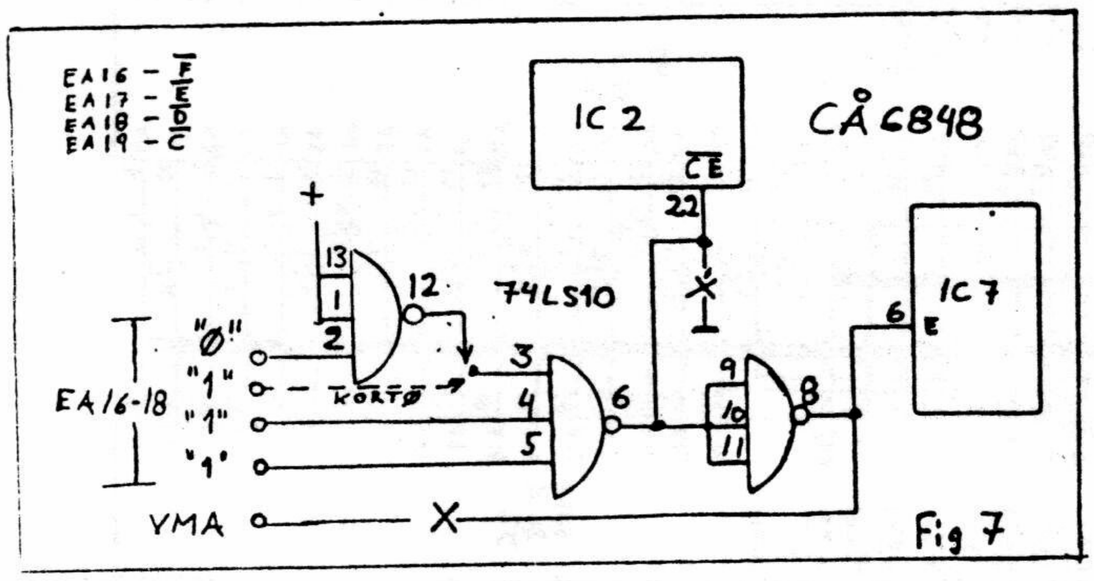


Fig 7

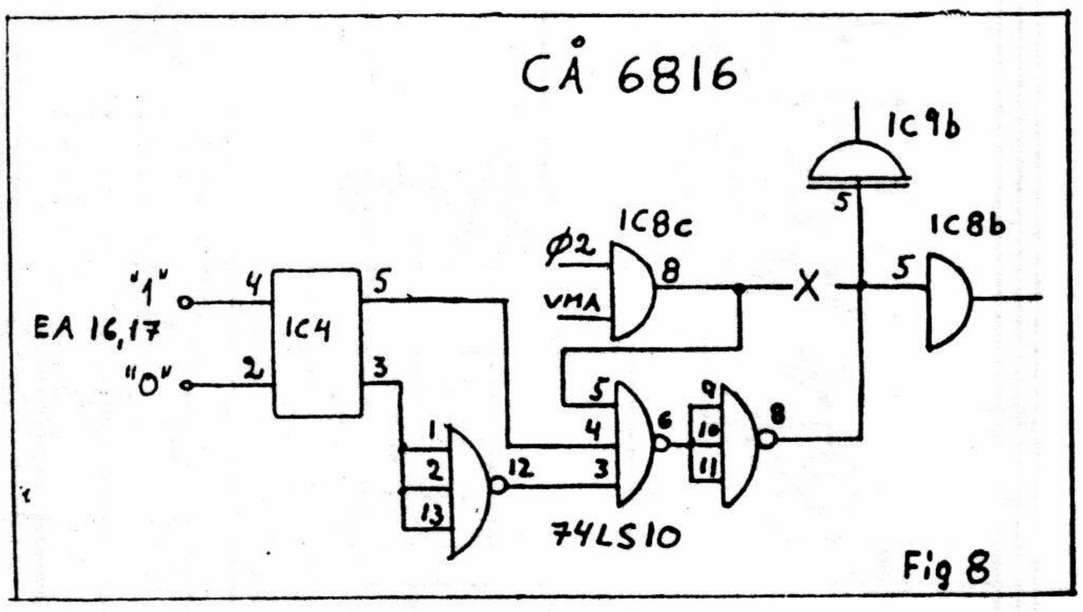


Fig 8

48k Minne CÅ-6848.
Se fig 7.

Modifiera minneskorten med en 74LS10 som kopplas till IC-7 pin 6 och till IC-2 pin 22 (kapa mönstret till jord). MONTERING: Böj benen på en 14-ben IC- sockel och löd den på ett litet Veroboard vänt så lödsidan är mot sockeln limma sedan fast kortet på minneskortet i närheten av IC-10.
För att få dom kontaktstift till buss- kontakten som saknas på minneskortet kan man värma loss sådana från ett kasserat kretskort. Dom limmas fast på minneskortet med tex cyanoakrylat-lim (sk snabblim).
Dom fyra adressledningarna EA16 - EA19 avkodas ej fullständigt, man kan an- vända max 4 st minneskort med den här kopplingen. OBS att alltid ett kort måste avkodas som kort 0 (User-ram).

--- EA ----				Kort Hög del	
19	18	17	16	DAT-RAM	
Kort 0.	x	1	1	1	Adress $F ($0)
Kort 1.	x	1	1	0	Adress $E ($1)
Kort 2.	x	1	0	1	Adress $D ($2)
Kort 3.	x	0	1	1	Adress $B ($4)

16k minne CÅ-6816.
Se fig 8.

Modifieras på liknande sätt men här är endast 2 st EA-ledningar avkodade. Då kan max tre minneskort användas.

Exempel.
Ett minneskort är avkodat för Extended Adress $F och ett för $B (kort 0,3).

Skriv in $4F i DAT-RAM på adress $FFF0
och $4E i DAT-RAM på adress $FFF1

Då kopplas kort 3 in på alla adresser inom området $0000 - 1FFF.

Skriv in:

$0000 = $00 Page 0
$0FFF = $00
$1000 = $11 Page 1
$1FFF = $11
$2000 = $22 Page 2

Skriv in:

$FFF0 = $4E Kasta om page
$FFF1 = $4F 0,1 på kort 3

Läs adr:
$0000 = $11
$0FFF = $11 Page 0 och 1
$1000 = $00 har bytt
$1FFF = $00 PLATS !!
$2000 = $22

Återställ till kort 0, Page 0,1:
Skriv in:

$FFF0 = $0F
$FFF1 = $0E

Eller tryck på RESET-knappen

Läs adr:

$0000 - 1FFF = som innan ex
$2000= $22

CÅ:s CPU-kort se nästa sida.

Principschemat för CPU-kortet. i ORIGINALUTFÖRANDE

RADIO & TELEVISION - NR 12 - 1979

VME-BUSSEN

Nedanstående saxat ur '68' Micro Journal Volume VI Issue II, Februari 1984, sid 40.
Översättning Ingemar Skarpås.

PRESS INFORMATION (Mostek Corporation, Motorola Inc, Signetics Corp)
UTÖKAD ARKITEKTUR, NYTT CHIP STÖD ÖKAR VMEbus FÖRMÅGAN. Phoenix, Arizona, 3:e November 1983...
För att öka marknads- storleken till 750M$ för VMEbus kompatibla produkter år 1988, i en total marknad för mikrodatorkort av 2,2B$, har styrelserna för Mostek, Motorola och Signetics/Philips tillkännagivit en teknisk Överenskommelse om viktiga tilläggsbeståndsdelar i hela VMEsystem arkitekturen. I de nyligen definierade buss-stöds strukturerna ingår en tilläggs, höghastighets- minnes-buss, VMXbus, och en 'self-arbitrating' höghastighets seriell buss, VMSbus, som kan läggas in i nuvarande system byggda på den redan existerande VME-buss standarden.

De här tilläggs-bussarna möter den grundläggande VMEbus kopplingen som ursprungligen introducerades i oktober 1981, och som sedan dess har anammats av över 60 tillverkare över hela världen. Ett större mått på det breda mottagande den fått inom industrin är att VME-bussen för närvarande blir formellt standardiserad av både IEEE (som P1014) och IEC (International Electronical Committe, i Geneve). VME-bussen och dess tilläggsbussars strukturer lägger grunden till hela VMEsystemets arkitektur som förenklar sammankopplingen till kompletta system från högpresterande 8, 16, och 32-bitars systembeståndsdelar på kortnivå. Den existerande VME-buss kopplingens standard ger den grundläggande möjligheten till data överföring mellan de mest betydelsefulla beståndsdelarna, medan VMX-bussen underlättar utökningen av lokalt processorminne med typiskt mycket höga prestanda. För snabb kommunikation av korta meddelanden mellan systemmoduler, så använder VMS-bussen bara två ledare (klocka och data) för att ge en effektiv 'party line' mellan systemdelar.

Mostek, Motorola och Signetics/Philips har överenskommit om detaljerade tekniska specifikationer för de nya VMX- & VMS-bussarna, för vilka dokumentation har gjorts och hålls tillgänglig. Alla tre bolagen har överenskommit om fortsatt högnivå specifikation för hela VME-system arkitekturen, för att främja kompatibiliteten mellan VME-system produkter från många tillverkare. Dessutom, har bolagen gjort antydanden att de kommer att presentera självständigt utvecklade produkter med VME-buss arkitektur under 1984.
För att följa den tradition som skapades i och med VME-bussen kommer bolagen att ställa de nya buss-specifikationerna och hjälpdokumenten till allmännhetens förfogande, utan några copyright eller licensavtals-bestämmelser. Man tror att den här taktiken främjar en utbredd tillgänglighet av VME-system-kompatibla podukter när slutanvändare och tillverkare ännu mer vill ha VMEsysemets arkitektur för sina högpresterande 16- & 32-bitars maskiner. Mostek och Signetics/Philips producerar båda den 16/32-bitars M68000 MPU som ursprungligen utvecklades av Motorola. Utan att därför begränsa applikationerna till 68000-baserade system, har VME-bussens arkitektur gjorts så att den möjliggör ett antal systemnivåer som är önskvärda för högpresterande mikrodatorsystem där oftast M68000-familjen kommer att användas.

Sedan den introducerades har VME-bussen blivit använd i en industriell process-styrning, bildbehandling, arbetsplatser för ingenjörsteknik, och digitala kommunikations-nätverk av stor bredd. Man har också talat om nya IC i M6800-familjen för tilläggs- bussarna för att underlätta utvecklingen av VME-system- kompatibla kretskort. De här nya ICna inkluderar 68172 VME-buss kontrollern (E-BUSCON), 68173 VMS-buss kontrollern (S-BUSCON) och 68174 Bus Arbiter. Av dessa kommer Signetics att utveckla 68172an och 68173an, medan Motorola kommer att utveckla 68174an. Bolagens planer är att prov av dessa nya buss-stöds chips kommer att finnas tillgängliga under andra halvan av 1984. Fortsatta tekniska diskussioner mellan de tre bolagen förväntas ge till resultat preciseringar av ytterligare tilläggsbuss-stöds chips för att komplettera den totala VMEsystem arkitekturen.

*

KRYPTERINGSPROGRAM

Alla har vi väl någon gång fascinerats av 'hemliga chiffer', åtminståne när vi var i 'Kalle Blomkvist'-åldern. Det finns faktiskt en del praktiska tillämpningar av chiffer, t ex när du skall lägga en fil i klubbdatorns 'mailbox' till någon. Alla kan ju läsa den filen, om den inte krypteras fårståss. Då gör man så här: Man kommer överens med den man brukar skicka texter till om en kodnyckel och sedan kallar man bara filen till.XXXX.från.YYYY och så skickar man över den krypterade filen. Mottagaren vet då vem som skickat meddelandet och behöver bara ta in texten till sin dator och använda den överenskommna kodnyckeln.

Programbeskrivning

rad 130: Du anger en kodnyckel, den kan bestå av en eller flera STORA BOK-

STÄVER, att den gör det kollas av subrutinen på rad 340. Nyckeln kan även bestå av siffror.

rad 140: Du frågas om du skall koda eller dekoda. Endast 1 eller 2 duger som svar. Detta kollas på rad 175 och 176. I annat fall åker du tillbaka till rad 140.

rad 160: Här namnger du den fil som ligger på arbets-driven och som skall krypteras. Det skall vara en textfil med extension .TXT! Se subrutin under 170.

rad 170: Här namnger du den fil som ligger på arbetsdriven och som skall dekodas. Det är då en textfil som skapats av detta program. Den skall ha extension .TXT!
I subrutinen på rad 500 kontrolleras om du har givit en fil som används som input eller output är namngiven med för många tecken, eller om den redan innehåller filnamn med extension .TXT. Om filnamnet inte har ext .TXT så lägger programmet till detta i rad 540.

rad 190: Här namnges den fil som blir output från programmet. Samma saker gäller för detta filnamn som för dom som anges i rad 160 eller 170, men här kontrolleras filnamnet i subrutinen på rad 600, som annars fungerar likadant som den på rad 500.

Så här krypterar/dekrypterar programmet.
Det tecken, eller de tecken som ingår i kodnyckeln omvandlas till ett heltal av det genomsnittliga ASCII-värdet på kodnyckeln. Sedan läser programmet in den fil som skall behandlas. De tecken som ingår i den filen omvandlas ett och ett till ett ASCII-nr. Detta ASCII-nr läggs sedan samman med ASCII-nr:et som skapades av kodnyckeln. Detta sker på rad 250 om du kodar text, och på rad 260 om du dekodar text. Eftersom du kan få ASCI-värden som är större än dom som finns i ASCII-tabellen finns det på dessa rader (250,260) en 'wrap-around'-funktion som ser till att alla ASCII-världen håller sej mellan 32 och 126, de är ju de printbara ASCII-tecknen.
Rad 281 adderar upp kodnyckelns ASCII-värden med 1 om du kodar text. Rad 290 subtraherar kodnyckelns ASCII-värden med 1 om du dekodar text. Detta gör det svårare för en utomstående att 'knäcka' en kodad text, eftersom det ju skapas en ny nyckel för varje rad!
Rad 300 printar ut den text som skall behandlas, och printar ut den behandlade, färdiga texten på raden under. Rad 310 skriver ut den text som då står på den undre raden till disk.

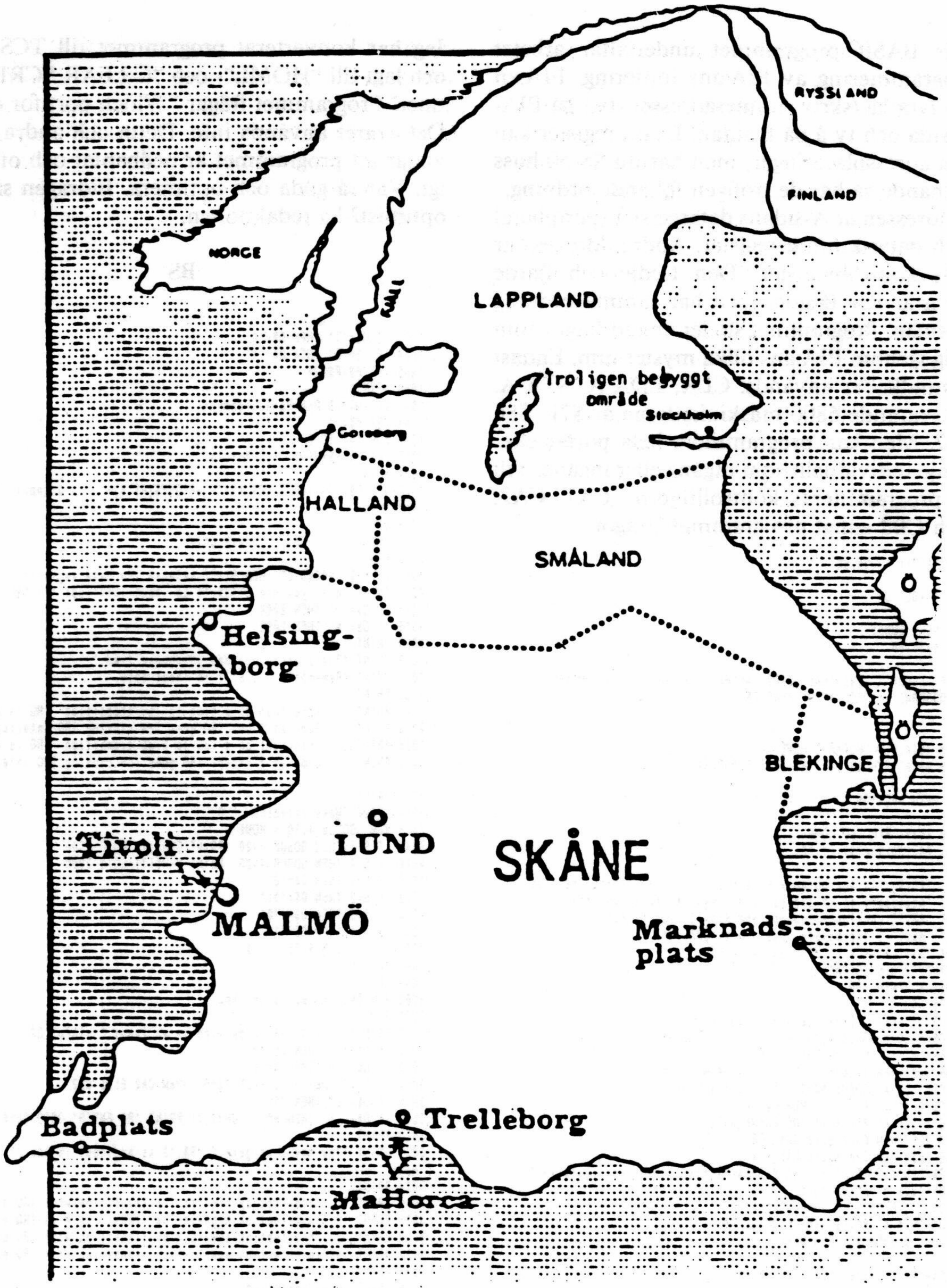
Mycket nöje! Ingemar Skarpås

Anm. Rad 220 i programmet avslutar krypteringen efter en tom rad.

Jag har provat med att stryka raden och det har till synes gått bra.
Annars får man ändra i programmet på annat sätt om tomma rader finns med i textfilen som ska krypteras.

Bernt

```
1 REM ***************A BASIC Program for HOME Cryptography***************
2 REM ---------------by Ralph Roberts, POB 8549, Asheville----------------
3 REM ---------------NC 28814, USA                          ----------------
4 REM ************************************************************************
5 REM +++++++++++++++Publicerat i Byte, April 1982, sid 432+++++++++++++++
6 REM +++++++++++++++Tillfixat från Computerware's BASIC 6.0+++++++++++++++
7 REM +++++++++++++++till TSC:s BASIC och översatt till     +++++++++++++++
8 REM +++++++++++++++svenska av Ingemar Skarpås 840401      +++++++++++++++
9 REM ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10 REM .............Program för kodning och dekodning.....................
100 PRINT "PROGRAM FÖR KODNING":PRINT:PRINT
101 PRINT "Du använder en redan skapad textfil som skall finnas på work-dr
iven."
102 PRINT "De filer du skapar genom kodning eller dekodning kommer också a
tt"
103 PRINT "hamna på workdriven."
104 PRINT:PRINT "Programmet ger automatiskt alla filer namn som slutar på
.TXT "
107 PRINT : PRINT : PRINT
130 INPUT "Ange KODNYCKEL (endast stora bokstäver) " ; P$
131 LET FF%=0
135 GOSUB 340
136 IF FF%=1 THEN 130
140 INPUT "ÖNSKAS KODNING (1) ELLER DEKODNING (2) " ; I$
155 PRINT
160 IF I$="1" THEN INPUT "NAMN PÅ FIL SOM SKALL KODAS " ; A$ : GOSUB 500
170 IF I$="2" THEN INPUT "NAMN PÅ FIL SOM SKALL DEKODAS " : A$ : GOSUB 500
175 IF I$="1" THEN 190
176 IF I$="2" THEN 180 ELSE 140
180 IF I$="2" THEN LET A=A*(-1)
190 INPUT "GE NAMN ÅT DEN NYA FIL SOM SKAPAS GENOM KODNING/DEKODNING " ; C
$ :GOSUB 600
205 ON ERROR GOTO 1000
206 OPEN OLD A$ AS 1
209 OPEN NEW C$ AS 2
210 INPUT LINE £1, B$
215 IF ERR<>8 THEN ON ERROR GOTO 330
220 IF B$="" THEN 330
230 FOR X=1 TO LEN(B$)
240 LET B=ASC(MID$(B$,X,1))
250 IF I$="1" THEN LET B=B+A : IF B>127 THEN LET B=(B-127)+32
260 IF I$="2" THEN LET B=B+A : IF B<32 THEN LET B=(B+127)-32
270 LET D$=D$+CHR$(B)
280 NEXT X
281 IF I$="1" THEN LET A=A+1 : IF A=126 THEN LET A=1
290 IF I$="2" THEN LET A=A-1 : IF A=-126 THEN LET A=-1
300 PRINT : PRINT B$ : PRINT D$
310 PRINT £2,D$ : D$="" : B$=""
320 GOTO 210
330 CLOSE 1,2
333 END
340 LET FF%=0
341 LET A=0 : P=0 : P1=0
345 LET P=LEN(P$)
350 FOR X=1 TO LEN(P$)
360 LET P1=P1+ASC(MID$(P$,X,1))
361 NEXT X
370 LET A=INT(P1/P)
375 IF A>96 THEN 376 ELSE FF%=0 :GOTO 380
376 PRINT "KODNYCKEL OGILTIG !" :LET FF%=1
380 RETURN
500 ZZ%=LEN(A$)
501 IF ZZ% >8 THEN INPUT "FILNAMN FÖR LÅNGT. NYTT FILNAMN (max 8 tkn) " :


A$ : GOTO 500 ELSE GOTO 509
509 TA$=".txt"
510 PA%=INSTR(1,A$,TA$)
520 IF PA%=0 THEN TA$=".TXT" ELSE GOTO 550
530 PA%=INSTR(1,A$,TA$)
540 IF PA%=0 THEN A$=A$+TA$ ELSE 550
550 RETURN
600 ZY%=LEN(C$)
601 IF ZY% >8 THEN INPUT "FILNAMN FÖR LÅNGT. NYTT FILNAMN (max 8 tkn) " :
C$ : GOTO 600 ELSE GOTO 609
609 TC$=".txt"
610 PC%=INSTR(1,C$,TC$)
620 IF PC%=0 THEN TC$=".TXT" ELSE GOTO 650
630 PC%=INSTR(1,C$,TC$)
640 IF PC%=0 THEN C$=C$+TC$ ELSE 650
650 RETURN
1000 IF ERR=4 THEN PRINT "DETTA FILNAMN SAKNAS PÅ DISKETTEN: " : A$
1010 RESUME 1
```



Sverige sett ur Skånes perspektiv

PROGRAMINITIERING AV PIA

Det här BASIC-programmet underlättar arbetet vid programmering av PIA:ans initiering. PIA:an tar upp fyra läs/skriv minnesadresser, två på PIA-:ans A sida och tv å på B sidan. Dessa register kan ha olika adressplaceringar, men har du SS-50 buss eller liknande så har de troligen följande ordning. Första adressen är A-sidans dataregister (peripheral reg.) och datariktnin gsregister. Andra adressen är A-sidans kontrollregister. Den tredje och fjärde adresse n är fÖr B-sidan och har samma ordning som A-sidan. Programmet ställer en rad frågor som leder dig igenom PIA:ans olika myster ium. Endast tre operandkoder används: CLR, LDA A, STA A. De motsvarande 6800 maskinkoderna är $7F, $86 och $B7. Man kan programmera hela portar eller varje linje individuellt som utgång eller ingång. Till sist så programmeras kontrollinjerna CA1, CA2, CB1 och CB2 enligt pro grammets frågor.

Jag har konverterat programmet till TCS BASIC och lagt till "HOME"- och "C LEAR SCREEN"rutiner. Programmet frågar "Which port for output". Det svaret används inte. Detta och andra detaljer vissar att programmet är nedbantat och ofullständ igt. Var så goda och optimera. Bidragen sänds till optimistiska redaktionen.

BS

```
1 REM' INKNAPPAT AV OS/BS
2 REM'
3 REM'     PIASET.UP
4 REM'
5 REM'     GENE EMBRY
6 REM'     9/28/80
7 REM'
8 REM' REF. LANCE A.LEVENTHAL, 6800 ASSEMBLY LANGUAGE PROGRAMMINGG
9 REM'    OSBORNE & ASSOCIATES, CHAPTER 11
10 REM'
11 REM'
0015 GOSUB 4010:REM  CLEAR + HOME
0020 GOSUB 9800: REM TO INITIALIZE THE PROGRAM VARIABLES
0099 REM
0100 REM  MAIN SECTION
0101 REM
0110 GOSUB 4020: REM  DO HOME
0112 PRINT TAB(W-LEN(T$)/2);T$
0114 PRINT
0116 PRINT TAB(5);"This program will assist you to determine the code"
0118 PRINT "necessary to initialize a PIA device. We assume that the"
0120 PRINT "SWTPC configuration for the PIA is used.":PRINT
0122 INPUT "What is the memory location of CRA ",A$
0130 INPUT "Will Side A of PIA be used ",Q$
0132 IF Q$="n" THEN Q$="N"
0134 IF Q$="N" THEN 200
0150 LET P$=A$
0152 LET P$="$"+P$+"  Clear out control register"
0154 LET C$="CLR ":GOSUB 8000: REM Print a line of code
0156 LET S$="A":IF F1 <>0 THEN S$="B"
0160 GOSUB 1000:REM Determine how the side will be used.
0162 LET C$="LDA A":GOSUB 8000: REM Print a line
0166 LET O$="-1":IF F1 <> 0 THEN O$="+1"
0168 LET C$="STA A":P$="$"+A$+O$:GOSUB 8000
0170 GOSUB 2000:REM Configure CA1 (CB1)
0180 GOSUB 3000:REM Configure CA2 (CB2)
0182 LET P$="£%"+LEFT$(P$,4)+"-"+RIGHT$(P$,4)+"   Note configuration"
0184 LET C$="LDA A":GOSUB 8000
0186 LET O$="":IF F1 <> 0 THEN O$="+2"
0188 LET C$="STA A":P$="$"+A$+O$:GOSUB 8000
0190 IF F1 <> 0 THEN 900:REM Done
0199 REM
0200 REM Check for Side B configuratioN
0201 REM
0210 IF F1=0 THEN GOSUB 9700:REM Might want to configure Side B
0220 IF F1=2 THEN 900:REM Side B will not be used
0230 LET P$=A$+"+2"
0240 GOTO 152:REM Configure Side B
0299 REM
0900 REM FINISHED
0901 REM
0970 PRINT
0980 PRINT "Bye!"
0990 END
0999 REM
1000 REM SELECTING OF INPUT & OUTPUT LINES
1001 REM
1010 LET P$=""
1020 PRINT "Will all lines of side ";S$;" be inputs";: INPUT Q$
1022 IF Q$="Y" THEN Q$="Y"
1024 IF Q$="Y" THEN P$="£%0000-0000  All lines are inputs":GOTO 1090
1030 PRINT "Will all lines of Side ";S$;" be outputs";:INPUT Q$
1032 IF Q$="y" THEN Q$="Y"

1034 IF Q$="Y" THEN P$ ="£%1111-1111   All lines are outputs":GOTO 1090
1040 FOR X=1 TO 8:REM Get data line bit structure
1046 PRINT "Specify i/o configuration of the data line £";X;
1048 INPUT " (I=Input O=Output) ",Q$
1050 IF Q$ <> "I" AND Q$<> "O" THEN PRINT CHR$(7):PRINT "Reenter":GOTO 1046
1052 IF Q$="I" THEN Q$="0"
1054 IF Q$="O" THEN Q$="1"
1058 LET P$=P$+Q$
1060 NEXT X
1062 LET Q$=P$:REM Re-assignment
1064 LET P$=""
1070 FOR X=1 TO 8
1072 LET P$=P$+MID$(Q$,8-X+1,1)
1074 NEXT X
1080 LET P$="£%"+LEFT$(P$,4)+"-"+RIGHT$(P$,4)+"     0=input  1=output"
1090 RETURN
1099 REM
2000 REM  CA1 or CB1 - inputs only
2001 REM
2010 LET P$="00000100":REM Bit 2 selects the data register
2012 PRINT "Will you use C";S$;"1 as input interrupt ";:INPUT Q$
2014 IF Q$="n" THEN 2090
2016 IF Q$="N" THEN 2090
2026 PRINT
2028 PRINT "There are 4 options to choose from."
2030 PRINT "Remember: C";S$;"1 is input only."
2032 PRINT
2034 PRINT "1. Interrupt on a high-to-low transition- IRQ is disabled."
2036 PRINT "2. Same as 1 but IRQ goes low causing IRQ interrupt to cpu."
2038 PRINT "3. Interrupt on a low-to-high transition- IRQ is disabled."
2040 PRINT "4. Same as 3 except IRQ goes low causing IRQ interrupt to cpu.
"
2042 PRINT
2044 INPUT "Make selection ",S
2045 REM  GOSUB 4020 = HOME
2046 IF S<1 THEN GOSUB 4020 :PRINT CHR$(7):GOTO 2000
2048 IF S>4 THEN GOSUB 4020 :PRINT CHR$(7):GOTO 2000
2050 IF S=1 THEN Q$="00"
2052 IF S=2 THEN Q$="01"
2054 IF S=3 THEN Q$="10"
2056 IF S=4 THEN Q$="11"
2060 LET P$=LEFT$(P$,6)+Q$
2090 RETURN
2099 REM
3000 REM CA2 or CB2 as an input
3001 REM
3010 PRINT "Will C";S$;"2 be used as an input ";:INPUT Q$
3012 IF Q$="n" THEN Q$="N"
3013 IF Q$<>"N" THEN 3016
3014 IF F1=0 THEN 3100 ELSE 3200 : REM CA2 ELSE CB2
3016 IF Q$="y" THEN Q$="Y"
3018 IF Q$<>"Y" THEN PRINT CHR$(7):PRINT "RE-ENTER":GOTO 3010
3020 PRINT
3022 PRINT "You may configure C";S$;"2 as an input in one of the following
 ways."
3024 PRINT
3026 PRINT "1. Interrupt on high-to low transition with IRQ disabled."
3028 PRINT "2. Interrupt on high-to low transition with IRQ enabled."
3030 PRINT "3. Interrupt on low-to-high transition with IRQ disabled."
3032 PRINT "4. Interrupt on low-to-high transition with IRQ enabled."
3034 PRINT
3036 INPUT "Make selection ",S
3038 IF S<1 THEN PRINT CHR$(7):PRINT "REENTER":GOTO 3020

3040 IF S>4 THEN PRINT CHR$(7):PRINT "REENTER":GOTO 3020
3042 IF S=1 THEN Q$="000"
3044IF S=2 THEN Q$="001"
3046 IF S=3 THEN Q$="010"
3048 IF S=4 THEN Q$="011"
3060 LET P$=LEFT$(P$,2)+Q$+RIGHT$(P$,3)
3090 IF F1=1 THEN 3200
3099 REM
3100 REM Control of CA2 as an output
3001 REM
3110 INPUT "Will CA2 be used as an output ",Q$
3112 IF Q$="n" THEN Q$="N"
3114 IF Q$="N" THEN 3190
3120 PRINT
3122 PRINT "Configuring CA2 as an output."
3123 PRINT "Note that CRA5 is high."
3124 PRINT
3126 PRINT "1. Goes low on negative transition of E after a READ operation
."
3128 PRINT TAB(4);"Goes high when CRA-7 is set!"
3130 PRINT "2. Goes low as above. Goes high on negative edge of E during a
 deselect."
3132 PRINT "3. Goes low when CRA-3 goes low as a result of a WRITE to CRA.
```

```
3134 PRINT TAB(4);"Goes high when a WRITE to CRA sets CRA-3 high."
3136 PRINT "4. Goes low when a WRITE to CRA sets CRA-3 to low."
3138 PRINT TAB(4);"Goes high when CRAA-3 is set high as a result of WRITE
to CRA."
3140 PRINT
3142 INPUT "Make selection ",S
3144 IF S<1 THEN 3122
3146 IF S>4 THEN 3122
3148 IF S=1 THEN Q$="100"
3150 IF S=2 THEN Q$="101"
3152 IF S=3 THEN Q$="110"
3154 IF S=4 THEN Q$="111"
3160 LET P$=LEFT$(P$,2)+Q$+RIGHT$(P$,3)
3190 RETURN
3199 REM
3200 REM Control of CB2 as an output
3201 REM
3210 INPUT "Will CB2 be used as an output ",Q$
3212 IF Q$="n" THEN Q$="N"
3214 IF Q$="N" THEN 3190
3222 PRINT "Configuring CB2 as an output."
3223 PRINT "Note that CRB-5 is high."
3224 PRINT
3226 PRINT "1. Goes low on positive transition of E after a WRITE to DRB."
3228 PRINT TAB(4);"Goes high when CB1 is set!"
3230 PRINT "2. Goes low as above. Goes high on positive edge of E during a
 deselect."
3232 PRINT "3. Goes low when CRB-3 goes low as a result of a WRITE to CRB.
"
3234 PRINT TAB(4);"Goes high when a WRITE to CRB sets CRB-3 high."
3236 PRINT "4. Stays high as long as CRB3 is high. Cleared when WRITE to C
RB."
3238 PRINT TAB(4);"Goes high when CRB-3 is set high as a result of WRITE t
o CRB."
3240 PRINT
3242 INPUT "Make selection ",S
3244 IF S<1 THEN 3222
3246 IF S>4 THEN 3222
3250 GOTO 3148
3299 REM
```

```
4000 REM    HOME  (RT-HOME)
4001 REM 4010 PRINT CHR$(27);CHR$(5)
4002 REM 4020 PRINT CHR$(27);CHR$(12)
4010 PRINT CHR$(26) :REM  ADM-3A HOME  ÜZ
4020 PRINT CHR$(30) :REM  ADM-3A CLEAR SCREEN ÜÜ
4100 RETURN
4110 REM
8000 REM The code
8001 REM
8010 PRINT £Q
8020 PRINT £Q,TAB(10);C$;TAB(20);P$
8080 PRINT £Q
8090 RETURN
8099 REM
9700 REM Test for Side B
9701 REM
9710 INPUT "Do you want to configure Side B of the PIA ",Q$
9712 IF Q$="n" THEN Q$="N"
9714 IF Q$="N" THEN F1=2:GOTO 9790
9720 LET F1=1:REM Side B flag
9790 RETURN
9799 REM
9800 REM Initalize the program variables
9801 REM
9810 REM STRING=50
9812 REM LINE=0: REM Unlock the line-feed/carriage return format
9814 LET W=35
9816 LET F1=0: REM Flag for Side A or Side B of the PIA
9820 LET T$="PIA Initialization"
9840 INPUT "Which port for output ",G
9842 IF G<>3 THEN G=1
9890 RETURN
9899 REM
```

EN FLEXIBEL LÅGPRIS STYR-MODUL MED MIKROPROCESSOR

Gör ditt senaste projekt intillegent till minimala kostnader.

Av David L Craig
134 Victor St.
Holland Park 4121
Queensland, Australia

(översättning av Ingemar Skarpås)
Ur BYTE, December 1982 sid 486.

Ett av de ursprungliga målen med mikroprosessorn var att ersätta komplexa, diskreta, och småskale-integrerade kretsar (IC) med generellt användbara, programmerbara enheter som skulle vara tillräckligt flexibla för att klara behoven hos ett otal tillämpningar. Mikroprocessorn kan krympa storleken och komplexiteten hos de digitala logikkretsarna tillräkligt mycket för att utföra en specifik uppgift och göra kretsar praktiskt möjliga som annars skulle vara för komplexa att ens tänka på. Lite bättre 'uppförande' och användarbekvämlighet kan ofta läggas till utan extra kostnader. Problemet med experiment- och 'specialare', eller en styck, projekt har varit att finna en metod att bygga en kontroller med mikroprocessor som är tillräckligt billig och liten för att tävla med mer konventionell kretsdesign i 'små' eller osofistikerade tillämpningar.

Många små en-korts datorer och mikroprocessorutvecklings kort finns tillgängliga som kan tjäna som kontroller- moduler, men dom är för stora, onödigt kraftfulla och alldeles för dyra för dessa små styr-tillämpningar. Det finns nu också ett antal mikrodatorer på ett chip (t ex Intel 8048, Zilog Z8, Motorola 6805), men de flesta av dessa är opraktiska för konstruktioner i små upplagor, därför att dom har mask-programmerad ROM (läs-minne), inbyggt på chippet, sålunda blir den minsta kvantitet man kan beställa flera tusental styck.

De få mikrodatorer som endera har EPROM (raderbara, programmerbara läs-minnen) på chippet)(t ex Intel 8748, och Motorola 68705) eller 'piggy-back' sockel (t ex Zilog Z8-03) som standard EPROM skulle vara idealiska i tillämpningar i små kvantiteter därför att dom innehåller en processor, RAM (skriv-&läsminne), EPROM, I/O (input/output) ledare och en 'timer', vilka alla är nödvändiga för generellt användbara kontrollers. Oturligt nog så är dessa fortfarande mycket svåra att få tag i och inte billiga nog för att göra enstaka kontrollers.

Målet med den skapelse som beskrivs i denna artikel var att få fram en mycket liten och billig mikroprocessor-modul som skulle passa till ett stort antal styrändamål, och bara använda de mest lättillgängliga komponenterna. Den totala kostnaden för den modul som här presenteras är omkring 30 dollar. Modulen består av ett enkelsidigt tryckt kretskort, 100 x 60 mm och med 24-poligt kontaktdon. Det finns plats för en Motorola 6802 mikroprocessor, en 2716 EPROM (+5V en spänning), en 6820 (eller 6821)

PIA (periferal interface adapter), en kristallstyrd klock-krets med 60 Hz avbrottsgenerering, och en +5V spänningsregulator. Det finns på det här viset tillgängligt 128 bytes av programmerbart minne, 2048 bytes av EPROM, 19 programmerbara I/O snören, och en möjlighet till realtids-avbrott. All kommunikation med externa kretsar sker genom de 19 I/O snörena.

Man kan på detta sätt med denna uppställning skapa ett kraftfullt styrdon. Det kompletta kretsschemat till modulen visas i figur 1.

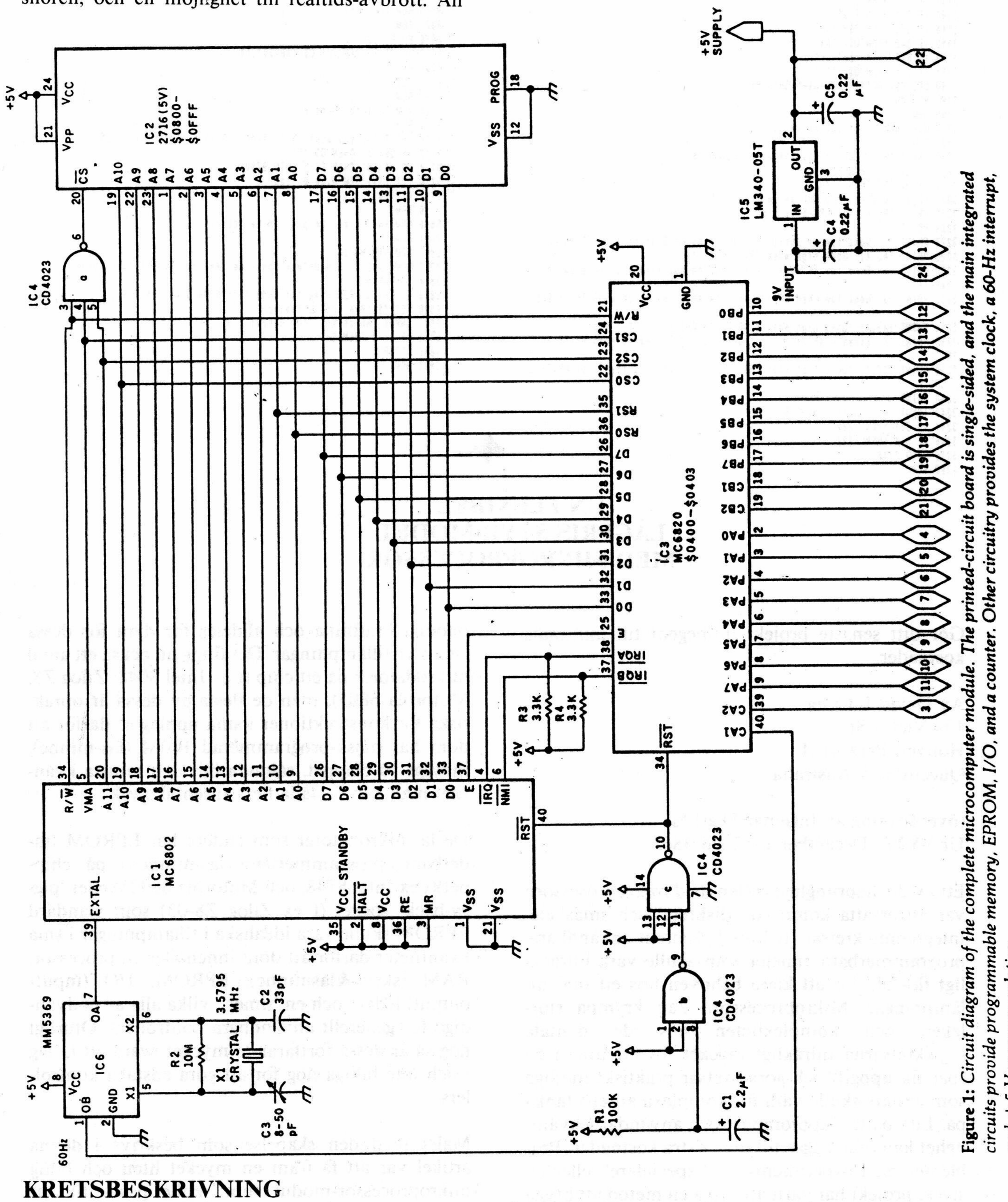


Figure 1: *Circuit diagram of the complete microcomputer module. The printed-circuit board is single-sided, and the main integrated circuits provide programmable memory, EPROM, I/O, and a counter. Other circuitry provides the system clock, a 60-Hz interrupt, and +5-V power regulation.*

488 December 1982 © BYTE Publications Inc

KRETSBESKRIVNING

Den mikroprocessor som används i modulen är Motorola MC6802. MC6802 valdes därför att den finns tillgänglig nu och till en låg kostnad (mindre än 10$ styckevis). Dessutom innehåller den ett 128 bytes minne på chippet, så att inga extra minnes-IC behövs till 'kladd-block' (minne för lagring av tillfälliga värden på data).

för att styrdonet skall kunna utföra användbart arbete måste det utföra de instruktioner som lagrats i dess minne. styrdonet använder sej av EPROM av typ 2716 vilka kan lagra 2048 bytes. 2716 valdes däför att det är lätt att anskaffa, kostar ca 6$ styckevis, och är lätt att programmera och radera. Kapaciteten på 2716 tillåter ganska långa masinspråksprogram.

Modulen är utvecklad för att arbeta med spänning från ett 9 volts likriktat, ostabiliserat nätaggregat - som lätt tas från en nätadapter till en räknedosa. Stabiliserad 5 V likspänning fås via IC 5, en 7805a (eller LM340-T-5) tre-bens regulator. +5V finns tillgängligt i modulens kretskorts-kontakt för att användas till tillkopplade kretskort. Modulen drar ca 250 milliampere (mA) ström, men regulatorn kan ge säker ström på ca 400 milliampere utan kylning. (Försiktighet måste visas så att inte regulatorn belastas med för många extra kretar.)

Fastän mikroprocesson, MC6802, själv har inbyggd oscillator- krets som bara erfordrar en kristall för att fungera så används en annan oscillatorkrets. IC6 är en kristalloscillator och med en frekvensdelare 1 17 steg. När en 3.579545-MHz kristall används, den är en NTSC (National Television System Committee) 'television color subcarrier' kristall, blir frekvensdelarens signal 60 Hz. Den ansluts till ingången CA1 på PIAn MC6820 för att ge möjlighet till avbrott. Detta är mycket användbart i styr-tillämpningar för att tillåta tidtagning av förlopp utan att därför låsa fast processorn i tidssamordning via mjukvaru-loopar. MC6802 delar internt ned klockfrekvensen med 4, och går därför med 894886Hz - 10 % långsammare än mikroprocessorns högsta tillåtna klockfrekvens på 1 MHz.

Enda sättet att resetta som finns är genom att bryta strömmen. Kombinationen av R1 och C1 buffrad via två grindar (b och c) på IC4 ger en återställningspuls till MC6802 och MC6820 när strömen slås på. Det tvingar processorn att börja utföra en uppstartsrutin i EPROMet.

De åtta databuss snörena till processorn dras till motsvarande snören på 2716 och 6820. Dessa snören är inte buffrade och finns inte tillgång till utanför modulen.

Någon överdriven adressavkodning används inte; bara de av de 16 adress-snörena på MC6802 som behövs för att utskilja RAM, EPROM och I/O ur adressområdet. Det blir ett antal 'ogiltiga' adresser över. (Se figur 2 för en enkel översikt av adressområdet.)

PIAn MC6820 tillhandahåller 16 I/O snören som är individuellt programmerbara till ingångar eller utgångar. Dessutom har MC6820 4 programmerbara kontroll-snören. En av dessa kontroll-snören, CA1, används till ingång för 60Hz avbrottssignalen. De andra tre kontroll-snörena och de 16 I/O snörena finns tillgängliga på modulens kretskortskontakt för att ansluta kontrollern till andra kretsar. All kommunikation mellan kontrollermodulen och dessa extra kretsar sker via de 19 I/O snörena.

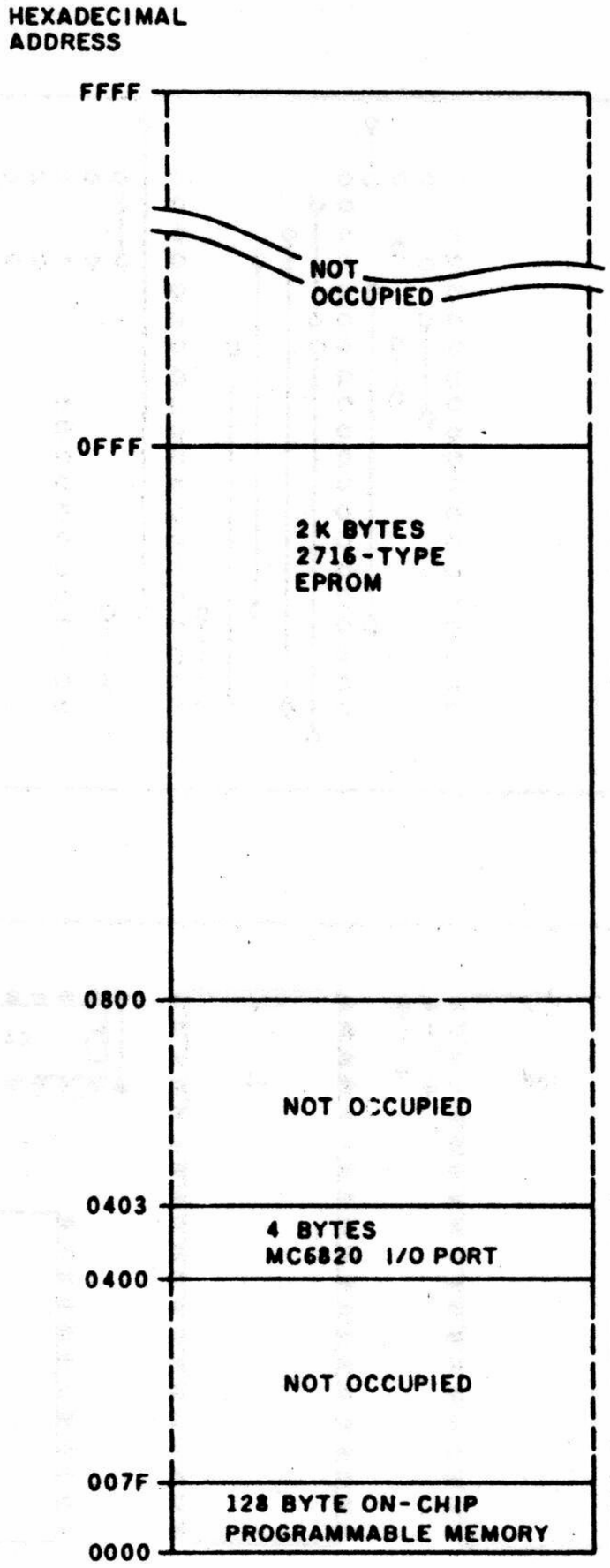
DET TRYCKTA KRETSKORTET

Det tryckta kretskortet mäter 100 x 60 mm med en 24-pinnars kretskortskontakt (2.54 mm rasteravstånd). Kortet är enkelsidigt för att hålla kostnaderna nere. Detta gör det nödvändigt att ansluta ett antal tråd-byglingar på kortets komponentsida. Det tryckta kretskortet, trådbyglingarna och komponenternas placering framgår av figur 3.

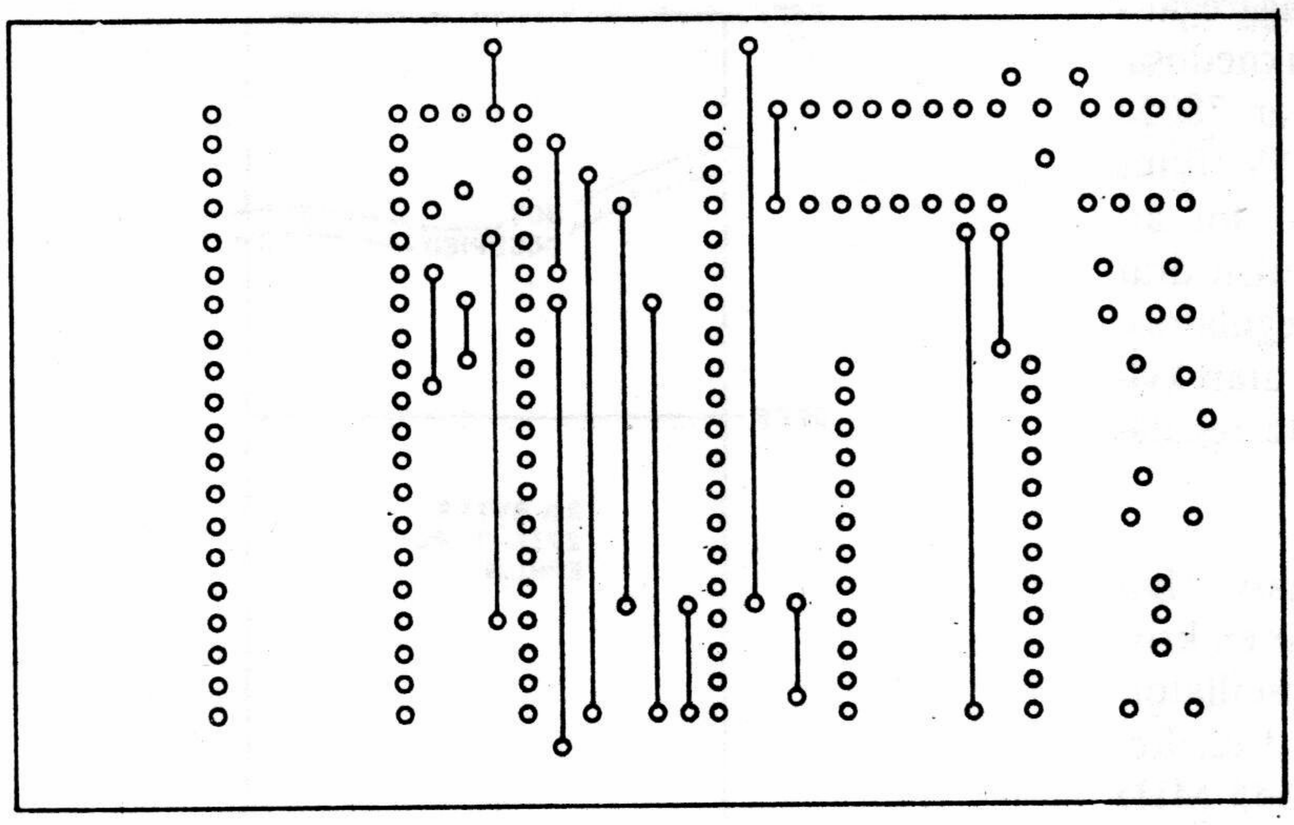
Prototyp-kort i samma storlek, med 24-pinnars kortkontakt, finns att köpa från en mängd tillverkare. Dessa matchande kort är idealiska att bygga anslutningsmoduler till kontrollermodulen.

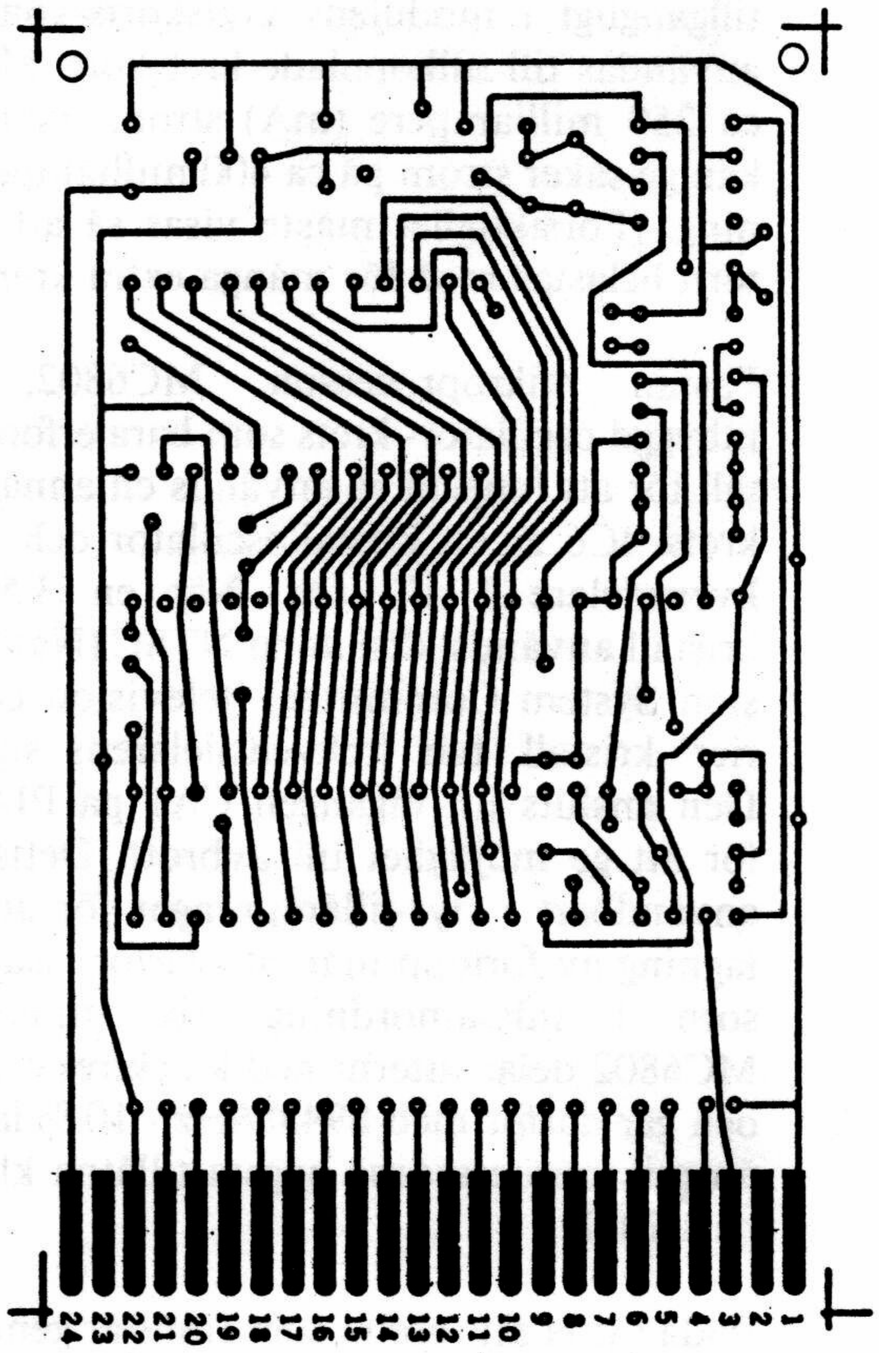
MJUKVARU OCH HÅRDVARU SAMMANKOPPLING

Processorn MC6802 använder en instruktionsuppsättning identisk med MC6800. Vilket som helst av Motorola MC6800 utvecklings-system eller utvärderings kit är lämpliga för att skapa mjukvara på. Ett Motorola D2 kit med 4 Kbytes minne är mycket lämpligt därför att dess adress-område täcker RAM och EPROM adressområdet på kontroller-modulen. Den PIA som användaren av D2 kitet har till buds kan användas till att simulera PIAn på kontroller-modulen. Man kopplar helt enkelt ihop kontakterna mellan korten, och så kan all avlusning av hårdvara och mjukvara utföras via D2 kitet. När systemet fungerar tillfredställande kan programmet överföras till EPROM och kontroller-modulen kan ersätta D2 kitet.

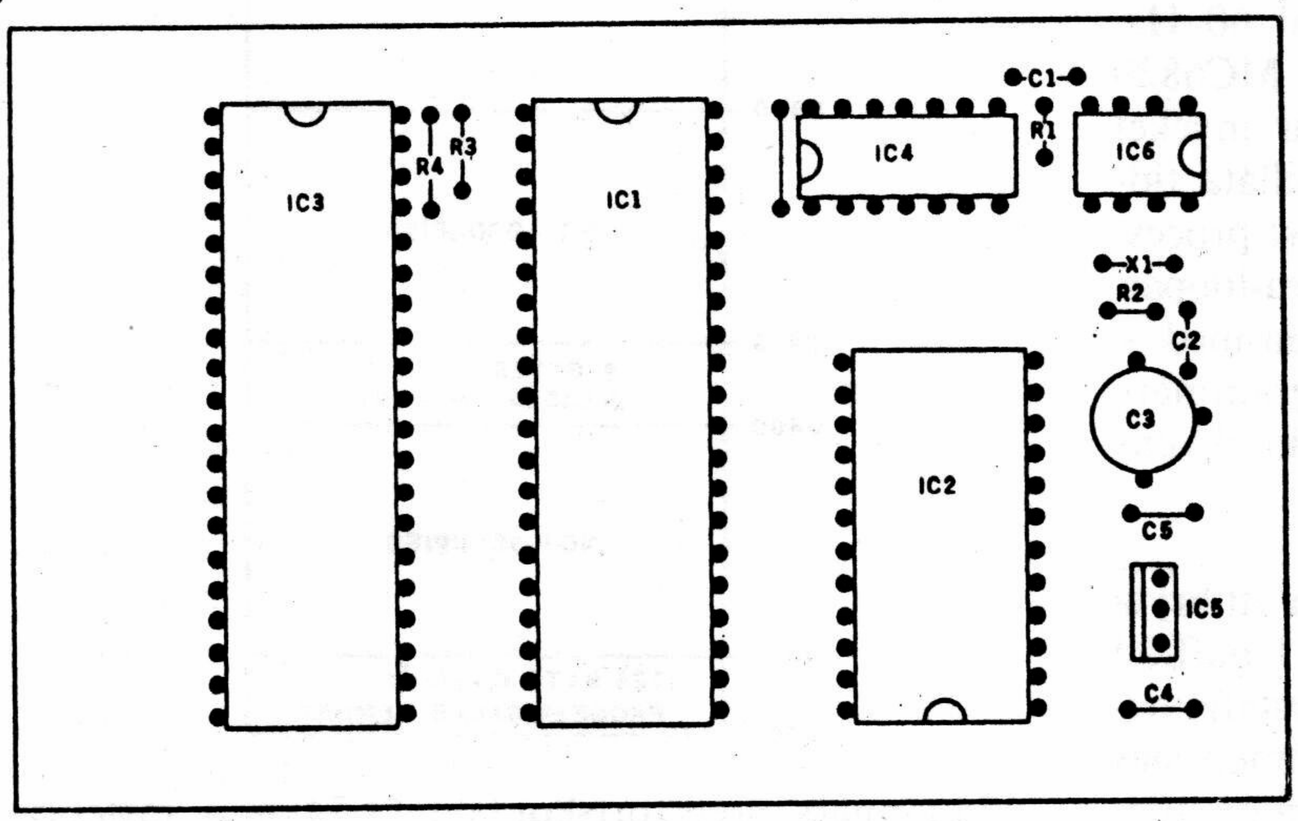


(3b)





(3c)



KRETSKORTSFILM

Klubben har film till kretskortet med 6802-processorn. Skriv och beställ så får Du låna.

De program-ändringar som behövs för att låta utveckla mjukvara för kontroller-modulen på D2 kitet är minimala. Den ändring som behövs i huvudprogrammet är att ändra PIAns adress från hexadecimalt 8004-8007 (på D2 kitet) till hexadecimalt 0400 - 0403 för kontroller-modulen. Detta fordrar bara att två bytes ändras om alla program-referencer till PIAn använder indexerad adressering så att index-registret laddas med från en dubbel-byte konstant i minnet varje gång det används.

En annan möjlig programförändring skulle vara att använda mjukvaru-loopar för att utföra tidtagningsfunktioner. Kontrollern går med en intern klockfrekvens på 894,886 kHz, jämfört med exempelvis D2 kitet, som går med en klockfrekvens på 614,886 kHz. Justeringar av tidtagningsslooparnas exekveringstider är enkel om loop-räknaren laddas från en konstant i minnet varje gång den används. Då behövs bara en enda konstant i minnet ändras, oavsett hur många gånger den används.

Ett antal hopkopplingsbara standard-moduler kan också tillverkas för att hantera de allra vanligaste gränssnitts behoven. Av en 8$:s kalkylator har jag utvecklat och byggt en gränssnittsmodul för att användas tillsammans med en 24-knappars tangentbord och en 8 siffrors, 7 segments LED-(light emitting diode)display. Jag har också byggt en modul (kostnad ca 4$) som kan göras till en 7-bitars A/D (analog till digital) eller D/A (digital till analog) omvandlare. När den är D/A omvandlare använder modulen bara två snören till PIAn, och som A/D omvandlare bara tre snören. Det är lätt att lägga till drivkretsar för reläer och lampor, och omkopplingsbara ingångar.

Standardiserade mjukvaru-rutiner kan användas för att hantera standardgränssnitts-moduler och andra ofta använda mjukvaru- funktioner, t ex en realtidklocka som styrs av 60 Hz avbrottsgeneratorn. Detta tillåter att ett brett fält av olika kontrollers kan skapas med ganska små ansträngningar genom att blanda och anpassa standard maskinvaru-& mjukvaru-moduler.

ANNONSER

SÄLJES

MODEMKORT

Fristående kort med autosvar med A och B kanal
300 Baud
1200/75 Baud
75/1200 Baud
Passar vilken dator som helst
Kretskort med telefonanpassad trafo 300kr per styck

Ian Birchman
Telefon 20 57 19

UTFÖRSÄLJNING
DISKDRIVERS

Några nya, endast demokörda BASF 6106 Single Side, Double Density, Slimline-drivar, med fronten i full höjd.
Servade med Dysan disc-drivetester.
Pris endast 900 kr/st inkl moms o manual.

Jan Nordström Telefon 0752/17094, 13487

1 ST BASF 6106 SS,SD säljes. Knappt använd, endast testad.
Pris 900 kr inkl moms o manual.
Ett antal 68B09 säljes endast 50 kr/st
2 st obestyckade statiska 16 k minneskort säljes för RT-datorns EXORCISER BUSS endast 100 kr.

Lennart Uusitalo
Skrakvägen 20
141 72 Huddinge
Tel : Hem 466628, Arb. 7499637

SÄLJES

■ ■ ■ ■ ■ **SOMMARREA** ■ ■ ■ ■ ■

1 st Seikosha GP100 Printer.
1 st Microline 80 Printer med serieinterface och traktormatningstillsats.
1 st Termoprinter m. sep. nätaggregat.
1 st 48k dynamiskt minneskort.
Billigt !

■ ■ ■ ■ ■ **SOMMARREA** ■ ■ ■ ■ ■

Gunnar Ejemoh
Tel: 08 / 7715562

✦ ✦

GAMLA MPU-LARE TILL SALU.

Efter ett stort sökande efter vart alla gamla, kvarvarande nummer av MPU-laren tagit vägen, lyckades vi lokalisera några undangömda kartonger längst inne i garderoberna hos några medlemmar. Dessa nummer säljs nu till det facila priset av 15:-/styck. Du skriver till klubben och beställer dina nummer, inget annat sätt tolereras, poststämpelns datum gäller som köbricka. Det enda som till kommer är brevpostförskottsavgift, f n 6.75.

Klubbens lager av MPU-lare per den 18/5-84:
nr 4/81 55st., nr 1/82 3 st., nr 2/82 80st., nr 1/83 10st., nr 2/83 13st.

KLUBBDATORN

Nu när vi har haft en klubbdator i drygt ett år får vi rikta ett gemensamt tack till ELFA Radio & Television AB, som har ställt utrustningen en PRIMAL dator PD2000 till vårt förfogande.

I dag är det omkring 90 medlemmar som har tagit ut password och c:a 50 kör kontiunerligt på den. Vi registrerar alla som loggar in på datorn för att få en uppfattningen om hur livlig trakfiken är. När vi drog i gång projektet ramlade det in mycket mer program och andra bidrag på datorn än vad det numera gör, m.a.o. LÄMNA bidrag på den istället för att som nu bara läsa vad som dyker upp.

NI som ibland glömmer att använda 'ESC'-tangenten när ni slutar, det finns EJ autologgut på datorn, detta innebär att DU måste logga ut från systemet själv INNAN du lägger på luren. OBS ! systemet bryter själv linjen vid rätt utloggning.

Nu följer en liten beskrivning av klubbdatorn:

Handledning för klubbdatorn.

Del 1

Kort simulering av körning.

När du har fått kontakt mellan modemen (CCITT V.21 300 BAUD, full duplex, 8 databitar, ingen paritet) trycker du på tangenten 'RETURN' och datorn svarar med:

```
OS-9 Fleranvändarsystem   Level II  V1.2   84/06/30 13:40:11

Namn?:                                    "Här skriver du ditt namn"
Lösenord:                                 "Här skriver du dit password"

Därefter svara operativsystemet med att du är inloggad:

Process #04 inloggad    84/06/30 13:40:53
Välkommen!

*****************************************
* Välkommen till PD68's klubbdator !    *
*****************************************

SPONSOR: E L F A  Radio & Television AB

OBS!!!  GLÖM  EJ  ATT  LOGGA  UT  MED
‹ESCAPE› tangenten ($1B). OS9 svarar
med "eof".


1...Flex
2...OS-9
3...Mailbox
4...Annonser
5...Info

Välj: 3                                   "Här väljer du vilket bibliotek
                                           du vill jobba med; i detta fall
                                           biblioteket 'MAILBOX'."
```

```
/D1/MAILBOX                                  "Klubbdatorn svarar med valet."

Shell                                        "Detta är kommando inte-
                                              pretatorn som du arbetar med."

OS9:dir                                      "Här har kommandot 'DIR' givits
                                              för att se vad som finns i vår
                                              brevlåda."

                                             "Detta är katalogen som OS9
                                              skriver ut den:"

    directory of .  13:41:20
Info                Info.Attribut   Info.Datum       Info.Bibliotek
winchester.fråga                    OPinfo          Info.Tmode
TERMINAL.CTRL.TECKEN                ASMB.PGM.HJÄLP.t.STYRELSEN
MPUlaren.2.84   A.T.L.igång.igen                    TIME09.FRÅGA
Fråga.till.red  TILL.ALLA
OS9:eof                                      "Här har 'ES givits för
                                              avslut av körningen och OS9
                                              svarade med 'eof'."
```

Del 2

Vissa användbara kommandon

Kommandon skrivs på samma sätt som i FLEX (t.ex DIR ger fil katalogen). Vilka kommando som finns kan man se med HELP. En av skillnaderna är att det finns flera kataloger. För att flytta till andra kataloger och diskenheter används tecknet "/" i kommando. T.ex. för att läsa filen "opinfo" under katalogen "MAILBOX" på disk 1 (D1) skriv:
LIST /D1/MAILBOX/opinfo

OBS! Utskriften stannar efter 24 rader, starta igen med spacetangenten.

Kommandot "CHD ‹katalognamn›" byter katalog och för att läsa en fil i den katalog man befinner sig i, behöver bara LIST ‹filnamn› skrivas.

För att ta reda på vilken katalog du befinner dig i, kan du använda kommandot "PWD".

Om du vill lägga in ett meddelande kan du använda kommandot:
BUILD ‹filnamn›.
Datorn svarar med frågetecken. Skriv in meddelandet (en eller flera rader). Avluta med tom rad, så återkommer OS9, och ditt meddelande finns nu som en textfil i det namn du skrev efter BUILD.
Kontrollera med LIST ‹filnamn› hur det blev.

Utloggning från systemet sker med ‹ESC› tangenten.
Datorn svarar med eof.

GLÖM EJ LOGGA UT INNAN DU BRYTER LINJEN. !!!

Kontrolltecken under OS9.

ctrl-A Visa senaste kommandoraden.
ctrl-C Avbryt.
ctrl-H Bakåtstegning.

ctrl-E Avsluta program.
ctrl-W Stoppa utmatningen.
        Starta med space-tangenten.
ctrl-X Radera raden.
-P   Ingen prompter ("OS9:").
P   Inkoppling av prompter.

I filnamn tillåts A-Ö, 0-9, punkt och understrykning. Max 29 tecken. Vissa kommandon är spärrade eftersom du kör direkt under operativqsystemet OS9. För att få vidare info om kommandon skriv HELP.

Uppdaterad:

83.06.15 SER
84.01.21 BER
84.05.03 BER

Information om kommandot ATTR.

Syntax ATTR ‹path› " ' ‹förkortningar av skyddskoder› ' "

Detta kommando används för att visa eller ändra en fils skyddskod. Kommandot ges genom att skriva "ATTR" följt av en beskrivning på var filen finns - ‹path› samt en lista på de skyddskoder som skall ändras. En viss skyddskod sätts genom att skriva dess förkortning. Den tas bort genom att sätta ett minustecken före förkortningen. Endast de skydds-koder som anges påverkas. Om inga skyddskoder anges visas de koder som gäller för filen. Koden kan endast ändras på egna filer. Användare 0, systemoperatören, kan ändra skyddskoder för alla filer oavsett ägare.

Följande skyddskoder finns att tillgå:

```
 d = Katalogfil.
 s = Filen kan delas med andra användare.
pe = Exekvering tillåten --- samtliga användare.
pw = Skrivning tillåten ---- samtliga användare.
pr = Läsning tillåten ------ samtliga användare.
 e = Exekvering tillåten --- ägaren.
 w = Skrivning tillåten ---- ägaren.
 r = Läsning tillåten ------ ägaren.
```

Exempel:

```
attr minfil -pr -pw          tar bort skyddskoderna pr och pw

attr minfil r w e pr pw pe   sätter alla skyddskoder utom d och s

attr datalogg                ger aktuella skyddkoder för
-s-wr-wr                      filen "datalogg
```

Information om kommandot date.

Syntax: DATE "t"

Detta kommando visar systemets datum och om optionen "t" ges även tiden.

Exempel:

```
date                         datum visas på terminalen
date t                       datum och tid visas på terminalen
```

Information om kommandot dir.

Syntax: DIR "e" "x" "‹path›"

Ger en formatterad lista på de filer som finns i en katalogfil. Om inga parametrar ges listas de filer som den aktuella datakatalogen innehåller. Optionen "x" visar innehållet i den aktuella kommandokatalogen. Om "path" angetts visas detta i utskriften av katalogen.

Optionen "E" ger en utskrift av alla data om en viss fil:

Ägare, datum och tid för senaste ändring, tillstånd (skydd), adress och storlek.

Exempel:

dir	visar den aktuella datakatalogen
dir x	visar den aktuella kommandokatalogen
dir x e	d:o med alla data om filerna
dir ..	visar innehållet i närmast högre datakatalog
dir flex	visar innehållet i datakatalogen "flex"
dir e flex	d:o men med fullständig information

Information om kommandot TMODE.

Syntax TMODE ".‹pathnum›" "‹arglist›"

Detta kommando används för att visa eller ändra parametrarna för den egna terminalen.

Om inga argument ges visas det värde varje parameter har för ögonblicket. Finns ett eller flera argument på kommandoraden ändras dessa parametrar. Kommatecken eller mellanslag används för att skilja olika argument åt. En punkt (.) eller ett tal (0, 1 el. 2) kan användas för att ange vilken väg (path) som skall påverkas. Saknas denna uppgift används standard inmatningsväg.

"Startvärde" är det värde som används om inget annat angetts.

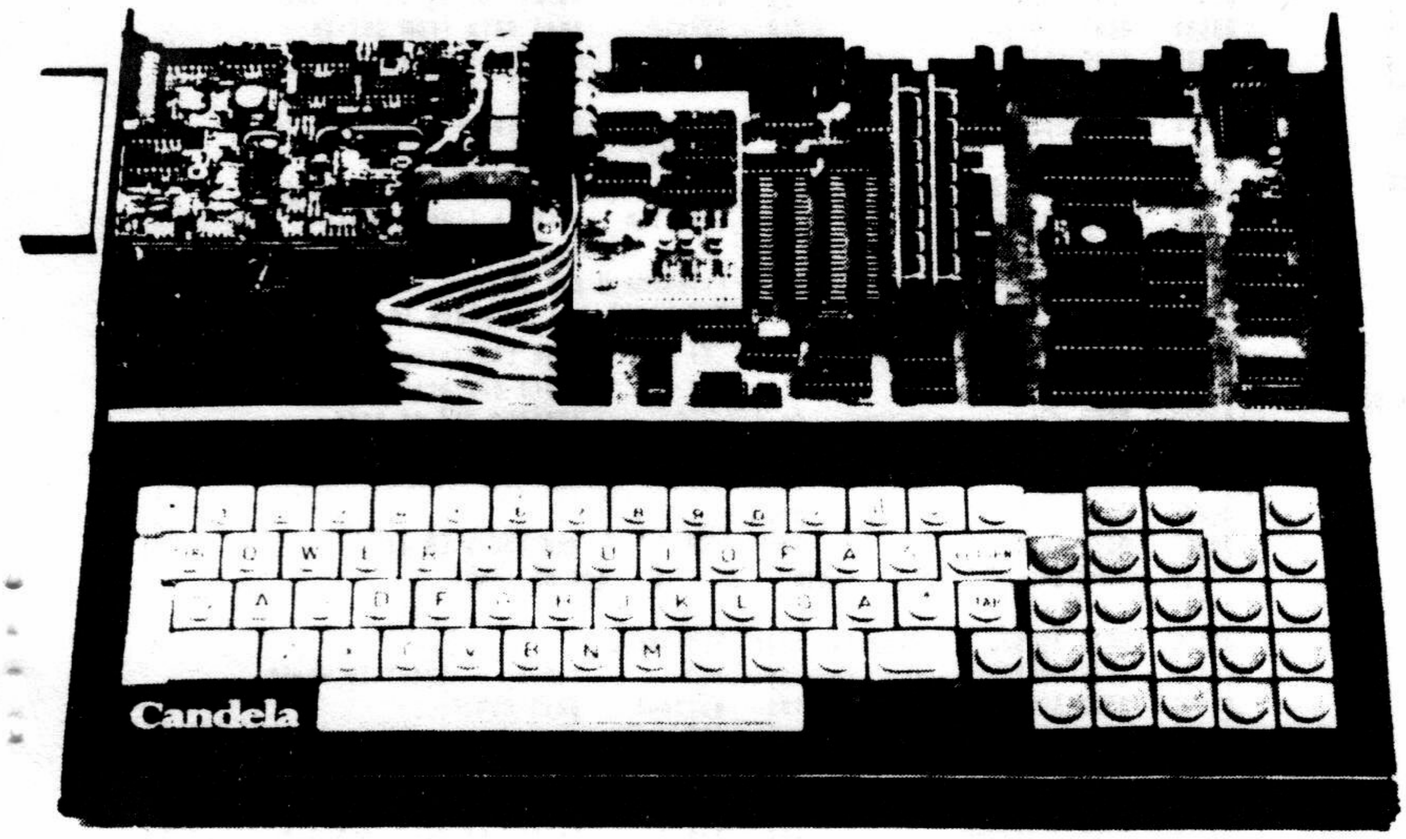
upc	Endast versaler. Gemener omvandlas automatiskt till versaler.
-upc	Versaler och gemener tillåts (startvärde)
bsb	Radering vid backning. Raderingen sker genom att eka sekvensen backa, mellanslag, backa (startvärde).
-bsb	Radera inte vid backning. Ekar endast ett bakåtsteg.
bsl	Backa över raden. En hel rad "raderas" genom att sända ett antal sekvenser med backa-mellanslag-backa tills raden har raderats. Avsett för bildskärmsterminaler (startvärde).
-bsl	Backa inte över raden. Raden "raderas" genom att ge ny rad (retur, radframmatning). Avsett för skrivande terminaler.
echo	Inkommande tecken ekas tillbaka till terminalen (startvärde).
-echo	Inget eko.

```
 lf     Automatisk radframmatning i samband med "retur" (startvärde).
-lf     Ingen automatisk radframmatning.

 pause  Utmatningen till bildskärmen avbryts efter en full skärm. An-
        talet rader som utgör "en full skärm" definieras av parametern
        "pag". Återstart sker med en godtycklig tangent (startvärde).
-pause  Kontinuerlig utmatning.


OBSERVERA: I beskrivningen nedan innebär "n" (ex. null=n) ett
           decimalt värde och "h" (ex. bsp=h) ett hexadecimalt värde.

null=n     Antalet utfyllnadstecken (hex 00) efter "retur". Används för
           att ge vissa skrivare tillräklig tid för vagnreturen.
           Startvärde: 0.

pag=n      Sätter sidlängden på bildskärmen till "n" rader. Används i
           "pausläget". Se ovan.

bsp=h      Det (inkommande) tecken som används för att backa.
           Startvärde: 08.

bse=h      Det tecken som matas ut vid backning. Startvärde: 08.

del=h      Det tecken som används för att radera en rad. Startvärde: 18.

bell=h     Det tecken som ger ljudsignal från terminalen. Startvärde: 07.

eor=h      Radsluttecken (retur). Startvärde: 0D.

eof=h      Filsluttecken (vid inmatning). Startvärde: 1B.

type=h     Kontrollord till den seriella porten. Sätter parity, ordlängd,
           antal stoppbitar etc. Startvärde: 15.

reprint=h  Repetera sista raden. Startvärde: 04.

dup=h      Kopiera förra (kommando)raden. Startvärde: 01.

psc=h      Paustecken (stoppar utmatningen). Startvärde: 17.

abort=h    Avbrottsignal till pågående program. Startvärde: 03 (ctl-C).

quit=h     Avbryt programmet, återgå till shell. Startvärde: 05 (ctl-E).

Exempel:

tmode -upc lf null=4 bse=1F pause
          versaler och gemener, 4 utfyllnadstecken, eka hex 1F
          vid backning, aktivera pausfunktionen

tmode pag=24 pause -echo bsp=8
          24 rader på bildskärmen, aktivera pausfunktionen, radera
          hela raden genom "backa-mellanslag-backa", inget eko av
          inkommande tecken, hex 08 = bakåtstegning från terminalen.
```

KLUBBDATORGRUPPEN
Anders Karlsson och Thomas Bergstam

Ny SVENSK dator med 6809

CANDELA



Datorn är framtagen av företaget Didact tillsammans med Liber. Dess användningsområde är utbildning och styr- och mät-applikationer. Candela innehåller: 6809 processor med operativsystemet OS-9. 128 kB RAM (utbyggbart till 384 kB internt). Tre tums mikrofloppy (320kB) inbyggd. Basic 09 och Assembler i ROM. Pascal, C, mm finns som PROM-kassetter. För mätändamål är CANDELA försedd med A/D och D/A omvandlare. Färdiga programrutiner för dessa omvandlare medföljer.

Grafiken består av tre olka moder med olika upplösning: 256X256, 512X256 eller 512X384 där varje punkt är adresserbar. Ett antal färger eller gråskalenivåer ger en stor variationsmöjlighet.

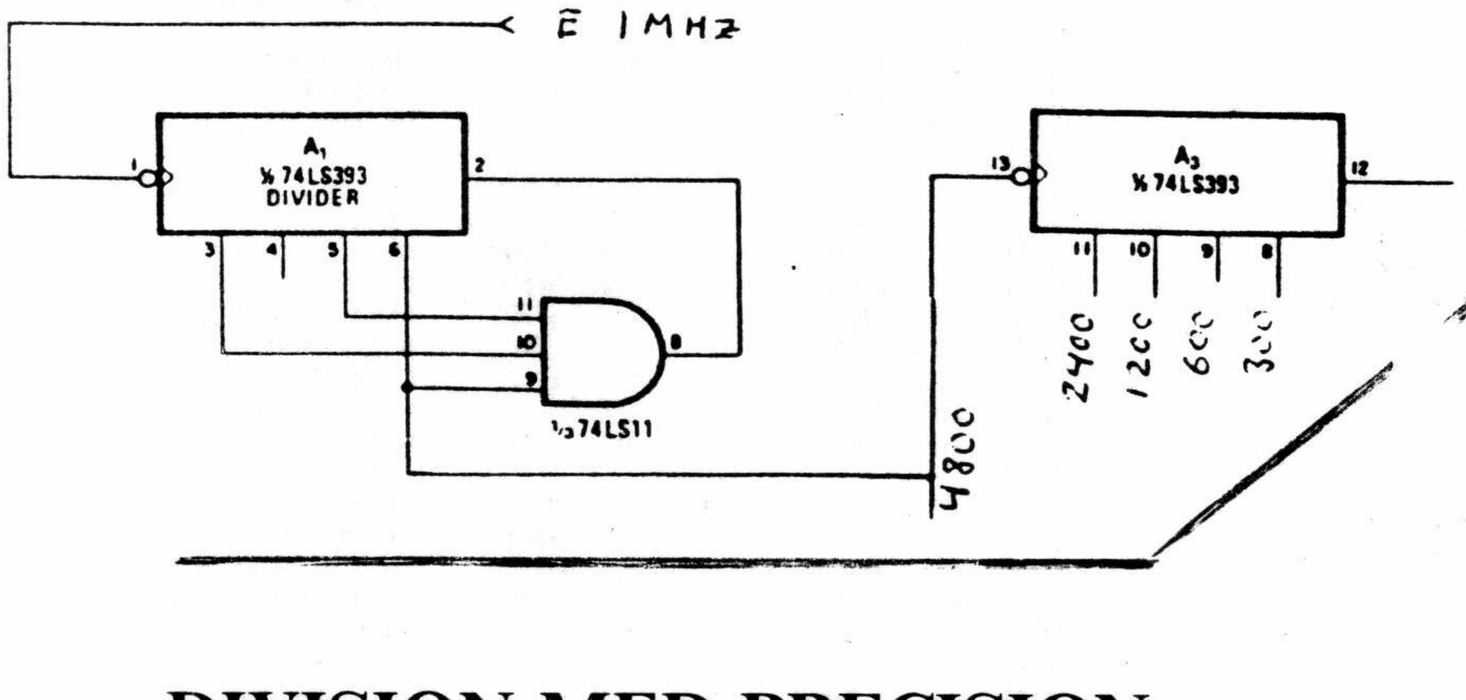
Candela kan också använda Flex. Priset lär vara ca 15000 kr. Då behöver man bara en monitor eller vanlig TV för att få det hela att snurra.

En förenklad variant för Flex kommer antagligen att tas fram för försäljning. Den skulle bestå av två kort där det ena innehåller 6809, ACIA, PIA, Timer, EPROM (för vissa Flexkommandon). Det andra innehåller 128kB Ram och samma grafik som beskrivits ovan.

SER

BAUDRATEGENERATOR

Det här är ett förslag på en enkel och billig baudrategenerator, som kräver en klocka på 1 Mhz. Först delas frekvensen i Dividerare A1 (binärräknare) med 13, som ger en baudrate på 16 x 4800. Nu blir det inte exakt 4800 baud untan ca 7.7 enheter för högt. Felet är dock ca 1.5 promille, så det får anses som försummbart. I andra halvan, A3, får man sedan de andra lägre hastigheterna, se figuren .



DIVISION MED PRECISION

Detta program kan göra en lång division med precision. Antalet decimaler kan man välja obegränsat, t.ex hundra decimaler. Programmet frågar först efter talet som ska divideras och därefter med vad det ska divideras. Sista frågan är " Hur många decimaler ?" man vill ha. Finessen, att dividera med noll, vägrar programmet att utföra. Jag hoppas att sjäva grund rutinen kan vara användbar i något sammanhang, förutom för rena matematikers talintressen.
Anm: Programmet kan köras i TSC BASIC förutom rad 1060 som nu använder PRINT

```
10 REM* OBEGRÄNSAD HELTALSDIVISION
20 REM*
30 REM* BYTE FEB -79 SID. 154
40 REM* INKNAPPAT AV BERNT SVENSSON
50 REM*
90 PRINT
100  PRINT "DET HÄR PROGRAMMET KAN GÖRA EN LåNG"
105  PRINT "DIVISION MED PRECISION"
107 PRINT
110 INPUT "VILKET TAL SKA DIVIDERAS",X
120 PRINT "MED VILKET TAL SKA ";X;" DIVIDERAS";
125 INPUT Y
126  IF Y=0 THEN GOTO 160
130 INPUT "HUR MåNGA DECIMALER",N
140 GOSUB 1020
150  GOTO 107
160  PRINT "DIVISION MED 0 ÄR ODEFINIERAT"
170 STOP
175 REM
1020 Q=INT(X/Y)
1030 PRINT Q;
1035 PRINT ".";
1036 A=(X-Q*Y)
1038 IF A=0 THEN PRINT 0:GOTO 107
1040 X=A*10
1050 Q=INT(X/Y)
1060 PRINT USING '£',Q;
1070 N=N-1
1080 IF N>0 THEN GOTO 1040
1090 RETURN
1100 END
```

OS9 ASSEMBLERPROGRAMM

Här kommer de första assemblerprogrammen för OS9 i vår tidning. De är bägge skrivna av Thomas Bergstam. Dname ändrar namnet på en skiva. Send är skrivet för att överföra filer till klubbdatorn. Bägge källfilerna finns på klubbdatorn under biblioteket OS9.

```
Microware OS-9 Assembler 2.1   07/01/84 11:27:08                Page 001
DName - name or rename disk

00001
00002                           nam   DName
00003                           ttl   name or rename disk
00004          ***************************
00005          *    DName by               *
00006          *    Thomas Bergstam        *
00007          *    4 Februari 1984        *
00008          *                           *
00009          *    Name or rename disk.   *
00010          *    Does NOT give error    *
00011          *    message if disk is     *
00012          *    write protected!!      *
00013          ***************************
00014          *
00015          *  OS9 Definitions
00016          *
00017   0006           F$Exit   equ   $06         terminate process
00018   0084           I$Open   equ   $84         open existing file
00019   008B           I$ReadLn equ   $8B         read line of ASCII data
00020   0089           I$Read   equ   $89         read data
00021   0001           READ.    equ   %00000001   file access mode
00022   0002           WRITE.   equ   %00000010   file access mode
00023   008C           I$WritLn equ   $8C         write line of ACII data
00024   008A           I$Write  equ   $8A         write data
00025   0010           Prgrm    equ   $10         type value
00026   0001           Objct    equ   $01         language value
00027   0080           ReEnt    equ   %10000000   attribute flag
00028   000D           CR       equ   $0D         carriage return
00029   0020           SPACE    equ   $20         space
00030   000A           LF       equ   $0A         line feed
00031   0001           Stdout   equ   1          standard output path
00032
00033          ***************************
00034          *  Module header
00035
00036   0000 87CD0118          mod   Size,Name,Prgrm+Objct,ReEnt+1,Entry ,Memor
00037
00038          ***************************
00039          * Module name, version copyright
00040
00041   000D 444E616D  Name     fcs   /DName/     module name
00042   0012 02        Version  fcb   2           version
00043   0013 54423139           fcs   /TB1984A2/
00044
00045          ***************************
00046          *  Memory rmb's
00047
00048 D 0000           Nameptr  rmb   2           file name pointer
00049 D 0002           Newname  rmb   33          new name buffer
00050 D 0023           Oldname  rmb   33          old name buffer
00051 D 0044           Sector   rmb   31          place to put sector
00052 D 0063           Filename rmb   224         file name
00053 D 0143           Memory   equ   .
00054
00055          ***************************
00056          *  Start of program
00057
00058   001B C101      Entry    cmpb  #$01        if no parameter
00059   001D 102700B3           lbeq  Errmsg      print error and jump out
00060
```

```
Microware OS-9 Assembler 2.1   07/01/84 11:27:16                Page 002
DName - name or rename disk

00061   0021 A680      Entry1   lda   ,X+        delete space in command
string
00062   0023 8120               cmpa  #SPACE     is it space
00063   0025 27FA               beq   Entry1     no, do it again
00064   0027 301F               leax  -1,X       back X one
00065   0029 A684               lda   ,X         check if any '/'
00066   002B 812                cmpa  #'/        if no slash,
00067   002D 102600A3           lbne  Errmsg     print error msg and jump  out
00068   0031 9F00               stx   Nameptr    save name pointer
00069
00070   0033 A680      Entry2   lda   ,X+        check length of paramete
r
00071   0035 810D               cmpa  #CR        is it car.ret.
00072   0037 26FA               bne   Entry2     no, loop again
00073   0039 86C0               lda   #'Ê+$80    set bit of last char.
00074   003B A71F               sta   -1,X       and store it ack
00075   003D 9E00               ldx   Nameptr    address of pathlist
00076   003F 8601               lda   #READ.     access mode
00077   0041 103F84             os9   I$Open     open a path to a device
00078   0044 10250089           lbcs  Error      branch if error
00079   0048 30C844             leax  Sector,U   address to store data
00080   004B 108E00FF           ldy   #255       number of bytes to read
00081   004F 103F89             os9   I$Read     read data from device
00082   0052 257D               bcs   Error      branch if error
00083
00084   0054 308D00A0           leax  Msg2,pcr   address of data to write
00085   0058 8601               lda   #Stdout    path number
00086   005A 108E000F           ldy   #15        number of bytes to write
00087   005E 103F8C             os9   I$WritLn   write a line of text
00088
00089   0061 31C863             leay  Filename,U ptr to name in sector
00090   0064 30C823             leax  Oldname,U  ptr to place to copy name
00091   0067 A6A0      Entry3   lda   ,Y+        read character in sector
00092   0069 A780               sta   ,X+        store character in works pace
00093   006B 4D                 tsta             reached end of name
00094   006C 2AF9               bpl   Entry3     no, loop again
00095   006E 847F               anda  #$7F       set bit
00096   0070 A71F               sta   -1,X
00097   0072 860D               lda   #CR        and add a CR at
00098   0074784                sta   ,X         end of line
0099
00100   0076 30C823    Prepare  leax  Oldname,U
00101   0079 108E0050           ldy   #80        number of bytes to write
00102   007D 8601               lda   #Stdout    path number
00103   007F 103F8C             os9   I$WritLn   write a line of text
00104
00105   0082 308D0081           leax  Msg3,pcr   address of data to write
00106   0086 108E000E           ldy   #14        number of data to write
00107   008A 8601               lda   #Stdout    path number
00108¿  008C 103F8C             os9   I$WritLn   write a line of text
00109
00110   008F C620               ldb   #32        max length of name
00111   0091 30C863             leax  Filename,U address of name
00112   0094 4F                 clra             clear entire
00113   0095 A780      Loop     sta   ,X+        name space in sector
00114   0097 5A                 decb             decrement counter
00115   0098 26FB               bne   Loop   ¿   no, loop again
00116
00117   009A 3042               leax  Newname,U  address to store data
00118   009C 108E0021           ldy   #33        max number of bytes to read +
00119   00A0 4F                 clra             Stdinpath input path number
00120   00A1 103F8B             os9   I$ReadLn   read a line of text
```

```
Microware OS-9 Assembler 2.1   07/01/84 11:27:25                Page 003
DName - name or rename disk

00121   00A4 A684               lda   ,X         is first char
00122   00A6 810D               cmpa  #CR        a carriage return
00123   00A8 2726               beq   Exit       yes, branch out
00124
00125   00AA 1F20               tfr   y,d        length of text
00126   00AC 5A                 decb
00127   00AD 31C863             leay  Filename,U ptr to name in sector
00128   00B0 3042               leax
00129   00B2 A680      Loop1    lda   ,X+        copy new name
00130   00B4 A7A0               sta   ,Y+        to place in sector
00131   00B6 5A                 decb
00132   00B7 26F9               bne   Loop1      loop til end of text
00133   00B9 8A80               ora   #$80       set bit 8 on
00134   00BB A73F               sta   -1,Y       the last char.
00135
00136   00BD 9E00               ldx   Nameptr    address of pathlist
00137   00BF 8602               lda   ¿ #WRITE.   access mode
00138   00C1 103F84             os9   I$Open     open a path to a file
00139   00C4 30C844             leax  Sector,U   address of data to write
00140   00C7 108E00FF           ldy   #255       number of data to write
00141   00CB 103F8A             os9   I$Write    writ data to file
00142   00CE 2501               bcs   Error      branch if error
00143   00D0 5F        Exit     clrb             no error code
00144   00D1 103F06    Error    os9   F$Exit     terminate the process
00145
00146   00D4 308D000B  Errmsg   leax  Msg1,pcr   address of data to write
00147   00D8 8601               lda   #Stdout    path number
0148    00DA 108E0050           ldy   #80        number of bytes to write
00149   00DE 103F8C             os9   I$WritLn   write a line of text
00150   00E1 20ED               bra   Exit       jump out
00151
00152          *  End of program
00153          ***************************
00154          *  Strings
00155
00156   00E3 5573653A  Msg1     fcc   'Use: dname  /device '
00157   00F7 0D                 fcb   CR
00158   00F8 0A        Msg2     fcb   LF
00159   00F9 4F6C6420           fcc   'Old diskname: '
00160   0107 4E657720  Msg3     fcc   'New diskname: '
00161
00162          ***************************
00163
00164   0115 2FE747             emod             module CRC
00165
00166   0118           Size     equ   *          size of module
00167
00168                           end


00000 error(s)
00000 warning(s)
$0118 00280 program bytes generated
$0143 00323 data bytes allocated
$025F 00607 bytes used for symbols
```

```
Microware OS-9 Assembler 2.1   07/01/84 11:36:36                    Page 001
send - send text file

00001                           nam   send
00002                           ttl   send text file
00003
00004         ********************************
00005         *                              *
00006         * Program 'send' by Thomas Bergstam *
00007         * June 23 1984 Version 1       *
00008         *                              *
00009         ********************************
00010
00011                           ifp1                use os9defs
00013                           endc
00014
00015   0007            BELL    equ   $07          bell
00016   0020            SPACE   equ   $20          space
00017   000A            LF      equ   $0A          line feed
00018   000D            CR      equ   $0D          carriage return
00019   0000            StdIn   equ   0            standard input path
00020   0001            StdOut  equ   1            standard output path
00021   0002            ErrOut  equ   2            error output path
00022
00023         ********************************
00024         * Module header
00025         *
00026
00027   0000 87CD018E           mod   Size,Name,Objct+Prgrm,ReEnt+1,Entry,Memor
00028
00029         *
00030         *
00031         ********************************
00032         *
00033         *
00034
00035   000D 73656EE4   Name    fcs   /send/       module name
00036   0011 01         Version fcb   1
00037   0012 42455231           fcs   /BER1984A1/
00038
00039 D 0000            Nameptr rmb   2            pointer to filename
00040 D 0002            Filpath rmb   1            path number for file
00041 D 0003            Devpath rmb   1            path number for device
00042 D 0004            Stat    rmb   1            device descriptor
00043 D 0005            Upc     rmb   1            upper case
00044 D 0006            Bso     rmb   1            backspace
00045 D 0007            Dlo     rmb   1            delete
00046 D 0008            Eko     rmb   1            echo
00047 D 0009            Alf     rmb   1            auto LF
00048 D 000A            Nul     rmb   1            end of line null count
00049 D 000B            Pau     rmb   1            pause
00050 D 000C            Pag     rmb   1            lines per page
00051 D 000D            Bsp     rmb   1            backspace char.
00052 D 000E            Del     rmb   1            delete line character
00053 D 000F            Eor     rmb   1            end of record char.
00054 D 0010            Eof     rmb   1            end of file char.
00055 D 0011            Rpr     rmb   1            reprint line char.
00056 D 0012            Dup     rmb   1            duplicate last line char.
00057 D 0013            Psc     rmb   1            pause char.
00058 D 0014            Int     rmb   1            keyboard interrupt char.
00059 D 0015            Qut     rmb   1            keyboard abort char.
00060 D 0016            Bse     rmb   1            backspace echo char.
00061 D 0017            Ovf     rmb   1            line overflow char.


Microware OS-9 Assembler 2.1   07/01/84 11:36:44                    Page 002
send - send text file

00062 D 0018                    rmb   12           the rest
00063 D 0024            String  rmb   255          string to trans.
00064 D 0123                    rmb   1024
00065 D 0523            Memory  equ   .
00066
00067         ********************************
00068         * Start of program
00069         *
00070
00071   001B A680       Entry   lda   ,X+          get command line
00072   001D 8120               cmpa  #SPACE       is it a space
00073   001F 27FA               beq   Entry        yes; loop back
00074   0021 810D               cmpa  #CR          is it a carriage return
00075   0023 1027013A           lbeq  Err          yes; go print error message
00076   0027 3082               leax  ,-X          back up one
00077 ¿ 0029 9F00               stx   Nameptr      save pointer
00078   002B A680       Entry2  lda   ,X+
00079   002D 810D               cmpa  #CR          if no device...
00080   002F 271D               beq   Default      jump to set default
00081   0031 8120               cmpa  #SPACE       is next char. space
00082   0033 26F6               bne   Entry2       no loop bak again
00083   0035 A680       Entry21 lda   ,X+
00084   0037 8120               cmpa  #SPACE
00085   0039 27FA               beq   Entry21
00086   003B 3082               leax  ,-X          back up one
00087   003D A684               lda   ,X           check next char.
00088   003F 810D               cmpa  #CR
00089   0041 270B               beq   Default
00090   0043 812F               cmpa  #'/          is it beginning of devic
e name
00091   0045 10260118           lbne  Err          if not; something is wrong
00092   0049 2007               bra   Entry3       jump and use device on command
00093
00094   004B 2F54B1     Def     fcs   '/T1'        default to this device
00095
00096   004E 308DFFF9   Default leax  Def,pcr      set default device
00097   0052 8603       Entry3  lda   #UPDAT.      read and write
00098   0054 103F84             os9   I$Open       open device
00099   0057 102500A0           lbcs  Exit         if not found jump out
00100   005B 9703               sta   Devpath      save path number
00101   005D 3044               leax  Stat,U       place for device decriptor
00102   005F 5F                 clrb               SS.OPT option ode
00103   0060 103F8D             os9   I$GetStt    get status o device
00104   0063 0F08               clr   Eko          no echo
00105   0065 0F09               clr   Alf  ¿       no line feed
00106   0067 0F0B               clr   Pau          no pause
00107   0069 103F8E             os9   I$SetStt    put status to device
00108
00109   006C 9E00               ldx   Nameptr      get pointer to filename
00110   006E 8601               lda   ¿ #READ.
00111   0070 103F84             os9   I$Open       open file
00112   0073 10250084           lbcs  Exit         if error; jump out
00113   0077 9702               sta   Filpath      save pathnumber
00114
00115   0079 308D00F4   Synk    leax  Hello,pcr
00116   007D 108E0006           ldy   #HelloLen
00117   0081 9603               lda   Devpath
00118   0083 103F8A             os9   I$Write      write "HELLO"
00119   0086 30C824             leax  String,U     place to store in text
00120   0089 108E0008           ldy   #SendLen
00121   008D 103F89             os9   I$Read


Microware OS-9 Assembler 2.1   07/01/84 11:36:54                    Page 003
send - send text file

00122   0090 C608               ldb   #SendLen
00123   0092 318D00E1           leay  Sendfil,pcr
00124   0096 103F11             os9   F$CmpNam    compare the strings
00125   0099 25DE               bcs   Synk        if not ok, hen loop back
00126
00127   009B 308D00E0           leax  Synksv,pcr
00128   009F 108E0009           ldy   #SynkLen
00129   00A3 103F8A             os9   I$Write     write "SYNKSVAR"
00130
00131   00A6 30C824     Trans   leax  String,U
00132   00A9 9603               lda   Devpath
00133   00AB 108E0002           ldy   #2
00134   00AF 103F8B             os9   I$ReadLn
00135   00B2 A684               lda   ,X
00136   00B4 813E               cmpa  #'>
00137   00B6 265D               bne   Nonsync     program out of sync
00138   00B8 30C824             leax  String,U
0019    00BB 108E00FF           ldy   #255
00140   00BF 9602               lda   Filpath
00141   00C1 103F8B             os9   I$ReadLn    read one line of the file
00142   00C6 9608               lda   Befpath
00144   00C8 103F8A             os9   I$Write     write the line from the file
00145   00CB 308D000B           leax  Dot,pcr
00146   00CF 108E0001           ldy   #DotLen
00147   00D3 8601               lda   #StdOut
00148   00D5 103F8A             os9   I$Write     write '.'
00149   00D8 20CC               bra   Trans       loop back
00150
00151   00DA 2E         Dot     fcc   /./
00152   0001            DotLen  equ   *-Dot
00153   00DB 656F66     Eoftx   fcc   /eof/
00154   00DE 0A0D               fcb   LF,CR
00155   0005            EoftxLen equ  *-Eoftx
00156
00157   00E0 308D00A4   Eofile  leax  Endof,pcr
00158   00E4 108E0003           ldy   #EndofLen
00159   00E8 9603               lda   Devpath
00160   00EA 103F8A             os9   I$Write     write '/*' to device
00161   00ED 308DFFEA           leax  Eoftx,pcr
00162   00F1 108E0005           ldy   #EoftxLen
00163   00F5 8601               lda   #StdOut
00164   00F7 103F8A             os9   I$Write     write 'eof'
00165   00FA 5F                 clrb              no error code
00166
00167   00FB 103F06     Exit    os9   F$Exit      terminate the program
00168
00169   00FE 0A07       Nonsy   fcb   LF,BELL
00170   0100 50726F67           fcc   /Program out of sync!/
00171   0114 0D                 fcb   CR
00172   0017            NonsLen equ   *-Nonsy
00173
00174   0115 308DFFE5   Nonsync leax  Nonsy,pcr
00175   0119 8601               lda   #StdOut
00176   011B 108E0017           ldy   #NonsLen
00177   011F 103F8C             os9   I$WritLn
00178   0122 5F                 clrb
00179   0123 20D6               bra   Exit
00180
00181   0125 5573653A   Errmsg  fcc   'Use: send (path) Ä(/devname)Ä'


Microware OS-9 Assembler 2.1   07/01/84 11:37:01                    Page 004
send - send text file

00182   0142 0A                 fcb   LF
00183   0143 20202020           fcc   /     Default devname is 'T1' /
00184   0160 0D                 fcb   CR
00185 ¿ 003C            ErrLen  equ   *-errmsg
00186
```

forts. sid. 32

```
********************************
* FLEX DISK DRIVE EXPANDER
* AND VIRTUAL DISK DRIVE
* FROM '68 MICRO JOURNAL' DEC-82,P.27
*
* ADAPTED BY NILS-OLOV JOHANSSON
* FOR:
* CA-6819 CPU-CARD WITH DAT-RAM
* AND CBUG 2D3 MONITOR (CBUG 2.3 WITH
* DATINIT PATCH)
* TSC FLEX9 DOS
* 48k EXTENDED MEMORY AS BOARD #1
*
* WORK VARIABLES AND DRIVE TABLE CAN
* USE $DF5F-$DF6F FREE RAM
* ALL CODE FROM 'TYPTBL' TO 'ZZZZZZ'
* FITS INTO AN EPROM AT $E800
*
* TO START THIS VERSION WHEN
* BURNED INTO THE EPROM USE
* 'VDISKR' COMMAND.
*
* DO AS FOLLOWS TO CREATE THE VDISKR.CMD:
* ASSEMBLE THIS FILE TO GET VDISK.BIN
* THEN
* +++GET VDISK.BIN
* +++SAVE.LOW VDISKR.CMD,C100,C1A0,C100
*
* IF YOU DONT WANT TO USE EPROM CHANGE
* EPROM SET DIRECTIVES TO AN RAM AREA
* FOR EXAMPLE $E100 IF YOU HAVE A 2K
* RAM AT $E100-E7FF (IC 32).
********************************

            DF5F  MEMORY  SET   $DF5F
            E800  EPROM   SET   $E800
            DE00  DRVTBL  SET   $DE00

DF5F                      ORG   MEMORY
DF5F EA0D           VTABLE FDB    VECVIR
DF61 EA37           VWRITE FDB    WRITV
DF63 02             NDRVRS FCB    DRVTOT
DF64 00             GDRIVE FCB    0
DF65 00             LDRIVE FCB    0
DF66 FF             NESTCT FCB    -1
DF67 00             DRIVET FCB    0

DF68                DIOVEC
DF68 00 00                 FCB    0,0      #0 GLOBAL DRIVE
DF6A 00 01                 FCB    0,1      #1   "      "
DF6C 01 00                 FCB    1,0      #2   "      "
DF6E 00 02                 FCB    0,2      #3   "      "
                    ***************
                    *
                    * START OF EPROM CODE
                    *

E800                       ORG    EPROM

E800                TYPTBL
E800 EA92                  FDB    VECGIM
E802 03                    FCB    3
E803 44                    FCC    'D'
E804 EA0D                  FDB    VECVIR
E806 00                    FCB    0
E807 56                    FCC    ,V,

            0002  DRVTOT  EQU    (*-TYPTBL)/4

            CC12  USRCMD  EQU    $CC12
E808                CMDTBL
E808 53 45 54 00           FCC    'SET',0
E80C E80F                  FDB    SET
                    * INSERT ANY OTHER CMD'S HERE
E80E 00                    FCB    0

                    ***************
                    * "SET" CMD     *
                    ***************
            CD03  WARMS   SET    $CD03
            CD4F  CKSTAT  SET    $CD4F
            CD0D  INCH2   SET    $CD0D
            CD13  OUTCH2  SET    $CD13
            CD27  NXTCHR  SET    $CD27
            CD18  PUTCHR  SET    $CD18
            CD1E  PSTRNG  SET    $CD1E
            CD24  PCRLF   SET    $CD24
            CC02  TTYEOL  SET    $CC02
            CC0B  SYSDRV  SET    $CC0B
            CC0C  WKDRV   SET    $CC0C
            CC11  LSTERM  SET    $CC11
            CC14  LBFPNT  SET    $CC14

E80F                SET
E80F 4F                    CLRA
E810 1F 8B                 TFR    A,DP
E812 B6 CC11               LDA    LSTERM
E815 B1 CC02               CMPA   TTYEOL
E818 1027 00A7             LBEQ   FLEX
E81C 8D 7C          SET0   BSR    NEXT
E81E 81 53                 CMPA   #'S
E820 26 07                 BNE    SET1
E822 86 FF                 LDA    #$FF
E824 B7 CC0B               STA    SYSDRV
E827 20 F3                 BRA    SET0
E829 81 57          SET1   CMPA   #'W
E82B 26 07                 BNE    SET2
E82D 86 FF                 LDA    #$FF
E82F B7 CC0C               STA    WKDRV
E832 20 E8                 BRA    SET0

E834 80 30          SET2   SUBA   #'0
E836 81 03                 CMPA   #3
E838 1022 008E             LBHI   SETERR
E83C B7 DF67               STA    DRIVET
E83F 8D 59                 BSR    NEXT
E841 81 53                 CMPA   #'S
E843 27 36                 BEQ    STSYS
E845 81 57                 CMPA   #'W
E847 27 3E                 BEQ    STWK
E849 108E E803             LDY    #TYPTBL+3
E84D 5F                    CLRB

E84E A1 A4          ST1    CMPA   ,Y
E850 27 0A                 BEQ    ST7
E852 31 24                 LEAY   4,Y
E854 5C                    INCB
E855 F1 DF63               CMPB   NDRVRS
E858 25 F4                 BLO    ST1
E85A 20 6E                 BRA    SETERR
E85C 8D 3C          ST7    BSR    NEXT
E85E 80 30                 SUBA   #'0
E860 A1 3F                 CMPA   -1,Y
E862 22 66                 BHI    SETERR
E864 34 02                 PSHS   A
E866 B6 DF67               LDA    DRIVET
E869 48                    ASLA
E86A 8E DF68               LDX    #DIOVEC
E86D 30 86                 LEAX   A,X
E86F 35 02                 PULS   A
E871 1E 89                 EXG    A,B
E873 ED 84                 STD    ,X
E875 8D 23                 BSR    NEXT
E877 81 53                 CMPA   #'S
E879 26 08                 BNE    ST8

E87B                STSYS
E87B B6 DF67               LDA    DRIVET
E87E B7 CC0B               STA    SYSDRV
E881 20 8C                 BRA    SET
E883 81 57          ST8    CMPA   #'W
E885 26 08                 BNE    ST9
E887 B6 DF67        STWK   LDA    DRIVET
E88A B7 CC0C               STA    WKDRV
E88D 20 80                 BRA    SET
E88F BE CC14        ST9    LDX    LBFPNT
E892 30 1F                 LEAX   -1,X
E894 BF CC14               STX    LBFPNT
E897 16 FF75               LBRA   SET

E89A                NEXT
E89A BD CD27               JSR    NXTCHR
E89D 81 0D                 CMPA   #$D
E89F 27 70                 BEQ    PRINTS
E8A1 B1 CC02               CMPA   TTYEOL
E8A4 27 1B                 BEQ    JMPWRM
E8A6 81 2F                 CMPA   #'/
E8A8 27 17                 BEQ    JMPWRM
E8AA 81 3F                 CMPA   #'?
E8AC 27 63                 BEQ    PRINTS
E8AE 81 30                 CMPA   #'0
E8B0 2D E8                 BLT    NEXT
E8B2 81 3D                 CMPA   #'=
E8B4 27 E4                 BEQ    NEXT
E8B6 81 61                 CMPA   #'a
E8B8 25 06                 BLO    NXTR
E8BA 81 7A                 CMPA   #'z
E8BC 22 02                 BHI    NXTR
E8BE 80 20                 SUBA   #'a-'A
E8C0 39             NXTR   RTS
E8C1 32 62          JMPWRM LEAS   2,S
E8C3 8D 0D          FLEX   BSR    CHECK
E8C5 26 4C                 BNE    PRINTL
E8C7 7E CD03               JMP    WARMS
E8CA 8E E984        SETERR LDX    #MSG
E8CD BD CD1E               JSR    PSTRNG
E8D0 20 41                 BRA    PRINTL
E8D2                CHECK
E8D2 7F DF67               CLR    DRIVET
E8D5 8E DF69               LDX    #DIOVEC+1
E8D8 C6 04                 LDB    #4
E8DA A6 84          CK1    LDA    ,X
E8DC 84 7F                 ANDA   #$7F
E8DE A7 84                 STA    ,X
E8E0 30 02                 LEAX   2,X
E8E2 5A                    DECB
E8E3 26 F5                 BNE    CK1
E8E5 8E DF68               LDX    #DIOVEC
E8E8                CK2
E8E8 EC 81                 LDD    ,X++
E8EA 1F 12                 TFR    X,Y
E8EC 8C DF70               CMPX   #DIOVEC+8
E8EF 27 1C                 BEQ    CK8
E8F1                CK4
E8F1 10A3 A4               CMPD   ,Y
E8F4 26 0D                 BNE    CK6
E8F6 A6 21                 LDA    1,Y
E8F8 8A 80                 ORA    #$80
E8FA A7 21                 STA    1,Y
E8FC CA 80                 ORB    #$80
E8FE E7 1F                 STB    -1,X
E900 7C DF67               INC    DRIVET
E903                CK6
E903 31 22                 LEAY   2,Y
E905 108C DF70             CMPY   #DIOVEC+8
E909 26 E6                 BNE    CK4
E90B 20 DB                 BRA    CK2
E90D                CK8
E90D 7D DF67               TST    DRIVET
E910 39                    RTS

E911                PRINTS
E911 32 62                 LEAS   2,S
E913 8D BD          PRINTL BSR    CHECK
E915 8E DF68               LDX    #DIOVEC
E918 BD CD24               JSR    PCRLF
E91B 5F                    CLRB
E91C 1F 98          PLOOP  TFR    B,A
E91E 8B 30                 ADDA   #'0
E920 BD CD18               JSR    PUTCHR
E923 86 3D                 LDA    #'=
E925 BD CD18               JSR    PUTCHR
E928 A6 80                 LDA    ,X+
E92A 108E E803             LDY    #TYPTBL+3
E92E 48                    ASLA
E92F 48                    ASLA
E930 A6 A6                 LDA    A,Y
E932 BD CD18               JSR    PUTCHR
E935 A6 84                 LDA    ,X
E937 84 7F                 ANDA   #$7F
E939 8B 30                 ADDA   #'0
E93B BD CD18               JSR    PUTCHR
E93E 86 20                 LDA    #$20
E940 BD CD18               JSR    PUTCHR
E943 7D CC0B               TST    SYSDRV
E946 2B 09                 BMI    WRITS
E948 F1 CC0B               CMPB   SYSDRV
E94B 27 04                 BEQ    WRITS
E94D 86 20                 LDA    #$20
E94F 20 02                 BRA    PL1
E951 86 53          WRITS  LDA    #'S
E953 BD CD18        PL1    JSR    PUTCHR
E956 7D CC0C               TST    WKDRV
E959 2B 09                 BMI    WRITW
E95B F1 CC0C               CMPB   WKDRV
E95E 27 04                 BEQ    WRITW
E960 86 20                 LDA    #$20
E962 20 02                 BRA    PL2
E964 86 57          WRITW  LDA    #'W
E966                PL2
E966 BD CD18               JSR    PUTCHR
E969 A6 80                 LDA    ,X+
E96B 85 80                 BITA   #$80
E96D 27 0A                 BEQ    PL3
E96F 34 16                 PSHS   D,X
E971 8E E984               LDX    #MSG
E974 BD CD1E               JSR    PSTRNG
E977 35 16                 PULS   D,X
E979                PL3
E979 BD CD24               JSR    PCRLF
E97C 5C                    INCB
E97D C1 04                 CMPB   #4
E97F 25 9B                 BLO    PLOOP
E981 7E CD03               JMP    WARMS
E984                MSG
E984 20 2A 2A 20           FCC    ' ** ERR **',8,4

                    * DISKIO SOURCE
E990                DRIVE
E990 34 C6                 PSHS   D,U,PC
E992 7C DF66               INC    NESTCT
E995 26 05                 BNE    DIO1
E997 A6 03                 LDA    3,X
E999 B7 DF64               STA    GDRIVE
E99C EC 66          DIO1   LDD    6,S
E99E ED 62                 STD    2,S
E9A0 AF 66                 STX    6,S
E9A2 CC E9E8               LDD    #DRVRTN
E9A5 ED 64                 STD    4,S
E9A7 35 06                 PULS   D
E9A9                DISKIO
E9A9 34 26                 PSHS   D,Y
E9AB B6 DF64               LDA    GDRIVE
E9AE 81 03                 CMPA   #3
E9B0 22 46                 BHI    ERROR
E9B2 48                    ASLA
E9B3 108E DF68             LDY    #DIOVEC
E9B7 EC A6                 LDD    A,Y
E9B9 B1 DF63               CMPA   NDRVRS
E9BC 22 3A                 BHI    ERROR
E9BE 5D                    TSTB
E9BF 2B 37                 BMI    ERROR
E9C1 F7 DF65               STB    LDRIVE
E9C4 7D DF66               TST    NESTCT
E9C7 26 02                 BNE    DIO2
E9C9 E7 03                 STB    3,X
E9CB 48             DIO2   ASLA
E9CC 48                    ASLA
E9CD 108E E800             LDY    #TYPTBL
E9D1 31 A6                 LEAY   A,Y
E9D3 B6 DF65               LDA    LDRIVE
E9D6 A1 22                 CMPA   2,Y
E9D8 22 1E                 BHI    ERROR
E9DA 10AE A4               LDY    ,Y
E9DD E6 65                 LDB    5,S
E9DF C0 03                 SUBB   #3
E9E1 31 A5                 LEAY   B,Y
E9E3 10AF 64               STY    4,S
E9E6 35 A6                 PULS   D,Y,PC
E9E8                DRVRTN
E9E8 35 10                 PULS   X
E9EA 34 01                 PSHS   CC
E9EC 7A DF66               DEC    NESTCT
E9EF 2A 05                 BPL    DIO4
E9F1 B6 DF64               LDA    GDRIVE
E9F4 A7 03                 STA    3,X
E9F6 35 81          DIO4   PULS   CC,PC
E9F8 35 26          ERROR  PULS   D,Y
E9FA 32 62                 LEAS   2,S
E9FC C6 0F                 LDB    #$0F
E9FE 1A 01                 SEC
EA00 39                    RTS
```

```
                        * VIRTUAL DISK DRIVERS
                        * 48K MEMORY AS EXT.PAGE 1 (CARD #1)
                        ************************************

EA01 1F 1E 1D 1C    VDAT    FCB     $1F,$1E,$1D,$1C,$1B,$1A,$19,$18
EA09 17 16 15 14            FCB     $17,$16,$15,$14
             000B   MAXTRK  EQU     *-VDAT-1
             DF70   LRARAM  EQU     $DF70
             FFF0   DATRAM  EQU     $FFF0

EA0D 16   001B      VECVIR  LBRA    READV
EA10 16   0024              LBRA    WRITV
EA13 16   007A              LBRA    VERIFV
EA16 16   0077              LBRA    RESTOV
EA19 16   0074              LBRA    DRIVEV
EA1C 16   0071              LBRA    CKRDYV
EA1F 16   006E              LBRA    QUICKV
EA22 16   006B              LBRA    DINITV
EA25 16   0065              LBRA    DWARMV
EA28 16   0065              LBRA    DSEEKV
EA2B                READV
EA2B 34   01                PSHS    CC
EA2D 1A   10                SEI
EA2F 8D   2F                BSR     SETDAT
EA31 1E   12                EXG     X,Y
EA33 8D   1D                BSR     COPYXY
EA35 20   08                BRA     RESDAT
EA37                WRITV
EA37 34   01                PSHS    CC
EA39 1A   10                SEI
EA3B 8D   23                BSR     SETDAT
EA3D 8D   13                BSR     COPYXY
EA3F B6   DF70      RESDAT  LDA     LRARAM
EA42 B7   FFF0              STA     DATRAM
EA45 B6   DF72              LDA     LRARAM+2
EA48 B7   FFF2              STA     DATRAM+2
EA4B 35   01                PULS    CC
EA4D C6   00                LDB     #0
EA4F 1C   FE                CLC
EA51 39                     RTS
EA52 C6   80        COPYXY  LDB     #128
EA54 34   04                PSHS    B
EA56 EC   81        LOOP    LDD     ,X++
EA58 ED   A1                STD     ,Y++
EA5A 6A   E4                DEC     ,S
EA5C 26   F8                BNE     LOOP
EA5E 35   84                PULS    B,PC

EA60 5A             SETDAT  DECB
EA61 C1   0F                CMPB    #15
EA63 22   1F                BHI     SEEKER
EA65 81   0B                CMPA    #MAXTRK
EA67 22   1B                BHI     SEEKER
EA69 108E EA01              LDY     #VDAT
EA6D A6   A6                LDA     A,Y
EA6F 8C   1000              CMPX    #$1000
EA72 25   05                BLO     SD1
EA74 B7   FFF0              STA     DATRAM
EA77 20   05                BRA     SD2
EA79 B7   FFF2      SD1     STA     DATRAM+2
EA7C CB   20                ADDB    #$20
EA7E 1F   98        SD2     TFR     B,A
EA80 5F                     CLRB
EA81 1F   02                TFR     D,Y
EA83 39                     RTS
EA84 32   62        SEEKER  LEAS    2,S
EA86 35   01                PULS    CC
EA88 C6   10                LDB     #$10
EA8A 1A   01                SEC
EA8C 39                     RTS
EA8D 5F             DWARMV  CLRB
EA8E 1F   9B                TFR     B,DP
EA90                VERIFV
EA90                RESTOV
EA90                DRIVEV
EA90                CKRDYV
EA90                QUICKV
EA90                DINITV
EA90                DSEEKV
EA90 5F                     CLRB
EA91 39                     RTS
                    ******
                    * THIS CODE IS NOT USED BY FLEX9
                    *GIMDSK SET $DE59
                    *GIMDSK SET $DE56
                    *GVECTR
                    *PSHS D
                    *LDD 2,S
                    *ADDD #$DE00-VECGIM
                    *STD 2,S
                    *PULS D
                    *JMP GIMDSK
                    ***********

EA92                VECGIM
EA92 7E   DE31              JMP     FREAD       FLEX9 VECTORS
EA95 7E   DEA2              JMP     FWRITE
EA98 7E   DED9              JMP     FVERFY
EA9B 7E   FEF3              JMP     FRESTO
EA9E 7E   DF19              JMP     FDRIVE
EAA1 7E   DF41              JMP     FCHECK
EAA4 7E   DF41              JMP     FQUICK
                    *
EAA7 7E   DE26      GCOLD   JMP     FINIT
EAAA 7E   DE26      GWARM   JMP     FINIT
                    *JSR GVECTR NOT USED
EAAD 8D   F8        COLD    BSR     GCOLD
EAAF BD   EA22              JSR     VECVIR+$15
                    *
EAB2 39                     RTS
EAB3 8D   F5        WARMD   BSR     GWARM
EAB5 BD   EA25              JSR     VECVIR+$18
                    *
EAB8 39                     RTS
          EAB9      ZZZZZZ  EQU     *
                    * END OF EPROM CODE
```

```
                    ****************************
                    * INSTALL VIRTUAL DISK DRIVER
                    * +++VDISKR ( OPT )
                    * OPTIONS S , B
                    * S=INSTALL ONLY SET.CMD
                    * B=RESTORE FLEX DRIVER
                    ****************************
                    *  FLEX9 DISK DRIVER VECTOR TABLE
                    * CHANGE TO MATCH YOUR VERSION OF FLEX
                    *
          DE31      FREAD   EQU     $DE31
          DEA2      FWRITE  EQU     $DEA2
          DED9      FVERFY  EQU     $DED9
          FEF3      FRESTO  EQU     $FEF3
          DF19      FDRIVE  EQU     $DF19
          DF41      FCHECK  EQU     $DF41
          DF41      FQUICK  EQU     $DF41
          DE26      FINIT   EQU     $DE26

C100                        ORG     $C100
C100                START
C100 BD   CD27              JSR     NXTCHR      CHECK IF OPTION
C103 24   09                BCC     BACK
C105 81   0D                CMPA    #$D         CR NO OPTION
C107 27   2E                BEQ     VIRT        INSTALL VDISK
C109 B1   CC02              CMPA    TTYEOL
C10C 27   29                BEQ     VIRT
C10E 81   42        BACK    CMPA    #'B
C110 27   10                BEQ     BCOPY       REMOVE VDISK
C112 81   53                CMPA    #'S
C114 26   06                BNE     EXIT

                    *'S' INSTALL SET.CMD
C116 8E   E808              LDX     #CMDTBL
C119 BF   CC12      XUSR    STX     USRCMD
C11C BD   CD27      EXIT    JSR     NXTCHR
C11F 7E   CD03              JMP     WARMS

                    *'B' RESTORE FLEX DRIVER
C122 8E   C178      BCOPY   LDX     #FLXDR
C125 108E DE00              LDY     #DRVTBL
C129 C6   18                LDB     #(FXEND-FLXDR)
C12B A6   80        SU3     LDA     ,X+
C12D A7   A0                STA     ,Y+
C12F 5A                     DECB
C130 26   F9                BNE     SU3
C132 8E   0000              LDX     #$0000      NO CMD TABLE OR DEFINE
                    * YOUR OWN CMD TABLE
C135 20   E2                BRA     XUSR        AND EXIT

                    * START VIRTUAL DISK AND SET.CMD
C137 8E   E808      VIRT    LDX     #CMDTBL
C13A BF   CC12              STX     USRCMD

                    * COPY WORK VARIABLES AND DIOVEC
                    * TO MEMORY

C13D 8E   C190              LDX     #RAMVAR
C140 108E DF5F              LDY     #MEMORY
C144 C6   11                LDB     #(VAREND-RAMVAR)
C146 A6   80        CPYVAR  LDA     ,X+
C148 A7   A0                STA     ,Y+
C14A 5A                     DECB
C14B 26   F9                BNE     CPYVAR

C14D 8E   C160      CPYNEW  LDX     #JMPTBL
C150 108E DE00              LDY     #DRVTBL
C154 C6   18                LDB     #(ENDTBL-JMPTBL)
C156 A6   80        SU6     LDA     ,X+
C158 A7   A0                STA     ,Y+
C15A 5A                     DECB
C15B 26   F9                BNE     SU6
C15D 7E   E80F              JMP     SET

C160                JMPTBL
C160 BD   E9A9              JSR     DISKIO
C163 BD   E9A9              JSR     DISKIO
C166 BD   E9A9              JSR     DISKIO
C169 BD   E990              JSR     DRIVE
C16C BD   E990              JSR     DRIVE
C16F BD   E990              JSR     DRIVE
C172 BD   E990              JSR     DRIVE
C175 7E   EAAD              JMP     COLD
                    *JMP WARMD NOT USED BY FLEX
          C178      ENDTBL  EQU     *

C178                FLXDR
C178 7E   DE31              JMP     FREAD
C17B 7E   DEA2              JMP     FWRITE
C17E 7E   DED9              JMP     FVERFY
C181 7E   FEF3              JMP     FRESTO
C184 7E   DF19              JMP     FDRIVE
C187 7E   DF41              JMP     FCHECK
C18A 7E   DF41              JMP     FQUICK
C18D 7E   DE26              JMP     FINIT
          C190      FXEND   EQU     *
                    *
C190                RAMVAR
C190 EA0D                   FDB     VECVIR
C192 EA37                   FDB     WRITV
C194 02                     FCB     DRVTOT
C195 00 00 FF 00            FCB     0,0,-1,0

                    * SAME AS DIOVEC
C199 00 00                  FCB     0,0
C19B 00 01                  FCB     0,1
C19D 01 00                  FCB     1,0
C19F 00 02                  FCB     0,2
C1A1                VAREND
                            END     START

0 ERROR(S) DETECTED
```

```
                    NAM     VFORMAT2
                   *OPT PAG
                    PAG
                   ********************************
                   *  VIRTUAL DISK FORMATTER
                   * FROM '68 MICRO JOURNAL DEC-82,P.31
                   *  MAXTRK = $0B  FOR 48k MEMORY
                   *
                   * MODIFIED VERSION
                   * DIR INCLUDE SEC 1,2 ON TRK 0
                   ********************************

                   * NOTE STRT TRK & SEC MUST BE >= 00,06

              0000 STRTRK  EQU     0
              0006 STRTSC  EQU     6          * 1 SECT. FOR DIR (10 FILES)
                   * ENTIRE DIR = 30 FILES
C100                ORG     $C100      ,CMD

                   * MAXTRK MUST MATCH SAME IN VIRT DRIVERS
              000B MAXTRK  SET     $B         * 48K. VIRTUAL
              DF61 VWRITE  SET     $DF61      IND.ADR FOR VIRTUAL WRITE

              CC0E DATE    SET     $CC0E
              CD1E PSTRNG  SET     $CD1E
              CD18 PUTCHR  SET     $CD18
              CD39 OUTDEC  SET     $CD39
              CD42 GETHEX  SET     $CD42
              CD15 GETCHR  SET     $CD15
              CD24 PCRLF   SET     $CD24
              CD1B INBUF   SET     $CD1B
              CD2D GETFIL  SET     $CD2D
              CD48 INDEC   SET     $CD48
              D406 FMS     SET     $D406
              D403 FMSCLS  SET     $D403
              CD3C OUT2HS  SET     $CD3C
              CD03 WARMS   SET     $CD03

C100 20 18         FMTVIR  BRA     FMT1
C102 02                    FCB     2          EXTENDED DIR VERSION
C103 BD CD1E       PROMPT  JSR     PSTRNG
C106 BD CD15               JSR     GETCHR
C109 84 5F                 ANDA    £$5F
C10B 81 59                 CMPA    £'Y
C10D 39                    RTS
C10E 8E C1E2       EXIT    LDX     £ABORT
C111 BD CD1E               JSR     PSTRNG
C114 BD D403               JSR     FMSCLS
C117 7E CD03               JMP     WARMS

C11A 8E C1AE       FMT1    LDX     £SURES
C11D 8D E4                 BSR     PROMPT
C11F 26 ED                 BNE     EXIT
C121 8E C239               LDX     £WORK
C124 5F                    CLRB
C125 6F 80         FMT2    CLR     ,X+
C127 5A                    DECB
C128 26 FB                 BNE     FMT2
C12A CC 0002               LDD     £2
C12D 8E C239               LDX     £WORK
C130 ED 84                 STD     0,X
C132 CC 0001               LDD     £1
C135 AD 9F DF61    FMTGO   JSR     [VWRITE]   INDIRECT
C139 25 6E                 BCS     WOUPS
C13B 8E C239               LDX     £WORK
C13E EC 84                 LDD     ,X
C140 34 06                 PSHS    D
C142 C1 10                 CMPB    £MAXSEC
C144 25 02                 BLO     NXTS1
C146 4C                    INCA
C147 5F                    CLRB
C148 5C            NXTS1   INCB
C149 ED 84                 STD     ,X
C14B 35 06                 PULS    D
C14D 10B3 C230             CMPD    LSTSEC
C151 26 E2                 BNE     FMTGO
C153 6F 84                 CLR     ,X
C155 6F 01                 CLR     1,X
C157 FC C230               LDD     LSTSEC
C15A AD 9F DF61            JSR     [VWRITE]
C15E 25 49                 BCS     WOUPS

                   * EXTRA DIR 00,01 00,02
C160 8E C239               LDX     £WORK
C163 CC 0002               LDD     £2
C166 AD 9F DF61            JSR     [VWRITE]   CLEAR FWD LINK SEC 2
C16A 25 3D                 BCS     WOUPS
C16C 8E C239               LDX     £WORK
C16F CC 0001               LDD     £1         LINK TO 00,01
C172 ED 84                 STD     ,X

C174 FC C22E               LDD     FSTSEC
C177 C0 02                 SUBB    £2
C179 2A 03                 BPL     DR1
C17B C6 0F                 LDB     £MAXSEC-1
C17D 4A                    DECA
C17E 5C            DR1     INCB
C17F 8E C239               LDX     £WORK
C182 AD 9F DF61            JSR     [VWRITE]
C186 25 21                 BCS     WOUPS

C188 8E C211               LDX     £SIRBLK
C18B FC CC0E               LDD     DATE
C18E FD C234               STD     SDATE
C191 B6 CC10               LDA     DATE+2
C194 B7 C236               STA     SDATE+2
C197 CC 0003               LDD     £3
C19A AD 9F DF61            JSR     [VWRITE]
C19E 25 09                 BCS     WOUPS
C1A0 8E C1FD               LDX     £COMPLT
C1A3 BD CD1E       DONE    JSR     PSTRNG
C1A6 7E CD03               JMP     WARMS
C1A9 8E C1E2       WOUPS   LDX     £ABORT
C1AC 20 F5                 BRA     DONE

C1AE 41 52 45 20   SURES   FCC     'ARE YOU SURE YOU WANT TO FORMAT
C1CD 20 54 48 45           FCC     ' THE VIRTUAL DISK ? ',4
C1E2 2A 2A 2A 20   ABORT   FCC     '*** FORMATTING ABORTED ***',4
C1FD 46 4F 52 4D   COMPLT  FCC     'FORMATTING COMPLETE',4

              0010 MAXSEC  EQU     16

                   * SYSTEM INFO BLOCK

              C211 SIRBLK  EQU     *
C211 00 00         FWDLNK  FCB     0,0
C213 0000 0000             FDB     0,0,0,0,0,0
C221 56 49 52 54   VOLNAM  FCC     'VIRTUAL-MEM'
C22C 0000          VOLNUM  FDB     0
C22E 00 06         FSTSEC  FCB     STRTRK,STRTSC
C230 0B 10         LSTSEC  FCB     MAXTRK,MAXSEC
C232 00BB          TOTSEC  FDB     (MAXTRK-STRTRK+1)*MAXSEC-STRTSC+1
C234 00 00 00      SDATE   FCB     0,0,0
C237 0B            HITRK   FCB     MAXTRK
C238 10            HISEC   FCB     MAXSEC
C239               WORK    RMB     256

              C339 ZZZFMT  EQU     *

                   END     FMTVIR

0 ERROR(S) DETECTED
```

forts. fr. sid. 29

```
00187  0161 308DFFC0  Err       leax  Errmsg,pcr point to the text
00188  0165 108E003C            ldy   #Errlen
00189  0169 8601                lda   #StdOut
00190  016B 103F8C              os9   I$WritLn   write the error text
00191  016E 5F                  clrb             no error code
00192  016F 208A                bra   Exit
00193
00194  0171 48454C4C  Hello     fcc   /HELLO/
00195  0176 0D                  fcb   CR
00196  0006           HelloLen  equ   *-Hello
00197  0177 53454E44  Sendfil   fcc   /SENDFIL/
00198  017E 8D                  fcb   CR+$80
00199  0008           SendLen   equ   *-Sendfil
00200  017F 53594E4B  Synksv    fcc   /SYNKSVAR/
00201  0187 0D                  fcb   CR
00202  0009           SynkLen   equ   *-Synksv
00203  0188 2F2A      Endof     fcc   '/*'
00204  018A 0D                  fcb   CR
00205  0003           EndofLen  equ   *-Endof
00206
00207  018B 00893A              emod             module CRC
00208
00209  018E           Size      equ   *          end of program
00210
00211                            end


00000 error(s)
00000 warning(s)
$018E 00398 program bytes generated
$0523 01315 data bytes allocated
$11F5 04597 bytes used for symbols
```

64 k RAM

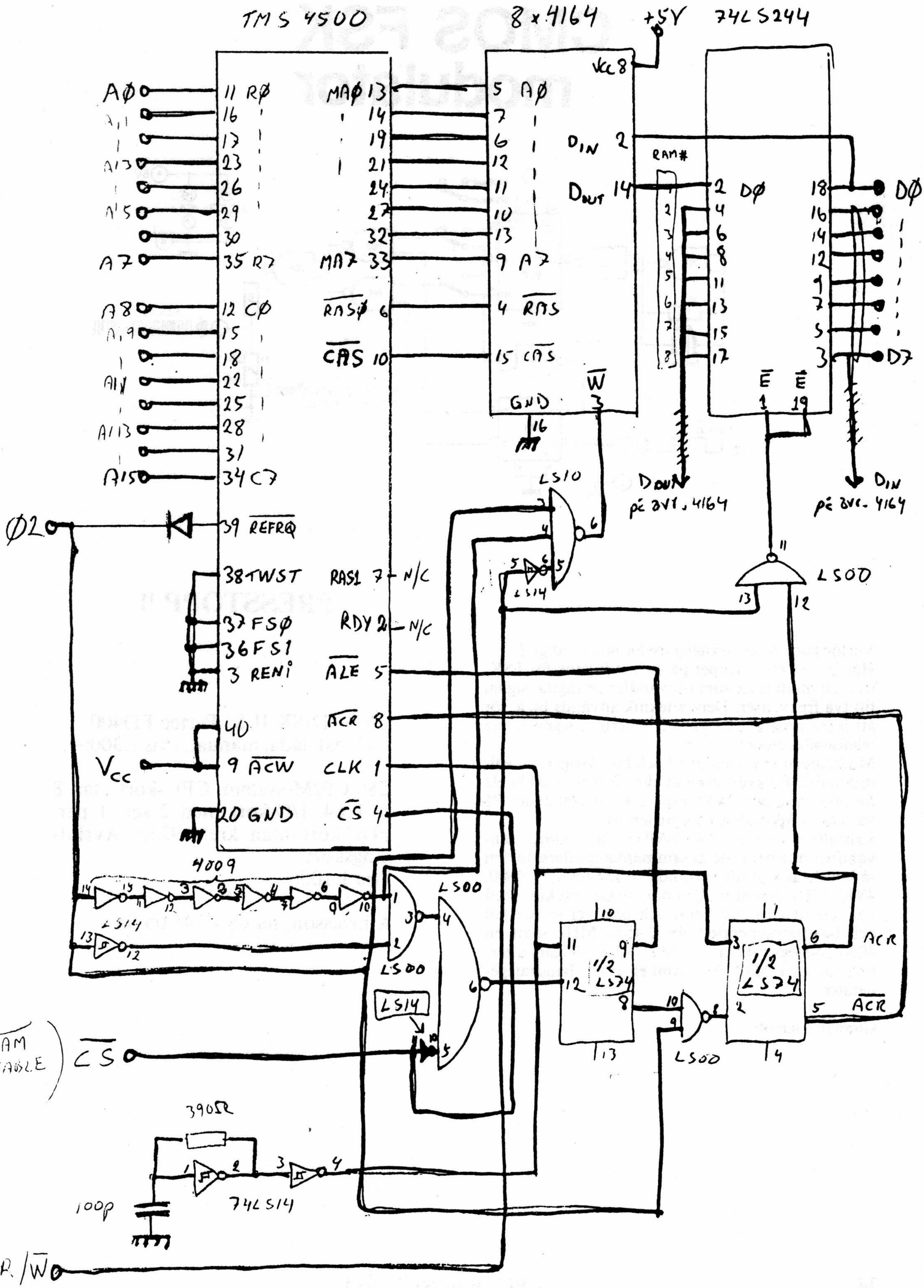
Visst kan man på ett enkelt sätt använda dynamiska 64k-kapslar till Motorola-kretsar. Med hjälp av Texas 4500 kontollerchip så räcker det alltså med 8 minneskapslar för att få 64k minne (eller mer, allt efter behag).
Jag har själv detta system i drift sedan c:a 1 år tillbaka och kan intyga att det fungerar utmärkt.

Idén kommer från Sune Johansson i Hässleholm som är en verklig snitsare på specialkonstruktioner.

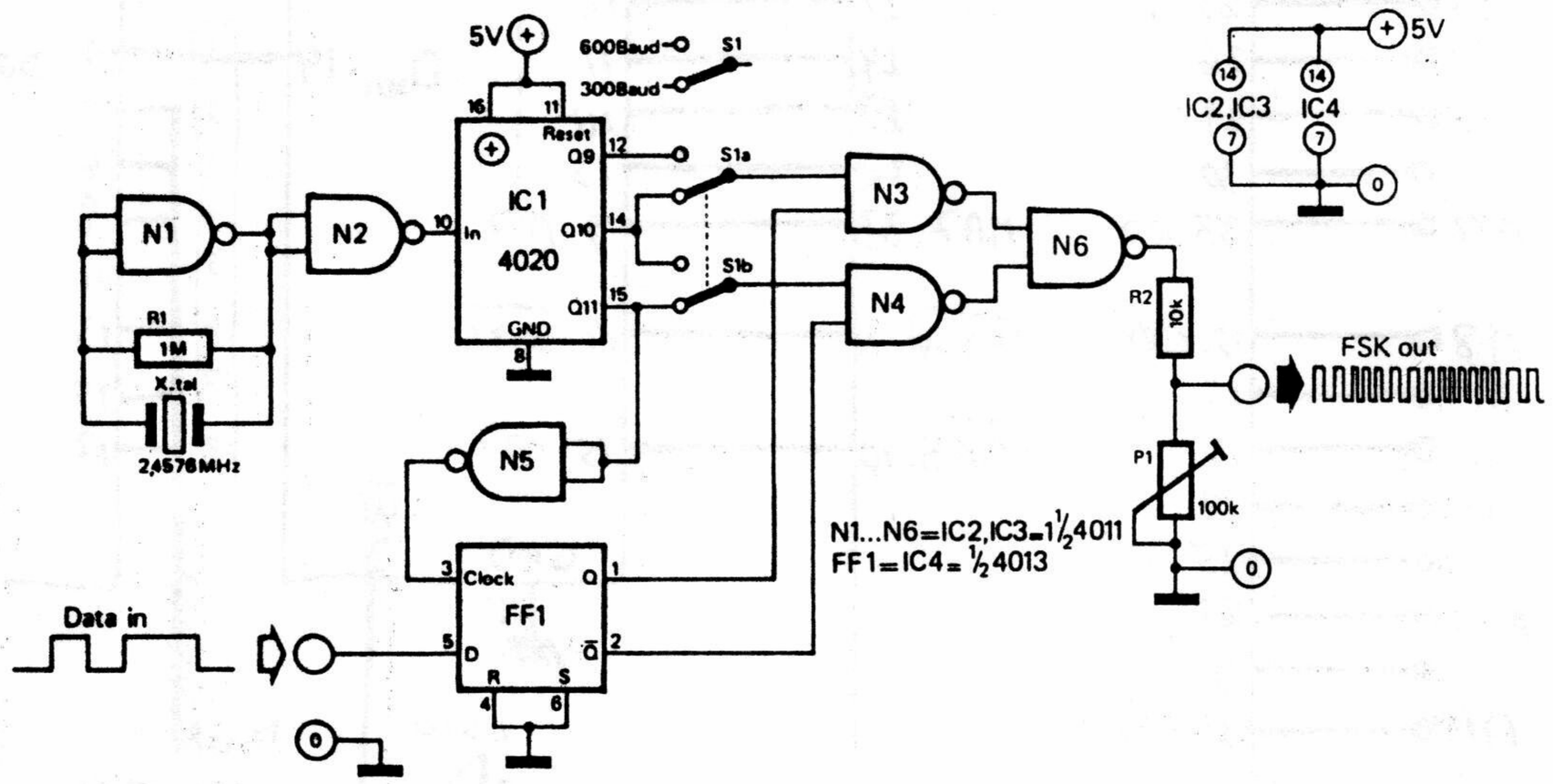
Som synes blir det inte speciellt många kretsar för ett komplett minneskort.
Tyvärr kunde inte kvaliteten på ritningen bli den bästa p.g.a tidsbrist men det får Ni ha överseende med.

Gunnar Ejemoh

Dyn. RAM-controller 64k



CMOS FSK modulator



Varför göra saker krångligare än nödvändigt ?
Här har vi ett exempel på en modulator för FSK, dvs. en modulator som omvandlar en digital signal till två frekvenser. Denna teknik används bl. a. för att lagra program på kassett, överföra signaler via telefonnät (modem) m.m.
Modulatorn är kristallstyrd och blir därigenom väldigt stabil. Frekvenserna ut blir 1200 resp. 2400 Hz för 300 baud och 2400 resp 4800 för 600 baud. P1 varierar amplituden på signalen ut.
Kristallfrekvensen 2,4576MHz är väldigt användbar inte bara i detta sammanhang eftersom den så lätt delas ned till välkända baudrates som 9600, 4800, 1200, 300 m.m. Om man bygger ett kort med t.ex. en ACIA (eller flera) och behöver en kristall och baudrategenerator blir 2,4576 MHz samt en 4060 14-stegs räknare ett MYCKET billigare alternativ än en 1,8432 MHz samt en 14411 baudrategenerator.

Gunnar Ejemoh

PRESSTOPP !!

EXORDISK II 8" Pertec FD400
inkl. nät, låda, manual. Pris 1.300:– .

Z80 CP/M-system. CPU-kort utan 8 st 4164. I/O-kort med 2 ser, 1 par. FDC-kort utan krets 9216. Avslutningskort.
Pris 1.100:– .

A Eriksson, tel 08 - 745 05 98