Årgång 1

VIC NEWS

Nr 1

- **FÄRG**
- **LJUD**
- **GRAFIK**

VIC 20 – *den första folkdatorn*

INNEHÅLLSFÖRTECKNING

ANNONSPRISLISTA VIC-NEWS

Radannonser gällande VIC från medlemmar i VIC-klubben införes kostnadsfritt.

Kommersiell annonsering debiteras enligt nedan:

1/1 sida	2.000 kr
1/2 sida	1.200 kr
1/4 sida	650 kr

För annonsering över en längre tidsperiod kan frekvensrabatter lämnas. Kontakta VIC-klubben.

VIC-NEWS Upplaga 20.000 ex.

VIC-KLUBBEN - en klubb för alla datorintresserade.

Var med från början i det som snabbt blir Skandinaviens största klubb för datorintresserade människor. Först nu har datorteknologin utvecklats så långt att datorn kan bli var mans egendom. Och nu när datorn finns, bildas också naturligtvis klubbar för alla som är intresserade. Datorn som gör detta möjligt är VIC och klubbarna blir följdaktligen VIC-klubbar.

Som medlem i en VIC-klubb har du många förmåner.

Eftersom varje lokal VIC-klubb tillhör moderklubben för Sverige, har du möjligheter att knyta kontakter både på din egen ort och över hela Sverige.

- Du kan byta program med andra VIC-klubbsmedlemmar. I VIC-klubben finns en programbytarservice du kan utnyttja. Programbytarservicen beskrivs utförligt i nästa nummer av VIC-news. Redan där hittar du ett antal program som kan vara roliga att byta till sig. Här ska bara sägas att det är gratis att byta program. Du betalar bara returporto plus expeditionskostnader.

- Du får VIC-NEWS hem i brevlådan 6 gånger per år. VIC-NEWS tar upp nyheter kring VIC, både nationella och internationella. VIC-NEWS tar också upp nya program, nya användningsområden, ger programmeringstips etc, etc. VIC-NEWS ger dig också möjlighet att själv komma i kontakt med andra VIC-klubbsmedlemmar. Du skriver ner några rader om dina erfarenheter och det tas in i VIC-NEWS.

- Om du privat vill välja eller köpa någonting, sätter du in kostnadsfria radannonser i VIC-NEWS.

- VIC-tidningens redaktion bevakar fortlöpande vad som händer internationellt. Redaktionen har direktkontakter med motsvarande redaktioner i andra länder, exempelvis USA, England, Tyskland, Holland, Japan etc. Dessutom bevakas generell internationell datorpress.

Var finns närmaste VIC-klubb?

Idag finns lokala VIC-klubbar på ett fåtal ställen i landet. I kommande nummer av VIC-NEWS kommer vi att fortlöpande meddela adresser till nystartade

VIC-klubbar. Du som vill starta eller redan har startat en VIC-klubb - hör snarast av dig till moderklubben.

Så här blir du snabbast medlem i VIC-klubben!

Skicka in medlemsavgiften 100 kr på postgirokonto 84646-9 och ange att det gäller medlemsskap i VIC-klubben. Så fort dina pengar kommer in registreras du automatiskt som medlem.

Kontakt med redaktionen

Vill du skicka in redaktionellt material till VIC-NEWS, radannonser eller vanliga annonser är postadressen till VIC-klubben:

VIC-klubben
Box 1509
436 00 Askim

MINNESORGANISATION

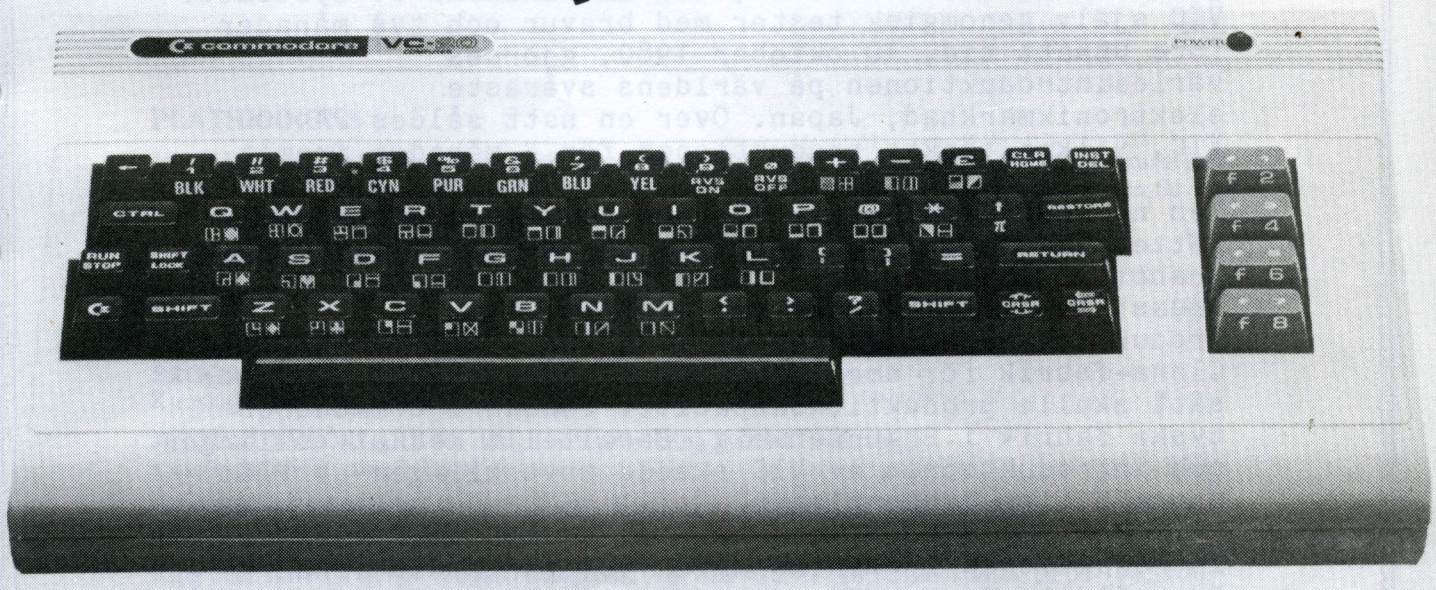
VIC's microprosessor 6502 kan adressera upp till 32000 olika användarminnes celler (med minnesexpansion).Du kan se VIC,s minne som en bok med 125 "sidor" med 256 minnesplatser på varje sida.Exempelvis sidan ¤80 (hexadecimal 80) består av 256 minnesplatser som börjar med adressen ¤8000 (hex 8000) eller decimalt uttryckt32768. För dig som inte är riktigt säker på hur man räknar om mellan hexadecimal och decimalt kommer följande exempel.
Om du önskar omvandla det hexadecimala talet ¤7342 till decimaltal räknar man enligt följande:

7 * 4096 + 3 * 256 + 4 * 16 + 2*1 = 29506

VIC-20 FÄRG DATOR

PROGRAMMERINGS-HANDBOK

149 kr



commodore COMPUTER

VIC ÄNTLIGEN HOS DIG!

Första gången VIC visades för en större grupp var för exakt 2 år sedan (5 jan 1980). Visningen skedde i en privat svit under CES-mässan i Las Vegas.

VIC tog oss alla med storm. Vi som var församlade tyckte att det omöjliga hade uppnåtts. En dator som även den enskilda människan hade råd att köpa!

Men varför dröjde det så länge innan vi kunde köpa Vic här hemma i Sverige? Den VIC vi såg för två år sedan var en prototyp, och vi var väl medvetna om att det skulle ta bortåt ett år innan någon storskalig produktion kunde börja.

Det som skulle göras under detta år var:

1. Testa ett nyutvecklat chip som utvecklats speciellt för VIC, och som givit namnet åt VIC, Video Interface Chip.

2. Testa Vic hos ett störe antal användare.

3. Största massproduktion med allt vad det innebär.

VIC-chipet vars egentliga beteckning är 6560 gick igenom testproceduren utan några större komplikationer. Detta gällde för NTSC-versionen, dvs amerikanska TV-systemet. VIC själv genomgick tester med bravur och två månader före utsatt tid, 10 oktober 1980, gjordes världsintroduktionen på världens svåraste elektronikmarknad, Japan. Över en natt såldes 24.000 VIC, vilket täckte produktionen för 4 månader framåt.

Men nu kom "problemen". Även om alla hade väntat sig en jättesucce hade ingen förstått att det skulle gå så snabbt. Besked för de 4 kommande månaderna togs att VIC endast skulle marknadsföras i Japan, samt att produktionskapacitet skulle lösgöras i Commodores Santa Clara-fabrik för amerikanska marknaden. På motsvarande sätt skulle produktionskapacitet finnas i Commodores tyska fabrik i Braunschweig. Parallellt med utvecklingen och introduktionen av VIC skedde utvecklingen av VICs stora tillbehörssortiment.

Introduktionen i Amerika mars 1981 var minst lika överväldigande som den japanska introduktionen.

Den europeiska marknadsintroduktionen stötte däremot på problem. Efterfrågan fanns där, men fälttest av VIC-chipets europeiska variant, anpassat till PAL-systemet hade "glömts bort". Beslutet togs att inga europeiska produkter skulle säljas, förrän testet genomförts. Konsumenten skulle inte bli lidande av en eventuell felaktighet. I juni var testerna genomförda och maskineriet började snurra igång.

Succen var given på förhand, över 125.000 VIC var förköpta i Europa. Alla visste vi att det skulle bli slagsmål om de första som kom ut i Europa. Tyskland och England introducerade VIC på marknaden i oktober, Sverige, Holland, Italien och Spanien i November alla dock med förhållandevis små kvantiteter, och i Sverige hann bara 1000 VIC säljas innan årsskiftet.

Vi alla beklagar naturligtvis att inte alla som ville ha en VIC kunde få den, men nu finns den här, om inte kanske för omgående leverans, så dock med en mindre leveranstid. Glädjande är också att alla tillbehör finns att tillgå.

Den avslutande reflektionen är att den verkliga datarevolutionen nu har börjat.

VIC ÄR ÄNTLIGEN HOS DIG.

PLATSANNONS

Vi vill ha några till i gänget. Det vi erbjuder är en kreativ miljö i Skandinaviens framgångsrikaste elektronikföretag. Vi jobbar med produkter och teknologi som är så färsk att alla som rör vid den får vara med att forma den. Vi har två grundläggande krav på dig.

1. Intresse för mikrodatorer och teknik. Helst skall du ha det som hobby.

2. Viljan att vilja utveckla dig.

Följande platser skall tillsättas.

PROGRAMMERARE för utveckling av administrativ programvara.
Kunskapskrav: programmeringskunnig.

JUNIORUTVECKLARE för design och utveckling av hård/mjukvara.
Kunskapskrav: kunnig i digital elektronik.

SENIORUTVECKLARE för självständig design och utveckling av hård/mjukvara.
Kunskapskrav: kunnig i digital teknik.

Ring eller skriv till Bengt Wirgart, Datatronic AB, Box 42054, 126 12 Stockholm, tel 08-744 59 20.

datatronic ab
Generalagent: Datatronic AB, Vretensborgsvägen 8,
Box 42094, S-126 12 Stockholm, Tel: 08/744 59 20, Telex: PET S 17828.

Många önskemål om program

Otaliga är de samtal vi får där folk frågar om program för användningsområde si eller så existerar. Tyvärr måste vi i de allra flesta fallen svara nej - programmet existerar inte.

För att närma sig en lösning på problemet vidarebefordrar vi här tips för program. Välj något programämne i listan och sätt igång. När programmen är klara bör du meddela detta i nästa nummer av VIC-news. Då får du alla VIC-användare som målgrupp.

Programtips!

1. Privatkassabok och budget

2. Aktieaffärer

3. Register för grammofonskivor

4. Register för matrecept

5. Travprogram Kan hålla reda på hästar, kusk, resultat, bana etc och ge förslag på vinnare.

6. Fotbollsprogram. Ska ha plats för engelska ligan eller allsvenskan. Ska kunna ge tipsrad som resultat.

7. Utbildningsprogram språk - olika nivåer

8. Utbildningsprogram matematik - olika nivåer

9. Lär dig stava-program

10. Lär dig räkna-program

11. Fysikprogram

12. Kemikprogram

13. Statistikprogram

14. Program som kan styra ljuset eller andra processer

15. Kommunikationsprogram

16. I nästa nummer av VIC-news kommer det fler programtips.

SKÄRMMATRIS

handic har tagit fram ett block med skärmmatris och oärgmatris med angivande av respektive adresser.Detta block underlättar när man vill rita upp något på skärmen och då speciellt när man använder olika färger.
Detta block tillsammans med ovannämnda programrad medger att man kan rita på skärmen med ganska enkla program.

exempelvis:

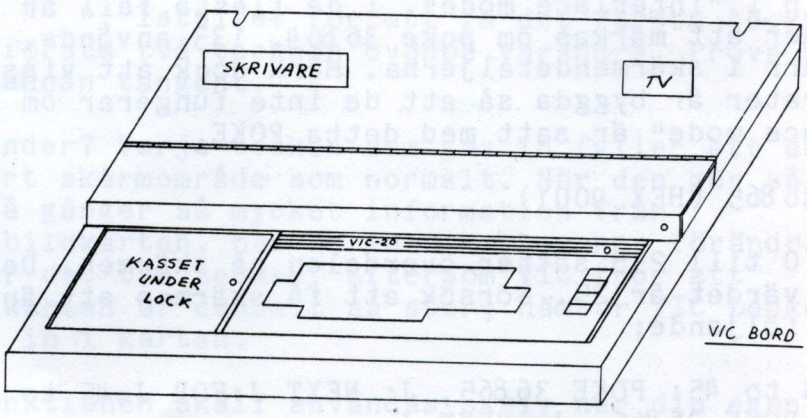
```
10?"HOME":S=PEEK(648)*256:C=PEEK(244)*256

20 FOR I = 1 TO 10:READ O(I):READ FG(I)

30 POKE S+O(I),83:POKE C+O(I),FG(I):NEXT

40 DATA 184,2,188,5,205,6,208,7,211,4,249,2,
   255,3,294,0,298,2,340,5
```

Ovanstående program ritar hjärtan på skärmen på positioner som bestämmes av data 1,3,5,7 etc.
Färgerna till varje individuellt hjärta bestämmes av värdet efter varje positionsdata.



Bokhyllebord till VIC 20

Till VIC 20 har vi utvecklat ett speciellt bokhyllebord. I bordet finns plats för din VIC 20, din kassettbandspelare (dold), din skrivare och monitor. Du bygger upp en snygg och trevlig arbetsplats med vårt bokhyllebord. Skissen visar hur bordet är byggt. Hör av dig till Lars Edvardsson på telefon 08-778 60 54 så får du mer information och priser. Du kan också skriva till mig på adress Lars Edvardsson, Skogsvägen 77, 146 00 Tullinge.

VIC chipet utvikt

VIC kallas datorn, förkortningen står då helt logiskt för Video Interface Computer. Men hör och häpna, det finns ett chip med VIC som namn i VIC-datorn. VIC stått här för Video Interface Chip. Nummerbeteckningen på chipet är 6560 eller 6561, chipet kan göra underverk på din tv-skärm.

Nybörjare har ofta inte fått grepp om att minnesadresser kan användas till annat än minneslagring. I VIC-datorn kan adresserna 36864 till 36879 PEEKas eller POOKas. Dessa adresser används inte som minne, de innehåller kontrollinformation för VIC-chipet.

Vi kommer att lära oss mycket nytt om vår vän, när vi går igenom adresserna och prövar oss fram steg för steg.

ADRESS 36864 (HEX 9000).

Värdena 0 till 127 sätter positionen för vänster kanten av skärmen. Det normala värdet är 5. Prova följande snabba "linjebyte":

```
FOR J=5 to 30: POKE 36864, J: NEXT J:FOR J=30 to 5 STEP
-1:POKE 36864, J: NEXT J
```

Om du adderar 128 till värdet i 36864, kommer skärmen att gå in i "interlace mode". I de flesta fall är allt som kommer att märkas om poke 36864, 133 används, ett litet darr i skärmendetaljerna. Märk dock att vissa TV-apparater är byggda så att de inte fungerar om inte "interlace mode" är satt med detta POKE.

ADRESS 36865 (HEX 9001)

Värdena 0 till 255 sätter överdelen på skärmen. Det vanliga värdet är 25. Försök att få skärmen att ändra sig med följande:

```
FOR J=25 to 45: POKE 36865, J: NEXT J:FOR J=45 to 25
STEP -1:POKE 36865, J: NEXT J
```

ADRESS 36866 (HEX 9002)

En del av denna adress talar om för chipet hur många kolumner som skall läggas ut på skärmen. Detta kommer alltid att vara 22. Men det finns ett extra värde på 128 som kan adderas för att sätta "alternativ" skärmläget. Normalt adderas 128 in och på adressen kommer 150 att lagras. Om en återgång till normalt skärmläge önskas, ta bort de extra 128 värdena med POKE 36866,22 och skärmen kommer att hämta sin information från ett nytt område.

ADRESS 36867 (HEX 9003)

En arbetsam "fan". Den förändras hela ju hela tiden. Försök att skriva PEEK 36867 flera gånger, det blir olika värden ibland 46 eller 174. Låt oss bortse ifrån de extra 128 för ögonblicket.

Basvärdet i denna adress är normalt 46, vilket är antalet rader multiplicerat med med 2 (det finns 23 rader). Inte behövs den ändras, eller hur?

Det finns en en funktion gömd på denna adress, och det är en viktig sådan. Om 1 adderas till värdet kommer teckengeneratorn att ändras till "dubbel karaktär läget". Detta betyder att varje karaktär kommer att dubbla sitt normala skärmutrymme.

Detta fungerar tyvärr inte automatiskt. Om vi vill göra dubbelt så stora tecken måste vi förse VIC med bilder över hur de nya tecknen skall se ut. De gamla teckenbilderna gör inte stor nytta eftersom de är för små för att fylla upp utrymmet. Var därför beredd på lite underligheter när du provar, detta på grund av att vi inte lagt upp nya tecken.

Skriv 36867,47 skärmen blir underlig. Ingen oro än så länge, tryck ner "screen clear" tangenten (skiftad naturligtvis). Skärmen blankas men cursorn blir konstig. Ingen oro vi fortsätter.

Det första tecknet i VICs teckenkarta är symbolen " ", nästa tecken är "A" sedan ett "B" och så vidare. Skriv nu ner " ". Istället för att få det första tecknet får vi de första två tecknen ovanpå varandra. Prova med någon annan tangent.

Vad händer? Varje tecken som ges in fyller ett dubbelt så stort skärmområde som normalt. När den gör så hämtar den två gånger så mycket information från teckenbildkartan, när nu inte kartan har förändrats betyder det två tecken. Eftersom VIC tror att teckenkartan är dubbelt så stor, hämtar VIC tecken längre in i kartan.

När funktionen skall användas skriv ner din egen teckenkarta och allt löser sig. Denna funktion är används oftas för hög upplösningsgrafik.

Vic återgår till normal läget genom POKE 36867,46, men det syns inget på skärmen. Om datorn slås av kommer allting att återgå till utgångsläget.

ADRESS 36868 (HEX 9004)

Denna adress ändrar sig ständigt, den är förbunden med "high-bit" (128 värdet) i föregående adress. I princip, så talar den om precis var någonstans på skärmen som bilden ritas upp. I praktiken så är det inte till mycket hjälp för BASIC programmerare, för när du läser det kommer en annan del av skärmen att vara aktiv.

ADRESS 36869 (HEX 9005)

Detta är en mycket viktig adress. Den kontrollerar adressen till videomatrisen som innehåller skärmtecknen och teckenmatrisen som innehåller teckenbilderna. Låt oss gå igenom dom.

Videomatrisen innehåller de 500 tecken eller där omkring som kan visas på skärmen. Låt oss kalkylera fram adresserna för skärmen.

Ta innehållet i adress 36869, dividera med 16, och ta bort återstoden. Detta kommer att ge dig ett nummer från 8 till 15. Subtrahera 8 och dubbla det, vilket ger ett jämnt nummer mellan 0 till 14. Nu om innehållet i 36866 är 128 eller större addera 1 för att få ett värde mellan 0 och 15. Multiplicera resultatet med 512. Vid denna punkt kommer du att ha ett värde från 0 till 7680. Det är där som videomatrisen är lokaliserad.

Detta var en kalkylering, en del av de saker som den innehåller kräver en separat artikel. För ögonblicket, observera att videoadresserna alltid måste vara i området 0 till 7680 samt en multiplikation av 512. Om du önskar att sätta upp din egen videomatris inom detta område, gör kalkyleringen åt andra hållet. Dividera med 512 subtrahera 1 om ojämnt, dividera med 2 addera 8 och slutligen multiplicera med 16. Den alternativa skärmbiten (128 värde) i 36866 är egentligen del av en mycket större skärmadress.

Teckenmatrisens adress är definerad i denna adress. Vi kommer att behöva ändra den om vi skall kunna definera nya tecken, enkla eller dubbla. Naturligtvis behöver vi också definera teckenbilder för alla tecken som vi vill skriva ut. Beräkni ngen av adressen är komplex.

Ta innehållet i adress 36869 och dividera med 16. Ta sedan återstoden och om den är större än 7, subtrahera 8. Om återstoden är mindre än 7, addera 32. Nu skall det finnas en justerad återstod som är antingen mindre än 7, eller mellan 32 och 39.

Multiplicera detta värde med 1024 och du har funnit ditt teckens adress. Det finns i intervallet 0 till 7168 eller 32768 till 39936.

Om du önskar konstruera din egen teckenmatris, är det önskvärt att den pekar på en adress i intervallet 0 till 7168. I det fallet vänd på ekvationen, ta adressen dividera med 1024 och addera 8.

Glöm inte att videomatrisen och teckenmatrisen är packade tillsammans i denna adress. Båda behöver ställas på samma gång.

Lek gärna med denna adress men om inte det planeras noga blir resultatet bara en "knasig" skärm.

ADRESS 36870 TILL 36871 (HEX 9006 OCH HEX 9007)

Det här är ingången för en ljuspenna. En ljuspenna är en "tingest" som liknar en penna och som ansluts till VIC. Peka på skärmen med ljuspennan och dessa adresser talar om var du pekar.

Kontakten för av- och påstänging av ljuspennan är inlagd i VIC-datorn men inte ansluten till VIC-chipet (det finns dock i adress 37151).

Det går att läsa X och Y positionerna för ljuspenna i adresserna 36870 och 36871. Det blir inte kolumn- eller radvärden, utan det blir värden som varierar från 0 till 255 och det behövs beräkningar som är ställda till den speciella modell av ljuspenna som används.

Var vaksam för "darr" på dessa värden. Även om inte ljuspennan rör sig kan värdena variera lite från olika inläsningar. Beroende på vad som görs kan en medelvärdesteknik behöva användas för att göra inläsningen smidigare.

En metod går ut på att att värde ignoreras om det varierar från föregående värden med ett fastställt värde.

ADRESS 36872 TILL 36873 (HEX 9008 OCH HEX 9009)

Detta är "paddle" ingångsvärden. Två "paddles" kan kopplas in och deras värden kan läsas här. Det kanske blir svårt att följa hela spektrat av rotationen.

Ännu en gång var vaksam mot "darr" på dessa värden.

Joy-stick kan också kopplas in i VIC men positionen av dess värden finns inte i VIC-chipet. De kommer i andra adresser 37151 och 37152.

ADRESSER 36874 TILL 36876 (HEX 900A TILL HEX 900C)

Detta är VICs röster. Värdesättning till 128 eller högre i någon av dessa adresser producerar ljud, värdet som POKEs producerar stämman. Genom att "poka" 2 eller 3 adresser, kan du producera toner.

Alla röster kontrolleras av ljudnivån som sätts i adress 36878, försök POKE 36878, 15 så att volymen blir bra.

Det är intressant att notera att rösten som kontrolleras av 36874 är den mildaste och att rösten i 36876 är den skarpaste. Använd 36876 för att bära melodin.

ADRESS 36877 (HEX 900D)

Denna är likartad med musikrösterna, med den skillnaden att den skapar oljud. Ett värde av 128 eller mer producerar oljud. Ju högre värde, desto högre ton på oljudet. Ännu en gång kontrolleras den från ljudkontrollen i 36878.

ADRESS 36878 (HEX 900E)

Om värdet här är mindre än 15, står det för ljudkontrollen. Om värdet är 16 eller mer står det för fler färg.

Normalt innehåller varje teckenposition 2 färger , bakgrund och förgrund. Vid användandet av fler färg kan 2 extra färger adderas till varje tecken. Gränsfärgen samt en till som vi väljer. Vi väljer denna färg i de höga delarna av adress 36878. Om vi dividerar innehållet i denna adress med 16, och bortser ifrån återstoden, får vi tillsatsfärgen.

Intressant att notera, är att varje tecken på VICs skärm kan bli utvalt oberoende av andra som en tvåfärg eller fler färg. Detta görs i färgmatrisen vilken ställer varje teckens färg.

Försök följande: Blanka skärmen och skriv in bokstaven "A" i övre vänstra hörnet. Gå nu till en ny rad och skriv POKE 38400,8. Vi kommer att märka att bokstaven A plötsligt har förändrats och blivit flerfärgad. Men att resten av skärmen är oförändrad. Notera att vi inte POKEade VIC-chipet, utan en totalt skild minnesadress som är öronmärkt för en skärmadress. För att göra ett bra jobb behövs teckenbilderna defineras.

ADRESS 36879 (HEX 900F)

Sista adressen i VIC-chipet men en flitig en. Låt oss bryta ner den i sina tre element.

Dividera innehållet i adressen med 16, och notera resultatet som skärm bakgrunds färg. Ta nu återstoden om den är 8 eller mer subtrahera 8 och notera Foreground/Background=ON. Det återstående värdet kan märkas ramfärg.

Ramfärgen är en personlig favorit, den gör det enkelt att ge en signal utan att störa innehållet i skärmen.

Om det är någon i ett program som skall utmärkas t.ex fara, fel eller spelexplosion. Kan ramen lätt göras röd genom POKE 36879, 26 och sedan återställas med POKE 36879, 27.

Ytterligare ett exempel, istället för att skriva ett "var snäll och vänta" meddelande kan skärmen ändra färg för att tala om att något händer.

Nu till Foreground/Background, normalt F/B=ON. Vi vet att vi kan skriva ner tecken i flera olika färger på en enda färgbakgrund. Men ibland kan det vara bättre att göra tvärt om.

Försök med POKE 36879, 19. Blanka nu skärmen skriv dit några tecken. Byt färg och skriv dit några till. Bakgrundsfärgen förändras men inte tecken färgen.

FUNKTIONSTANGENTERNA

Du kan givetvis använda VIC's funktionstangenter till dina egna program. Minnespositionen 197 innehåller information om någon tangent är nedtryckt.

```
100 ?PEEK(197):GOTO 100
```

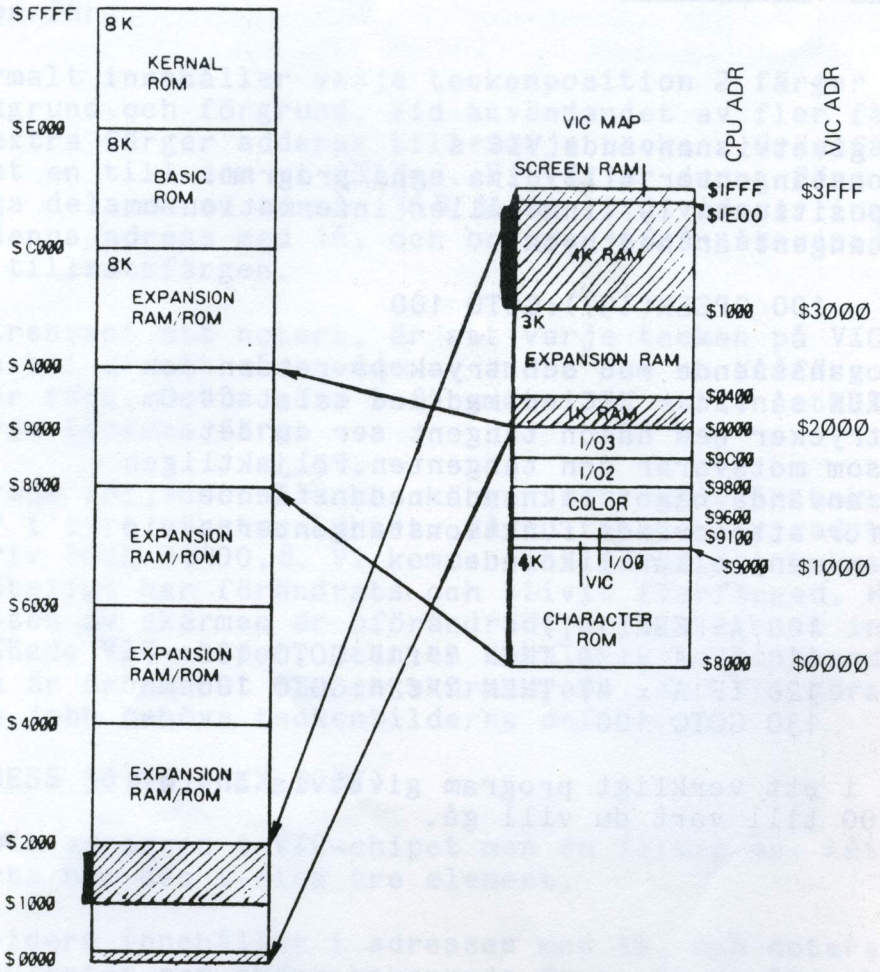
Skriv ovanstående rad och tryck på return och skriv RUN så visar VIC en rad med talet 64.Om du nu trycker ned någon tangent ser du det värde som motsvarar den tangenten.Följaktligen kan du använda något liknande nedanstående rader för att använda funktionstangenterna vid val i en meny eller liknande.

```
100 A=PEEK(197)
110 IF A = 39 THEN ?"f1":GOTO 100
120 IF A = 47 THEN ?"f2":GOTO 100
130 GOTO 100
```

Du får i ett verkligt program givetvis ändra GOTO 100 till vart du vill gå.

Bildskärmsminne och basicminne

Standard 5K

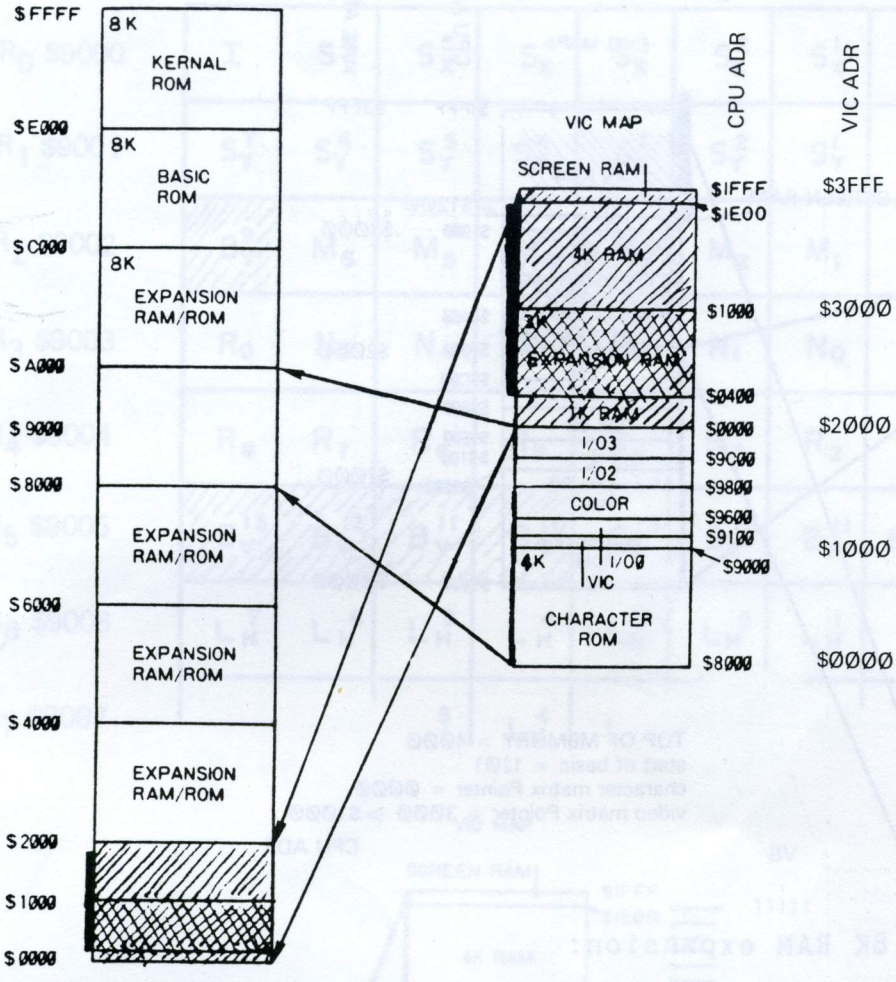


TOP OF MEMORY = 1EØØ
start of basic = 1ØØ1
character matrix Pointer = ØØØØ
video matrix Pointer = 3EØØ =>1EØØ (CPU ADR)

```
Standard VIC med 5K RAM:

Top of memory              = slutet för BAISC
text
Start of BASIC             = Början för BASIC
text
Character matrix pointer = Start för
teckengenerator
Video matrix pointer       = Start för
bildskärmsminne

3583 BYTES FREE.
```

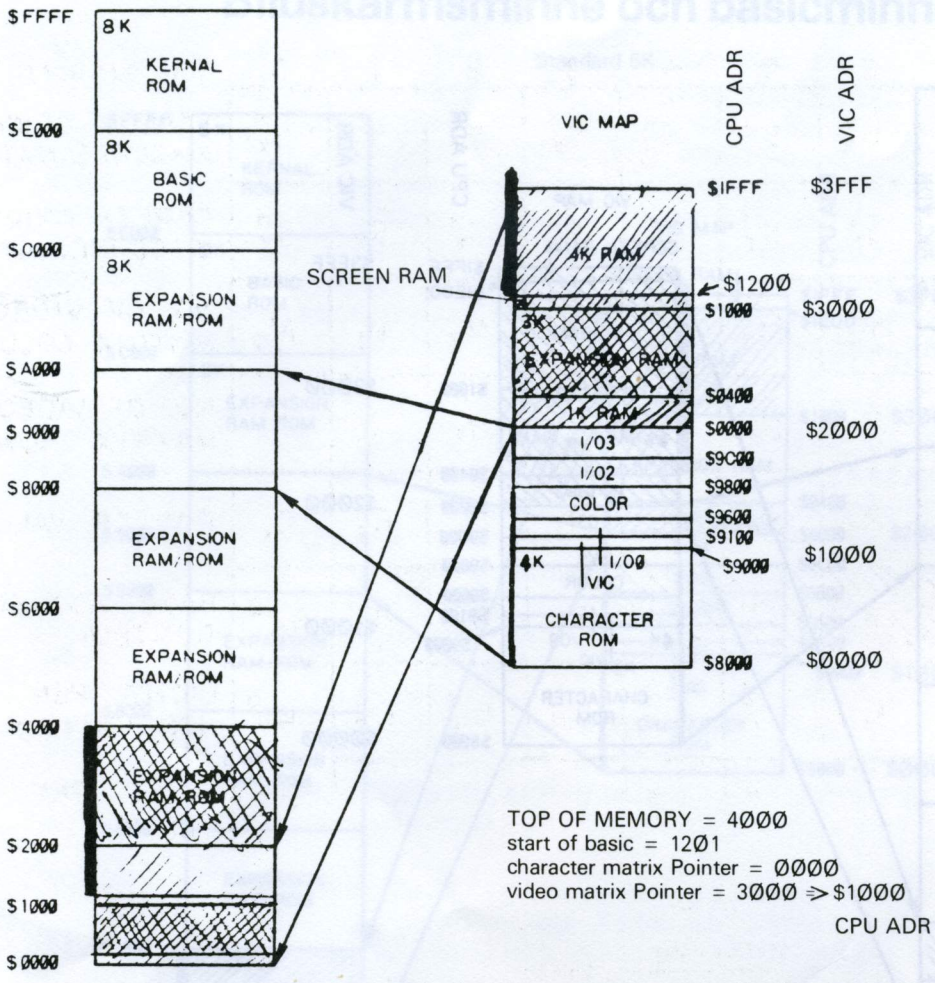


TOP OF MEMORY = 1EØØ
start of basic = Ø4Ø1
character matrix Pointer = ØØØØ
video matrix Pointer = 3EØØ =>1EØØ (CPU ADR)

VIC + 3K RAM expansion:

Det enda som skiljer sig från en standard VIC är att start of BASIC har flyttats ner till $0401.

6655 BYTES FREE.



VIC + 3K RAM + 8K RAM expansion:

Här börjar det hända saker! Bildskärmsminnet (video matrix pointer) åker ner till $1000 och start of BASIC börjar på $1200.

11775 BYTES FREE.

Varför bildskärmsminnet måste flyttas ned beror på att BASIC-arean måste vara sammanhängande utan avbrott.

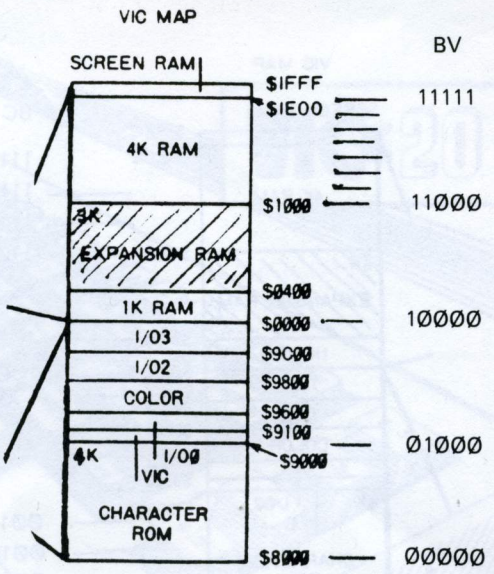
Men varför då inte flytta ned bildskärmsminnet till $0400. 3K minnet är ledigt vid denna expansion.

Anledningen varför det inte går att flytta ned bildskärmsminnet ned till $0400 beror på en 'konstruktionsteknisk finess'. VIC-chipet kan endast läsa av de minnen som finns inuti VICen. Med andra ord VIC-chipets adress och data signaler går ej ut på expansionsporten. Detta gör att VIC-chipet ej kan komma åt 3K RAMet.

Därav kommer det sig att bildskärmsminnet ej kan flyttas längre ned än $1000. Detta gäller även när teckengenaratorn skall flyttas.

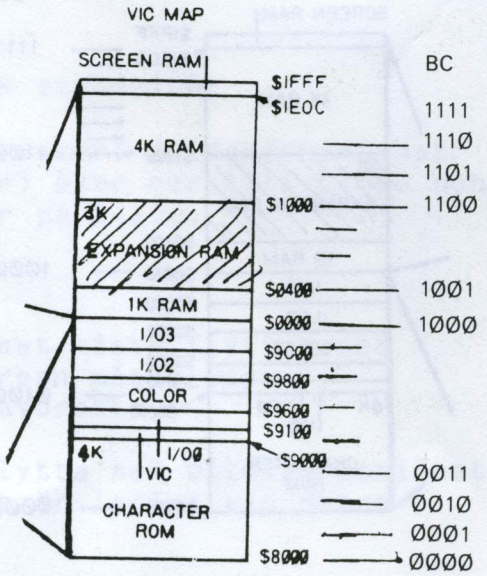
9000 ORIGIN

	7	6	5	4	3	2	1	0	(bit number)
CR_0 $9000	I	S_X^6	S_X^5	S_X^4	S_X^3	S_X^2	S_X^1	S_X^0	SCREEN ORIGIN X-COORDINATE
CR_1 $9001	S_Y^7	S_Y^6	S_Y^5	S_Y^4	S_Y^3	S_Y^2	S_Y^1	S_Y^0	SCREEN ORIGIN Y-COORDINATE
CR_2 $9002	B_V^9	M_6	M_5	M_4	M_3	M_2	M_1	M_0	NO. OF VIDEO MATRIX COLUMNS
CR_3 $9003	R_0	N_5	N_4	N_3	N_2	N_1	N_0	D	NO. OF VIDEO MATRIX ROWS
CR_4 $9004	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	RASTER VALUE
CR_5 $9005	B_V^{13}	B_V^{12}	B_V^{11}	B_V^{10}	B_C^{13}	B_C^{12}	B_C^{11}	B_C^{10}	BASE ADDRESS CONTROL
CR_6 $9006	L_H^7	L_H^6	L_H^5	L_H^4	L_H^3	L_H^2	L_H^1	L_H^0	LIGHT PEN HORIZONTAL
CR_7 $9007			5	L_V^4	1			0	LIGHT PE… VERTI…



I figuren ovan kan du se vilka bitar i vilka register som bestämmer var någonstans VIC-chipet skall gå och leta efter bildskärmsminnet. Eftersom vi kan ändra på de översta 5 bitarna och VIC-chipet adresserar 16K blir det i 512 bytes intervaller som bildskärmsminnet kan flyttas. Försöker man att flytta bildskärmsminnet till 3K RAMet, som VIC-chipet inte kan läsa, fås en mycket snabb slumpmässig presentation av tecken.

	7	6	5	4	3	2	1	0	(bit number)
CR_0 $9000	I	S_X^6	S_X^5	S_X^4	S_X^3	S_X^2	S_X^1	S_X^0	SCREEN ORIGIN X-COORDINATE
CR_1 $9001	S_Y^7	S_Y^6	S_Y^5	S_Y^4	S_Y^3	S_Y^2	S_Y^1	S_Y^0	SCREEN ORIGIN Y-COORDINATE
CR_2 $9002	B_V^9	M_6	M_5	M_4	M_3	M_2	M_1	M_0	NO. OF VIDEO MATRIX COLUMNS
CR_3 $9003	R_0	N_5	N_4	N_3	N_2	N_1	N_0	D	NO. OF VIDEO MATRIX ROWS
CR_4 $9004	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	RASTER VALUE
CR_5 $9005	B_V^{13}	B_V^{12}	B_V^{11}	B_V^{10}	B_C^{13}	B_C^{12}	B_C^{11}	B_C^{10}	BASE ADDRESS CONTROL
CR_6 $9006	L_H^7	L_H^6	L_H^5	L_H^4	L_H^3	L_H^2	L_H^1	L_H^0	LIGHT PEN HORIZONTAL
CR_7 $9007	L_V^7			L_V^4	L_V^3	L_V^2	L_V^1	L_V^0	LIGHT



Ovanstående figur visar teckengeneratorns startadress. Med dessa fyra bitar kan du välja 16 stycken olika ställen där teckengeneratorn skall börja. Samma sak för 3K RAMet gäller även här.

Skriv följande på en standard VIC:

```
PRINTCHR¤(147);:FORI=0TO127:PRINT"
";:POKE7680+I,I:NEXT:POKE9*4096+5,248
```

På skärmen visas nu hela zero-page hur den arbetar.

VICen fungera precis som vanligt, det är bara det att du får det litet svårt att tyda tecknen som skrivs på skärmen.

Notera övre högra hörnet, där ligger klockan och snurrar.

FÄRGMINNET:

Färgminnet ligger normalt i position $9600. Vid en expansion med RAM ovanför $2000 kommer färgminnet att flyttas till $9400!

Detta blir problem endast då tecknen pokas in i skärmen.

VIC 20

minnes organisation

BASIC ARBETSMINNE

FLD	00000	$0000	L=03	USR FUNCTION JMP
FLD	00003	$0003	L=02	FLOAT-FIXED VECTOR
FLD	00005	$0005	L=02	FIXED-FLOAT VECTOR
FLD	00007	$0007	L=01	SEARCH CHARACTER
FLD	00008	$0008	L=01	SCAN-QUOTES FLAG
FLD	00009	$0009	L=01	TAB COLUMN SAVE
FLD	00010	$000A	L=01	0=LOAD, 1=VERIFY
FLD	00011	$000B	L=01	INPUT BUFFER PTR/ SUBSCRIPT NUMBER
FLD	00012	$000C	L=01	DEFAULT DIM FLAG
FLD	00013	$000D	L=01	VARIABLE TYPE : FF=STRING, 00=NUMERIC
FLD	00014	$000E	L=01	VARIABLE TYPE : 80=INTEGER, 00=FLOATING
FLD	00015	$000F	L=01	'DATA' SCAN / 'LIST' QUOTE / MEMORY FLAG
FLD	00016	$0010	L=01	SUBSCRIPT / 'FN'X FLAG
FLD	00017	$0011	L=01	0=INPUT, 40=GET, 98=READ
FLD	00018	$0012	L=01	'ATN' SIGN / COMPARISON EVALUATION FLAG
FLD	00019	$0013	L=01	CURRENT I/O PROMPT FLAG
FLD	00020	$0014	L=02	INTEGER VALUE
FLD	00022	$0016	L=01	PTR TO TEMPORARY STORAGE STACK
FLD	00023	$0017	L=02	LAST TEMP STRING VECTOR
FLD	00025	$0019	L=09	STACK FOR TEMPORARY STRINGS
FLD	00034	$0022	L=04	UTILITY POINTER AREA
FLD	00038	$0026	L=05	MULTIPLICATION PRODUCT AREA
FLD	00043	$002B	L=02	PTR TO START OF BASIC
FLD	00045	$002D	L=02	PTR TO END OF PGM, START OF VARIABLES
FLD	00047	$002F	L=02	PTR TO END OF VARIABLES, START OF ARRAYS
FLD	00049	$0031	L=02	PTR TO END OF ARRAYS
FLD	00051	$0033	L=02	PTR TO LO ACTIVE STRINGS (HI TO LO)
FLD	00053	$0035	L=02	PTR TO HI ACTIVE STRINGS (HI TO LO)
FLD	00055	$0037	L=02	PTR TO END OF MEMORY
FLD	00057	$0039	L=02	CURRENT BASIC LINE NUMBER
FLD	00059	$003B	L=02	PREVIOUS BASIC LINE NUMBER
FLD	00061	$003D	L=02	PTR: 'CONT' ON BASIC STATEMENT NUMBER
FLD	00063	$003F	L=02	CURRENT 'DATA' LINE NUMBER
FLD	00065	$0041	L=02	PTR TO CURRENT DATA STATEMENT
FLD	00067	$0043	L=02	'INPUT' VECTOR
FLD	00069	$0045	L=02	CURRENT VARIABLE NAME
FLD	00071	$0047	L=02	CURRENT VARIABLE ADDRESS
FLD	00073	$0049	L=02	VARIABLE PTR FOR 'FOR'/'NEXT'
FLD	00075	$004B	L=02	Y-SAVE; OP-SAVE: BASIC PTR SAVE
FLD	00077	$004D	L=01	COMPARISON SYMBOL ACCUMULATOR
FLD	00078	$004E	L=06	MISC WORK AREA, PTRS, ETC.
FLD	00084	$0054	L=03	JUMP VECTOR FOR FUNCTIONS
FLD	00087	$0057	L=10	NUMERIC WORK AREA
FLD	00097	$0061	L=01	ACCUM#1: EXPONENT
FLD	00098	$0062	L=04	ACCUM#1: MANTISSA
FLD	00102	$0066	L=01	ACCUM#1: SIGN
FLD	00103	$0067	L=01	SERIES EVALUATION CONSTANT PTR
FLD	00104	$0068	L=01	ACCUM#1: HI-ORDER (OVERFLOW)
FLD	00105	$0069	L=06	ACCUM#2: EXPONENT, MANTISSA, SIGN
FLD	00111	$006F	L=01	SIGN COMPAR. AC#1 VS AC#2
FLD	00112	$0070	L=01	ACCUM#1: LOW ORDER (ROUNDING)
FLD	00113	$0071	L=01	CASSETTE BUFF LEN / SERIES POINTER
FLD	00115	$0073	L=18	GET BASIC CHARACTER ROUTINE

FLD	00122	$007A	L=02	BASIC PTR
FLD	00139	$008B	L=05	'RND' SEED VALUE
LAST	00143	$008F		

KERNAL ARBETSMINNE

FLD	00144	$0090	L=01	'ST' STATUS WORD
FLD	00145	$0091	L=01	KEYSWITCH PIA: 'STOP' AND 'RVS' FLAGS
FLD	00146	$0092	L=01	TIMING CONSTANT FOR TAPE
FLD	00147	$0093	L=01	LOAD=0, VERIFY=1
FLD	00148	$0094	L=01	SERIAL OUTPUT : DEFERRED CHAR FLAG
FLD	00149	$0095	L=01	SERIAL DEFERRED CHAR
FLD	00150	$0096	L=01	TAPE EOT RECEIVED
FLD	00151	$0097	L=01	REGISTER SAVE AREA
FLD	00152	$0098	L=01	HOW MANY OPEN FILES
FLD	00153	$0099	L=01	INPUT DEVICE : NORMALLY 0
FLD	00154	$009A	L=01	OUTPUT CMD DEVICE : NORMALLY 3
FLD	00155	$009B	L=01	TAPE CHAR PARITY
FLD	00156	$009C	L=01	BYTE-RECEIVED FLAG
FLD	00157	$009D	L=01	DIRECT=$80 , RUN=0: OUTPUT CNTL
FLD	00158	$009E	L=01	RS232 PASS 1 ERROR LOG / CHAR BUFFER
FLD	00159	$009F	L=01	RS232 PASS 2 ERR LOG CORRECTED
FLD	00160	$00A0	L=03	JIFFY CLOCK HML
FLD	00163	$00A3	L=01	SERIAL BIT COUNT / EOI FLAG
FLD	00164	$00A4	L=01	CYCLE COUNT
FLD	00165	$00A5	L=01	COUNTDOWN, TAPE WRITE / BIT COUNT
FLD	00166	$00A6	L=01	TAPE BUFFER PTR
FLD	00167	$00A7	L=01	RS232 RECEIVER INPUT BIT TEMP STORAGE RS232 WRT LDR COUNT / RD PASS/IN BIT
FLD	00168	$00A8	L=01	RS232 RECEIVER BIT COUNT IN RS232 WRT NEW BYTE / RD ERROR/IN BIT CNT
FLD	00169	$00A9	L=01	RS232 RECEIVER FLAG: START BIT CHECK RS232 WRT START BIT / RD BIT ERR/ST BIT
FLD	00170	$00AA	L=01	RS232 RECEIVER PTR: BYTE/BUFFER ASSEMBLY RS232 SCAN;CNT;LD;END / BYTE ASSEMBLY
FLD	00171	$00AB	L=01	RS232 RECEIVER PARITY BIT STORAGE RS232 WRT LEAD LENGTH / RD CHKSUM/PARITY
FLD	00172	$00AC	L=02	PTR TAPE BUFFER SCROLLING
FLD	00174	$00AE	L=02	TAPE END ADDR / END OF PGM
FLD	00176	$00B0	L=02	TAPE TIMING CONSTANTS
FLD	00178	$00B2	L=02	PTR: START OF TAPE BUFFER
FLD	00180	$00B4	L=01	RS232 TRANSMITTER BIT COUNT OUT 1=TIMER ENABLED; BIT COUNT
FLD	00181	$00B5	L=01	RS232 TRANSMITTER NEXT BIT TO BE SENT RS232 EOT/NEXT BIT TO SEND
FLD	00182	$00B6	L=01	RS232 TRANSMITTER PTR:BYTE/BUFF DISASSEMBLY READ CHARACTER ERROR/OUTBYTE BUFF
FLD	00183	$00B7	L=01	NUMBER OF CHARS IN FILENAME
FLD	00184	$00B8	L=01	CURRENT LOGICAL FILE
FLD	00185	$00B9	L=01	CURRENT SECONDARY ADDRESS
FLD	00186	$00BA	L=01	CURRENT DEVICE
FLD	00187	$$00BB	L=02	PTR TO FILE NAME
FLD	00189	$00BD	L=01	WR SHIFT WORD / RD INPUT CHAR
FLD	00190	$00BE	L=01	# BLOCKS REMAINING TO RD/WR
FLD	00191	$00BF	L=01	SERIAL WORD BUFFER
FLD	00192	$00C0	L=01	TAPE MOTOR INTERLOCK

FLD	00193	$00C1	L=02	I/O START ADDRESS
FLD	00195	$00C3	L=02	KERNEL SETUP PTR
FLD	00197	$00C5	L=01	MATRIX COORDINATE OF KEY DOWN
FLD	00198	$00C6	L=01	NUM CHARS IN KEYBOARD BUFFER
FLD	00199	$00C7	L=01	REVERSE MODE FLAG 18=ON
FLD	00200	$00C8	L=01	PTR: END OF LINE FOR INPUT
FLD	00201	$00C9	L=02	CURRENT CURSOR POSITION (ROW,COL)
FLD	00203	$00CB	L=01	MATRIX COORDINATE OF LAST KEY DOWN 64 IF NO KEY
FLD	00204	$00CC	L=01	CURSOR ENABLE FLAG 1=OFF 0=ON
FLD	00205	$00CD	L=01	DELAY BEFORE CURSOR BLINKS, COUNTDOWN
FLD	00206	$00CE	L=01	CHARACTER UNDER CURSOR
FLD	00207	$00CF	L=01	CURSOR BLINK FLAG
FLD	00208	$00D0	L=01	INPUT FROM SCREEN / KEYBOARD
FLD	00209	$00D1	L=02	PTR TO SCREEN LINE
FLD	00211	$00D3	L=01	POSITION OF CURSOR ON ABOVE LINE
FLD	00212	$00D4	L=01	? QUOTE MODE FLAG 0=OFF 1=ON ?0=DIRECT CURSOR, ELSE PROGRAMMED
FLD	00213	$00D5	L=01	CURRENT SCREEN LINE LENGTH
FLD	00214	$00D6	L=01	ROW WHERE CURSOR LIVES
FLD	00215	$00D7	L=01	ASCII VALUE OF LAST KEY PRESSED LAST INKEY / CHKSUM / BUFFER
FLD	00216	$00D8	L=01	NUMBER OF OUTSTANDING INSERTS
FLD	00217	$00D9	L=24	SCREEN LINE LINK TABLE
FLD	00241	$00F1	L=01	DUMMY SCREEN LINK
FLD	00242	$00F2	L=01	SCREEN ROW MARKER
FLD	00243	$00F3	L=02	SCREEN COLOR PTR
FLD	00245	$00F5	L=02	KEYBOARD PTR
FLD	00247	$00F7	L=02	RS232: PTR RECEIVER BUFFER BASE LOCATION
FLD	00249	$00F9	L=02	RS232: PTR TRANSMIT BUFFER BASE LOCATION EACH BUFFER IS 256 BYTES, MAY BE USED FOR PGMR USE, MUST ADJUST TOP OF MEMORY PTR TOO.
FLD	00251	$00FB	L=05	?????????
LAST	00255	$00FF		

BASIC & KERNAL STACK

START	00256	$0100	L=256	BASIC, KERNEL, PROCESSOR STACK AREA STACK IS HI TO LO
LAST	00511	$01FF		
START	00256	$0100	L=10	FLOATING TO ASCII WORK AREA
LAST	00266	$010A		
START	00256	$0100	L=64	TAPE ERROR LOG
LAST	00318	$0140		

BASIC OCH KERNAL ARBETSMINNE

FLD	00512	$0200	L=59	BASIC INPUT BUFFER
FLD	00601	$0259	L=10	LOGICAL FILE TABLE
FLD	00611	$0263	L=10	DEVICE NUMBER TABLE
FLD	00621	$026D	L=10	SECONDARY ADDRESS TABLE
FLD	00631	$0277	L=10	KEYBOARD BUFFER (SEE 198 $C6)
FLD	00641	$0281	L=05	?????????
FLD	00646	$0286	L=01	CURRENT COLOR CODE
FLD	00647	$0287	L=01	COLOR UNDER CURSOR
FLD	00648	$0288	L=01	SCREEN MEMORY PAGE

FLD	00649	$0289	L=01	MAX SIZE OF KEYBOARD BUFFER
FLD	00650	$028A	L=01	REPEATER FLAGS 0=NORM 127=NONE 128=ALL
FLD	00651	$028B	L=01	DELAY BEFORE REPEAT OCCURS
FLD	00652	$028C	L=01	DELAY BETWEEN REPEATS
FLD	00653	$028D	L=01	FLAG FOR SHIFT/CONTROL KEYS LAST USED 255=NONE 159=SHIFT 244=COMMODORE LOGO
FLD	00654	$028E	L=05	?????????
FLD	00659	$0293	L=01	RS232: PSEUDO 6551 CNTL REGISTER
FLD	00660	$0294	L=01	RS232: PSEUDO 6551 COMMAND REGISTER
FLD	06611	$0295	L=02	RS232: RESERVED FOR FUTURE USE
FLD	00663	$0297	L=01	RS232: STATUS REGISTER
FLD	00664	$0298	L=01	RS232: NUM BITS TO BE SENT / RECEIVED
FLD	00665	$0299	L=02	RS232: SYSTEM CLOCK DIVIDED BY BAUD RATE
FLD	00667	$029B	L=01	RS232: INDEX — END OF RECEIVE FIFO BUFFER
FLD	00668	$029C	L=01	RS232: INDEX — START RECEIVE FIFO BUFFER
FLD	00669	$029D	L=01	RS232: INDEX — START TRANSMIT FIFO BUFF.
FLD	00670	$029E	L=01	RS232: INDEX — END OF TRANSMIT FIFO BUFF.
FLD	00671	$029F	L=96	?????????
FLD	00768	$0300	L=02	ERROR MSG LINK
FLD	00770	$0302	L=02	BASIC WARM START LINK
FLD	00772	$0304	L=02	CRUNCH BASIC TOKENS LINK
FLD	00774	$0306	L=02	PRINT TOKENS LINK
FLD	00776	$0308	L=02	START NEW BASIC CODE LINK
FLD	00778	$030A	L=02	GET ARITHMETIC ELEMENT LINK
FLD	00780	$030C	L=08	?????????
FLD	00788	$0314	L=02	JUMP VECTOR TO IRQ INTERRUPT ENTRY
FLD	007909	$0316	L=02	JUMP VECTOR TO BRK INTERRUPT ENTRY
FLD	00792	$0318	L=02	JUMP VECTOR TO NMI INTERRUPT ENTRY
FLD	00794	$031A	L=02	JUMP VECTOR TO OPEN LOGICAL FILE
FLD	00796	$0318	L=02	JUMP VECTOR TO CLOSE LOGICAL FILE
FLD	00798	$031E	L=02	JUMP VECTOR TO SET INPUT DEVICE
FLD	00800	$0320	L=02	JUMP VECTOR TO SET OUTPUT DEVICE
FLD	00802	$0322	L=02	JUMP VECTOR TO RESET DEFAULT I/O
FLD	00804	$0324	L=02	JUMP VECTOR TO INPUT FROM DEVICE
FLD	00806	$0326	L=02	JUMP VECTOR TO OUTPUT TO DEVICE
FLD	00808	$0328	L=02	JUMP VECTOR TO TEST STOP KEY
FLD	00810	$032A	L=02	JUMP VECTOR TO GET FROM KEYBOARD
FLD	00812	$032C	L=02	JUMP VECTOR TO CLOSE ALL FILES
FLD	00814	$032E	L=02	JUMP VECTOR TO ?????????????????
FLD	00816	$0330	L=02	JUMP VECTOR TO LOAD FROM DEVICE
FLD	00818	$0332	L=02	JUMP VECTOR TO SAVE TO DEVICE
FLD	00820	$0334	L=08	?????????
LAST	00827	$033B		

KASSETTBUFFER

START	00828	$033C	L=195	
LAST	01023	$03FF		

3K EXPANSION RAM

START	01024	$0400	L=3072	
LAST	04095	$0FFF		

BASIC TEXT, VARIABLER, MATRISER, STRÄNGAR

START	04096	$1000	L=3583	(IF MORE THAN 8K RAM IN VIC, MOVES TO $1200)

				(AND THIS AREA IS SCREEN MEMORY (NEE $1E00))
FLD	04096	$1000	L=??	BASIC PGM IN TOKENIZED FORMAT (PTR 43,44)
FLD	?????	$????	L=??	VARIABLES LO TO HI (PTRS 45,46,47,48)
FLD	?????	$????	L=??	ARRAYS LO TO HI (PTRS 47,48 49,50)
FLD	?????	$????	L=??	ANY UNUSED AREA
FLD	?????	$????	L=??	STRINGS HI TO LO (PTRS 53,54 51,52)
FLD	07168	$1000	L=512	IF 36869 $9005 = 255,512 BYTES BEGINNING HERE CONTAIN 64 USER CHARACTERS. USER POINTS TOP-OF-MEMORY PTR (55,56) TO HERE THEN CLR. SHOULD ALSO COPY FIRST 64 STD CHARS FROM 32768 $8000 INTO THIS AREA, THEN OVERLAY. NOTE THAT IF A CHAR IS REV. MODE OR L/C THEN NOT FROM HERE. ONLY POKE CHAR CODES 00-63 FROM HERE. PRINT "@" OR POKE N,0 IS FIRST USER CHAR
LAST	07679	$1DFF		

BILDSKÄRMMINNE

START	07680	$1E00	L=512	
				23 LINES OF 22 CHARACTERS
				(IF MORE THAN 8K RAM IN VIC, MOVES TO $1000)
LAST	08191	$1FFF		

EXPANSION RAM / ROM

START	08192	$2000	L=8192	
				BLOCK 1
LAST	16383	$3FFF		

8K EXPANSION RAM / ROM

START	16384	$4000	L=8192	
				BLOCK 2
LAST	24575	$5FFF		

8K EXPANSION RAM / ROM

START	24576	$6000	L=8192	
				BLOCK 3
LAST	32767	$7FFF		

TECKEN GENERATOR ROM

START	32768	$8000	L=4096	
				(8 BYTES = 1 CHAR BIT MAP * 128
FLD	32768	$8000	L=1024	UPPER CASE AND GRAPHICS
FLD	33792	$8400	L=1024	REVERSED UPPER CASE ANND GRAPHICS
FLD	34816	$8800	L=1024	UPPER AND LOWER CASE
FLD	35840	$8C00	L=1024	REVERSED UPPER AND LOWER CASE
LAST	36863	$8FFF		

VIC ADRESS

START	36864	$9000	L=272		
					VIC 6560 CONTROL REGISTERS
FLD	36864	$9000	L=01	CRO	BITS 0-6 SCREEN HORIZ CENTER
					BIT 7 INTERLACE SCAN

FLD	36865	$9001	L=01	CR1	SCREEN VERT CENTER
FLD	36866	$9002	L=01	CR2	BITS 0-6 NUM VIDEO COLUMNS
					BIT 7 PART OF VIDEO MATRIX ADDRESS
FLD	36867	$9003	L=01	CR3	BITS 1-6 NUM VIDEO ROWS
					BIT 0 8X8 OR 16X8 CHARACTERS
FLD	36868	$9004	L=01	CR4	TV RASTER BEAM LINE
FLD	36869	$9005	L=01	CR5	CHARACTER CELL BASE ADDRESS BITS
					BITS 0-3 START OF CHARACTER INFO
					BITS 4-7 REST OF VIDEO ADDRESS
					240 = U/C + GRAPH 242 = U/C + LC
					255 = USER 64 CHARS AT 7168 L=512
FLD	36870	$9006	L=01	CR6	LIGHT PEN HORIZONTAL
FLD	36871	$9007	L=01	CR7	LIGHT PEN VERTICAL
FLD	36872	$9008	L=01	CR8	PADDLE-X VALUE
FLD	36873	$9009	L=01	CR9	PADDLE-Y VALUE
FLD	36874	$900A	L=01	CRA	LOW AUDIO OSC. FREQ. 128-255 3 OCT.
FLD	36875	$900B	L=01	CRB	MED " " " " "
FLD	36876	$900C	L=01	CRC	HIGH " " " " " "
FLD	36877	$900D	L=01	CRD	NOISE " " " " " "
FLD	38678	$900E	L=01	CRE	VOLUME 0-15 AND MULTICOLOR AUX. CODE
					BITS 0-3 SOUND VOLUME
					BITS 4-7 AUX. COLOR CODE FOR MULTICOLOR
FLD	36879	$900F	L=01	CRF	SCREEN / BORDER COLOR CODE 0-255
					BITS 0-2 BORDER COLOR
					IF BIT 3 FOREGROUND COLOR BECOMES
					BACKGROUND COLOR AND
					VICE VERSA FOR EACH CHAR.
					BITS 4-7 BACKGROUND COLOR
FLD	36880	$9010	L=256		?????????
LAST	37135	$910F			

I/O BLOCK 0

START	37136	$9110	L=16	6522 PIA #1
FLD	37136	$9110	L=01	PORT B OUTPUT REGISTER (USERPORT)
FLD	37137	$9111	L=01	PORT A OUTPUT REGISTER
				BIT2 = JOY 0
				BIT3 = JOY 1
				BIT4 = JOY 2
				BIT5 = LIGHT PEN / FIRE BUTTON
				BIT6 = SENSE CASSETTE SWITCH
				BIT7 = SERIAL ATN OUT
FLD	37138	$9112	L=01	DATA DIRECTION REGISTER B
FLD	37139	$9113	L=01	DATA DIRECTION REGISTER A
FLD	37140	$9114	L=01	TIMER 1 LOW BYTE
FLD	37141	$9115	L=01	TIMER 1 HIGH BYTE AND COUNTER
FLD	37142	$9116	L=01	TIMER 1 LOW BYTE
FLD	37143	$9117	L=01	TIMER 1 HIGH BYTE
FLD	37144	$9118	L=01	TIMER 2 LOW BYTE
FLD	37145	$9119	L=01	TIMER 2 HI BYTE
FLD	37146	$911A	L=01	SHIFT REGISTER
FLD	37147	$911B	L=01	AUX CONTROL REGISTER
FLD	37148	$911C	L=01	PERIPHERAL CTRL REGISTER (CA1,CA2,CB1,CB2)
				CA1 = RESTORE KEY
				CA2 = CASSETTE MOTOR CONTROL
FLD	37149	$911D	L=01	INTERRUPT FLAG REGISTER
FLD	37150	$911E	L=01	INTERRUPT ENABLE REGISTER

FLD	37151	$911F	L=01	MPS 6522 OUTPUT REG (PORT B) JOY FIRE SW PORT A(SENSE CASSETTE SWITCH) RS232: BIT 7 ON INDICATES DATA SENT
START	37152	$9120	L=16	6522 PIA #2
FLD	37152	$9120	L=01	PORT B OUTPUT REGISTER KEYBOARD COLUMN SCAN BIT 3 = CASSETTE WRITE LINE BIT 7 = JOY 3
FLD	37153	$9121	L=01	PORT A OUTPUT REGISTER KEYBOARD ROW SCAN
FLD	37154	$9122	L=01	DATA DIRECTION REGISTER B
FLD	37155	$9123	L=01	DATA DIRECTION REGISTER A
FLD	37156	$9124	L=01	TIMER 1 LOW BYTE LATCH
FLD	37157	$9125	L=01	TIMER 1 HIGH BYTE LATCH
FLD	37158	$9126	L=01	TIMER 2 LOW BYTE COUNTER
FLD	37159	$9127	L=01	TIMER 1 HIGH BYTE COUNTER TIMER 1 USED FOR 60 TIME/SEC INTERRUPT
FLD	37160	$9128	L=01	TIMER 2 LOW BYTE LATCH
FLD	37161	$9129	L=01	TIMER 2 HI BYTE LATCH
FLD	37162	$912A	L=01	SHIFT REGISTER
FLD	37163	$912B	L=01	AUX CONTROL REGISTER
FLD	37164	$912C	L=01	PERIPHERAL CTRL REGISTER (CA1,CA2,CB1,CB2)) CA1 = CASSETTE READ LINE CA2 = SERIAL CLOCK OUT CB1 = SERIAL DATA OUT CB2 = SERIAL SRQ OUT
FLD	37165	$912D	L=01	INTERRUPT FLAG REGISTER
FLD	37166	$912E	L=01	INTERRUPT ENABLE REGISTER
FLD	37167	$912F	L=01	MPS6522 OUTPUT REG (PORT A)
FLD	37168	$9130	L=719	?????????
LAST	37887	$93FF		

FÄRG RAM

START	38400	$9400	L=1024	—COLOR CODES LOCATION MAY VARY—
FLD	37888	$9400	L=512	(COLOR VALUES IF MORE THAN 8K RAM IN VIC)
FLD	38400	$9600	L=512	COLOR VALUES IF LESS THAN 8K RAM IN VIC BIT 0-2 = COLOR VALUE, BIT 3 = MULTICOLOR=1 OR HIRES=0
LAST	38911	$97FF		

I/O BLOCK 2

START	38912	$9800	L=1024
LAST	39935	$9BFF	

I/O BLOCK 3

START	39936	$9C00	L=1024
LAST	40959	$9FFF	

8K EXPANSION ROM

START	40960	$A000	L=8192

LAST	49151	$BFFF		

8K BASIC ROM

START	49152	$C000	L=8192	
LAST	57343	$DFFF		

8K KERNAL ROM

START	57344	$E000	L=8192	
FLD	60095	$EABF	L=??	ROUTINE: IRQ INTERRUPT ENTRY
FLD	61941	$F1F5	L=??	ROUTINE: GET FROM KEYBOARD
FLD	61966	$F20E	L=??	ROUTINE: INPUT FROM DEVICE
FLD	62074	$F27A	L=??	ROUTINE: OUTPUT TO DEVICE
FLD	62151	$F2C7	L=??	ROUTINE: SET INPUT DEVICE
FLD	62217	$F309	L=??	ROUTINE: SET OUTPUT DEVICE
FLD	62282	$F34A	L=??	ROUTINE: CLOSE LOGICAL FILE
FLD	62447	$F3EF	L=??	ROUTINE: CLOSE ALL FILES
FLD	62451	$F3F3	L=??	ROUTINE: RESET DEFAULT I/O
FLD	62474	$F40A	L=??	ROUTINE: OPEN LOGICAL FILE
FLD	62793	$F549	L=??	ROUTINE: LOAD FROM DEVICE
FLD	63109	$F685	L=??	ROUTINE: SAVE TO DEVICE
FLD	63344	$F770	L=??	ROUTINE: TEST STOP KEY
FLD	65197	$FEAD	L=??	ROUTINE: NMI INTERRUPT ENTRY
FLD	65234	$FED2	L=??	ROUTINE: BRK INTERRUPT ENTRY
FLD	65412	$FF84	L=03	CALL: READ/SET VECTORED I/O
FLD	65415	$FF87	L=03	CALL: RESTORE DEFAULT I/O VECTORS
FLD	65418	$FF8A	L=03	CALL: RESTORE OLD I/O VECTORS
FLD	65421	$FF8D	L=03	CALL: READ/SET VECTORED I/O
FLD	65424	$FF90	L=03	CALL: CONTROL KERNAL MESSAGES
FLD	65427	$FF93	L=03	CALL: TRANSMIT SEC. CMD ADDR AFT LISTEN
FLD	65430	$FF96	L=03	CALL: SEND SECONDARY ADDR AFTER TALK
FLD	65433	$FF99	L=03	CALL: READ/SET TOP OF MEMORY
FLD	65436	$FF9C	L=03	CALL: READ/SET BOTTOM OF MEMORY
FLD	65439	$FF9F	L=03	CALL: SCAN KEYBOARD
FLD	65442	$FFA2	L=03	CALL: SET TIMEOUT ON IEEE BUS
FLD	65445	$FFA5	L=03	CALL: INPUT BYTE FROM IEEE BUS
FLD	65448	$FFA8	L=03	CALL: OUTPUT BYTE TO IEEE BUS
FLD	65451	$FFAB	L=03	CALL: COMMAND IEEE BUS TO UNTALK
FLD	65454	$FFAE	L=03	CALL: COMMAND IEEE BUS TO UNLISTEN
FLD	65457	$FFB1	L=03	CALL: COMMAND IEEE TO LISTEN
FLD	65460	$FFB4	L=03	CALL: COMMAND IEEE TO TALK
FLD	65463	$FFB7	L=03	CALL: READ I/O STATUS WORD
FLD	65466	$FFBA	L=03	CALL: SET LOGICAL FIRST, SECOND ADDR
FLD	65469	$FFBD	L=03	CALL: SET FILE NAME INFO
FLD	65472	$FFC0	L=03	JMP (INDIR) OPEN FILE
FLD	65465	$FFC3	L=03	JMP (INDIR) CLOSE FILE
FLD	65478	$FFC6	L=03	JMP (INDIR) OPEN INPUT CHANNEL
FLD	65481	$FFC9	L=03	JMP (INDIR) OPEN OUTPUT CHANNEL
FLD	65484	$FFCC	L=03	JMP (INDIR) CLOSE I/P AND O/P CHANNEL
FLD	65487	$FFCF	L=03	JMP (INDIR) INPUT FROM DEVICE
FLD	65490	$FFD2	L=03	JMP (INDIR) OUTPUT TO DEVICE
FLD	65493	$FFD5	L=03	JMP (INDIR) LOAD FROM DEVICE

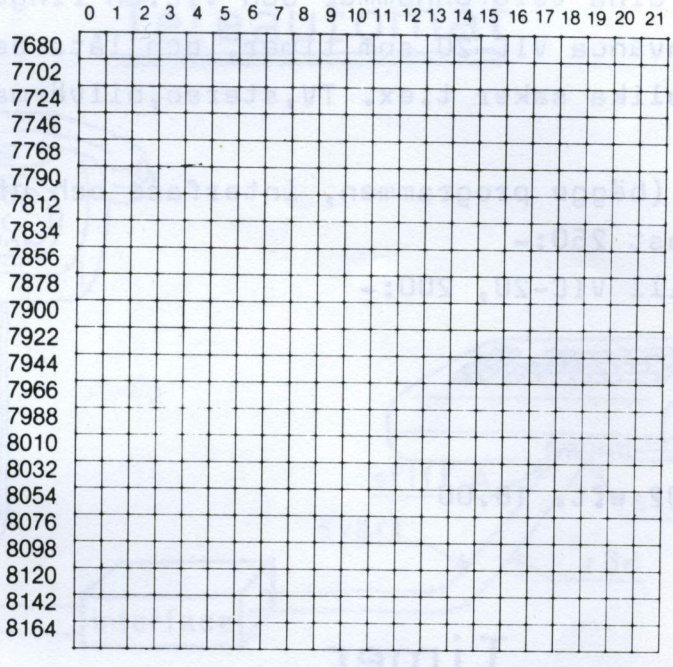
FLD	65496	$FFD8	L=03	JMP (INDIR) SAVE TO DEVICE
FLD	65499	$FFDB	L=03	JMP (INDIR) SET REAL TIME CLOCK
FLD	65502	$FFDE	L=03	JMP (INDIR) READ REAL TIME CLOCK
FLD	65505	$FFE1	L=03	JMP (INDIR) TEST STOP KEY
FLD	65508	$FFE4	L=03	JMP (INDIR) GET FROM KEYBOARD
FLD	65511	$FFE7	L=01	JMP (INDIR) CLOSE ALL FILES
FLD	65512	$FFE8	L=02	JMP (INDIR) GET FROM KEYBOARD QUEUE
FLD	65514	$FFEA	L=03	JMP (INDIR) INCREMENT REAL TIME CLOCK
FLD	65517	$FFED	L=03	JMP (INDIR) RETURN X, Y ORG OF SCREEN
FLD	65520	$FFF0	L=03	JMP (INDIR) READ/SET X, Y CURSOR POSITION
FLD	65523	$FFF3	L=03	JMP (INDIR) RETURN BASE ADDR OF I/O DEV
FLD	65511	$FFE7	L=01	JMP (INDIR) CLOSE ALL FILES
FLD	65512	$FFE8	L=02	JMP (INDIR) GET FROM KEYBOARD QUEUE
FLD	65514	$FFEA	L=03	JMP (INDIR) INCREMENT REAL TIME CLOCK
FLD	65517	$FFED	L=03	JMP (INDIR) RETURN X, Y ORG OF SCREEN
FLD	65520	$FFF0	L=03	JMP (INDIR) READ/SET X, Y CURSOR POSITION
FLD	65523	$FFF3	L=03	JMP (INDIR) RETURN BASE ADDR OF I/O DEV
LAST	65535	$FFFF		

VIC S & C

När VIC körs utan extra minne eller med endast 3K expansionsminne börjar bildskärmminnet vid 7680 och färgminnet vid 38400.
Med minnesexpansion över 3K flyttar skärmminnet till 4096 och färgminnet till 37888.
Detta är inget man behöver ta särskild hänsyn till i program med enbart PRINT-instruktioner för utskrift på bildskärmen.
Om man däremot använder POKE eller PEEK instruktioner måste man tänka på vilken minnesexpansion VIC använder.
Ett sätt att komma runt detta problem är att låta VIC själv kontrollera var bild och färg-minnesareorna startar och själv ge variablerna S (SKÄRM) och C (FÄRG) adressen till dessa.S är alltså adressen till skärmens övre vänstra hörn.C är följdaktligen adressen till färg-minnet för skärmens övre vänstra hörn.
Med POKE S+252,83 får man ett hjärta mitt på skärmen.Med POKE C+252,5 blir hjärtat grönt
Den enda programrad som krävs är följande:

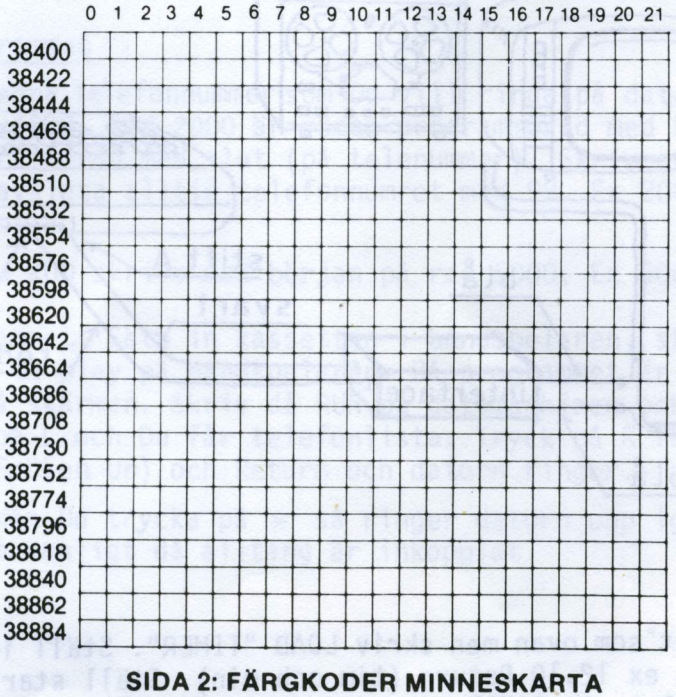
```
10 ?"HOME":S=PEEK(648)*256:C=PEEK(244)*256
```

S = 7680 (skärmstart) Vid > 3K minnesutbyggnad 4096



SIDA 1: SKÄRMTECKEN KODER

C = 38400 (färgminnestart) Vid > 3K minnesutbyggnad 37888



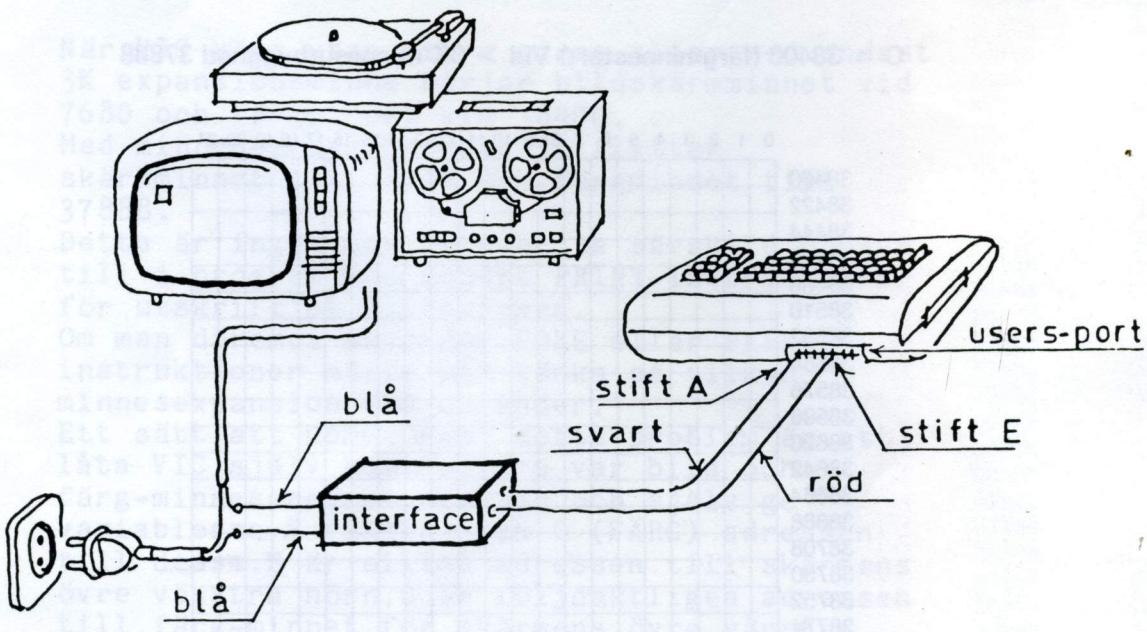
SIDA 2: FÄRGKODER MINNESKARTA

INTRODUKTIONSERBJUDANDE

VIC-20 ägare använd din dator som automatisk telefonuppringare. Programmera in dina telefonnummer och VIC:en ringer upp åt dig. Du kan också använda VIC-20 som timer, och låta den koppla på och stänga av olika saker t.ex. TV,stereo,bilvärmare,lampor, motorer etc.
Komplett paket (bägge programmen, interface och utförlig bruksanvisning) endast 250:-
Datahoroskop till VIC-20, 200:-

Kjell löfström
Yxgränd 5
183 43 Täby
Tel. 0762/116 02 eft. 18.00

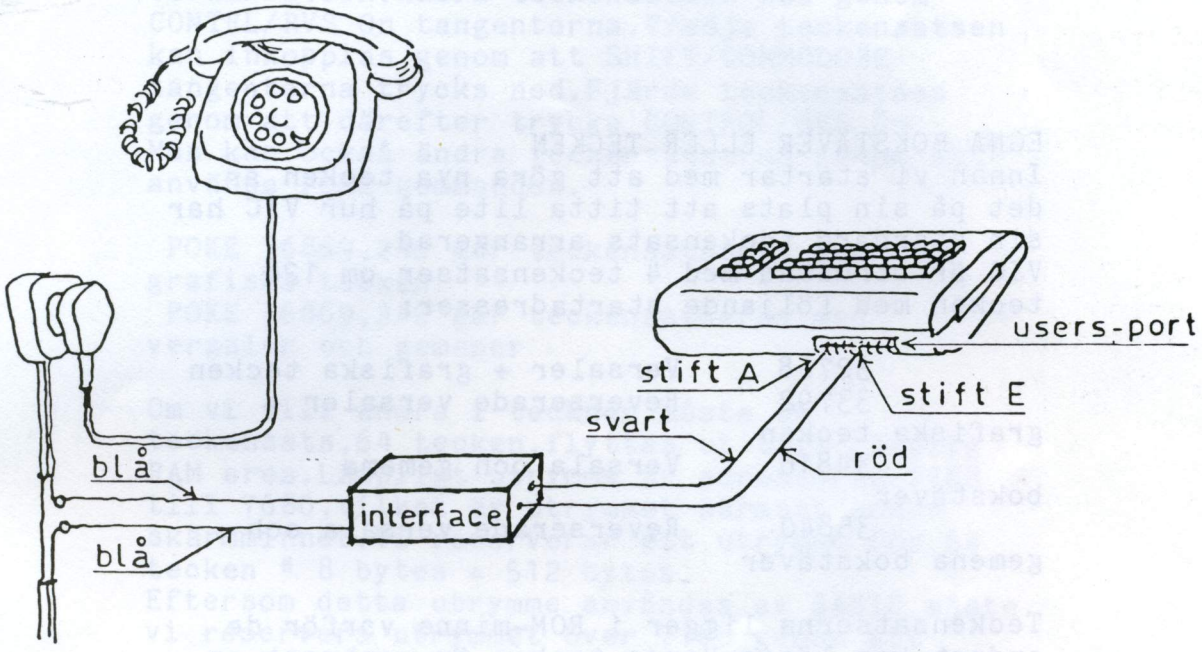
Timer



Timer

Ladda in programmet som ovan men skriv LOAD "TIMER". Ställ in rätt tid genom att skriva t ex 12,10 Return (tim och min). Ställ starttid genom att skriva t ex 12,11 Return. Ställ stopptid genom att skriva ex 12,12 Return. Tryck på en tangent och klockan startar. Sekunderna syns längst upp i vänstra hörnet.

Teleautomat



(c) Kjell Löfström

Teleautomat-programmet

Skriver in Dina egna telefonnummer som Du vill ringa på datarader inom intervallet 2010-4080. Rad 2000 är redan programmerad med Fröken Ur och rad 4090 är programmerad med slut (på telenummer). Skriv som exempel Fröken Ur. OBS! avsluta alltid telefonnumret med 99. Ex 2000 DATA "A",9,0,5,1,0,99

Telefonlistan med KOD skrivs med början på rad 5000. Ex 5000 PRINT "A=FR.UR 90510".
Körning av programmet: Sätt in kassetten i bandspelaren, skriv LOAD "TELE" på datorn, tryck in play på bandspelaren. Då programmet är inmatat i datorn står det READY på skärmen. Skriv då RUN på datamaskinen och Du har tre alternativ. Tryck på L och Du får telefonlista. Tryck på R för att ringa. Tryck på A (för Fröken Ur) och Return och datorn ringer.

Är det upptaget kan Du trycka på * så ringer datorn upp igen. Telefonen fungerar även som vanligt då allting är inkopplat.

EGNA TECKEN

EGNA BOKSTÄVER ELLER TECKEN
Innan vi startar med att göra nya tecken är det på sin plats att titta lite på hur VIC har sin standard teckensats arrangerad.
VIC är utrustad med 4 teckensatser om 128 tecken med följande startadresser:

32768	Versaler + grafiska tecken
33792	Reverserade versaler + grafiska tecken
34816	Versala och gemema bokstäver
35840	Reverserade versala och gemena bokstäver

Teckensatserna ligger i ROM-minne varför de endast kan läsas.Varje tecken är uppbyggt av en 8 * 8 matris,vilket kräver 8 bytes minnesutrymme per tecken.De första 8 bytena skapar tecknet" "(snabel-a),vilket har bildskärmkod 0.Nästa 8 byte skapar tecknet"A"vilket har bildskärmkod 1.Se bild.
Teckensatserna med bildskärmkoder finns på sidan 141 i användarmanualen. Vi kan se på en teckensats genom följande programrader:

```
 10  FOR I = 0 TO 127:POKE 7680+I,I:REM
SKÄRMADRESS,MAX 3K EXTRA MINNE
 20  POKE POKE 38400+I,0:REM
FÄRGMATRISADRESS+TECKENFÄRG 0 (SVART)
 30  NEXT
```

Skärmen visar nu tecknen i den första teckensatsen.
Varje byte i en teckenmatris motsvarar binära värdet av punkterna i en rad med början med övre raden.Se bild för bildskärmkod 0 i första teckensatsen.

VIC startar alltid med första teckensatsen.Andra teckensatsen nås genom CONTRL/RVS On tangenterna.Tredje teckensatsen kan inkopplas genom att SHIFT/COMMODORE tangenterna trycks ned.Fjärde teckensatsen genom att därefter trycka CONTROL/RVS On.
Man kan också ändra teckensatserna genom att använda POKE kommandon.

POKE 36869,240 ger teckensatserna med grafiska tecken
POKE 36869,242 ger teckensatserna med versaler och gemener

Om vi vill ändra i tecknen måste en teckensats,64 tecken,flyttas ut till lämplig RAM area.Lämpligt utrymme är adresserna 7168 till 7680,vilket är utrymmet närmast under skärmminnet.Vi reserverar ett utrymme för 64 tecken * 8 bytes = 512 bytes.
Eftersom detta utrymme användes av BASIC måste vi reservera utrymmet över 7168 genom att sätta pekarna för BASIC-utrymmets övre gräns 512 bytes(=2 pages)lägre än normalt genom instruktionen POKE56,PEEK(56)-2.Om vi inte flyttat pekarna hade BASIC programmet hela tiden skrivit i vår nya teckensats.
Nu är det dags att flytta ut vår teckensats från tecken-ROM till det reserverade området 7168 - 7680.Det gör vi med strängen:

```
FOR I = 0 TO 511:POKE 7168+I,PEEK
(32768+I):NEXT
```

När vi nu flyttat ut teckensatsen måste vi instruera VIC-chippet var tecknen nu finns att hämta.Detta sker genom att skriva:

```
POKE 36869,255
```

Nu har vi en teckensats i RAM vilken vi har möjlighet att ändra i.Låt oss ändra första tecknet " " till ett ö.Se fig .
Vi måste alltså läsa in nya data i 8 bytes med

start 7168.

```
 FOR I = 0 TO 7:READ J:POKE 7168+I,J:NEXT
 DATA 18,0,12,18,33,33,18,12
```

Det färdiga programmet för att göra ovanstående bör alltså se ut så här:

```
100  POKE 56,PEEK(56)-2:CLR:REM
   RESERVERA UTRYMME
110
FORI=0TO511:POKE7168+I,PEEK(32768+I):NEXT:REM
FLYTTA TECKEN
120  POKE 26869,255:REM
   AKTIVERA NYA TECKEN
130  FORJ=0TO7:READA:POKE7168+J,A:NEXT:REM
   ERSÄTT " " MED Ö
140  DATA 18,0,12,18,33,33,18,12
```

När du kört detta program kan du skriva ö på skärmen genom att trycka på " ".
Givetvis kan du byta ut flera tecken om du så önskar.Detta program ersätter dessvärre inte å,ä och ö perfekt eftersom markören blir osynlig.Vidare måste man tänka på att nyskapade bokstäver som inte ersätter bokstäver inte kan användas som variabler.
Vill man ersätta ytterligare tecken i teckensatsen gör man följande:
Leta rätt på skärmkoden för det tecken du vill ersätta.Om vi vill ersätta pundtecknet har det skärmkoden 28.Se användarmanuale sidan 141.
Vill vi nu ändra föregående programrader så att vi får ett "Ä" istallet för pundtecknet skriver vi:

```
   131 FOR
I=0TO7:READA:POKE7168+28*8+I,A:NEXT
   132 DATA 66,24,36,66,126,66,66,00
```

Du kan givetvis efter samma principer göra figurer till något spel om du så önskar.

POT X och Y

Pot x och y ingångarna på VIC's högersida är analogingångar som avger ett värde i VIC-chippets register CR8 respektive CR9.Dessa adresser är 4096*9+CR eller 36872 för X och 36873 för Y.
För att få ut dessa värden ska en variabel resistans ligga mellan + (pinne 7) och respektive POT ingång.Pot X anslutes till stift 9 och Y till stift 5.Värdet på denna resistans kan vara 25 kOhm.

```
100 ?"CLR":XPOT=PEEK(36872):YPOT=PEEK(36873)
120 X¤="22*cursor left"
130 Y¤="22*cursor down"
140 ?"home+23*blanks"
150 ?"homeX=";X,"Y=";Y
160 ?"home"LEFT¤(X¤,XPOT)+LEFT¤(Y¤,YPOT);"boll"
170 GOTO100
```

Ovanstående programavsnitt ger en boll på skärmen vilken kan styras i Y (vertikal) och X (horisontell) led.Dessutom visas värdena X och Y i överkanten.Potentiometerkonstanten 2 stämmer hyggligt för 50 kOhm potentiometrar.Genom att trycka på någon tangent raderas skärmen. Detta enkla system ger en dålig upplösning men ger en ide' om hur X och Y ingångarna reagerar.Den händige kan naturligtvis byta ut exempelvis pot Y mot ett lämpligt temperaturberoende motstånd och låta den i VIC inbyggda timern styra X axeln varefter VIC fungerar som en temperaturskrivare.
Kom gärna med ide'er och förslag till andra applikationer på dessa ingångar

VIC-20 FOLKDATORN

handic electronic ab

CIRKAPRISLISTA

Gäller från den 16/11 1981

Produkt	Cirkapris inkl. 21,51% moms	Cirkapris exkl. moms
VIC-20, CPU Inkl. nättransformator, RF-generator	2.499:–	2.056:–
ANVÄNDAR-MANUAL	79:–	65:–
VIC-1515, PRINTER Matrisskrivare, 80 tecken per rad, traktormatning. Trycker alla VIC-20's grafiska symboler. Inbyggd nätdel.	3.249:–	2.675:–
VIC-1530, KASSETT-BANDSPELARE med räkneverk. Bandspelarens motor styrs från VIC-20	549:–	450:–
VIC-1540, SINGLE DRIVE FLOPPY DISK "Intelligent" skivminne med microprocessor, 16 K rom och 2K ram Lagrar 165K bytes på 5 1/4 tum diskett. Sekvensiell filhantering, relativa filer. Möjligt att öppna och lägga till i i sekvensiella filer. Inbyggd nätdel.	4.995:–	4.110:–
VIC-1010, EXPANSIONSENHET Med 6 uttag för minnes- och/eller programkasetter. Ansluts direkt eller via kabel (tillbehör) till VIC-20's minnesexpansionsbus. Inbyggd nätdel.	1.650:–	1.357:–

Produkt	Cirkapris inkl. 21,51% moms	Cirkapris exkl. moms
VIC-1011, RS-232C INTERFACE Ansluts till VIC-20's "user port".	495:–	407:–
VIC-1110, 8K MINNES-KASSETT Pluggas in i VIC-1010 expansionsenhet. 8K statiskt ram.	595:–	491:–
VIC-1111, 16K MINNES-KASSETT Pluggas in i VIC-1010 expansionsenhet, 16K statiskt ram.	995:–	819:–
VIC-1210, 3K MINNES-KASSETT Pluggas in i VIC-20's minnesexpansionsbus eller i VIC-1010 expansionsenhet. 3K statiskt ram.	349:–	288:–
VIC-1211M, SUPER-EXPANDER Plugg-in kassett med 3K ram. Ansluts till VIC-20's minnesexpansinsbus eller till VIC-1010 expansionsenhet.	595:–	490:–
VIC-1212, PROGRAMMER's AID Plugg-in kassett. Ansluts till VIC-1010 expansionsenhet.	395:–	325:–
VIC-1213, MASKINSPRÅKSMONITOR Plugg-in kassett	395:–	325:–

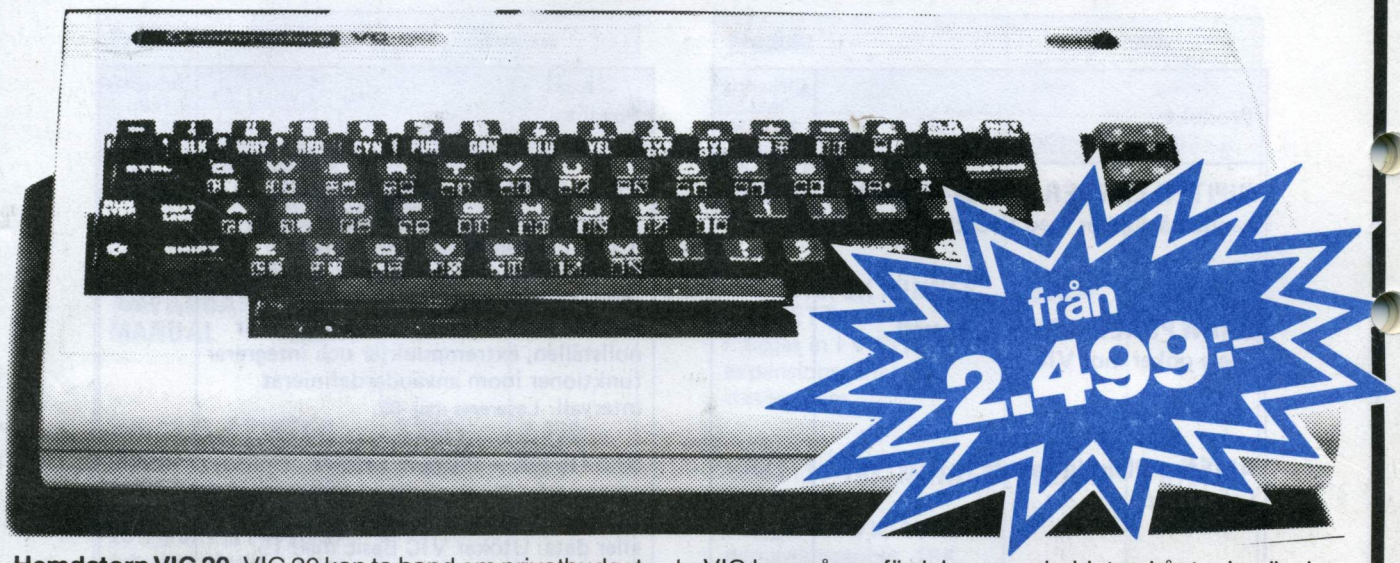
Program och tillbehör till VIC-20

Produkt	Cirkapris inkl. 21,51% moms
JUPITER LANDER – Plugg-in kassett Landa på Jupiter med VIC-20	195:-
DRAW POKER – Plugg-in kassett Spela poker mot VIC-20	195:-
ALIEN – Plugg-in kassett Labyrintspel	195:-
AVENGER – Plugg-in kassett Försvara din månbas mot anfallande rymdmonster	195:-
SLOT – Plugg-in kassett "enarmad bandit"	195:-
ROAD RACE – Plugg-in kassett Kör bil med VIC-20	195:-
VIC-PROGRAM Nr 1 – Bandkassett **Innehåller 6 program:** A. Skrivövning (maskinskrivning) B. Mattelabyrint C. Ränteberäkning D. "Elakingar" spelprogram E. Stavningsövning F. Biorytmprogram	45:-
VIC-PROGRAM Nr 2 – Bandkassett **VIC adress och VIC telefon** **Innehåller 4 program:** A. Adressregisterprogram B. Sorteringsprogram 1 C. Telefonregisterprogram D. Sorteringsprogram 2	69:-

Produkt	Cirkapris inkl. 21,51% moms
GRAFANALYS – Plugg-in kassett 3K expansionsminne inkluderat matematiskt pedagogiskt stödprogram. Plottar användardefinierad funktionsgraf (högupplösningsgrafik) med kurvanpassade koordinataxlar, samt beräknar nollställen, extrempunkter och integrerar funktioner inom användardefinierat intervall. Levereras maj -82.	495:-
STATISTIK – Plugg-in kassett Hjälpmedel för statistiska beräkningar och grafisk presentation av statistik eller data. Utökar VIC Basic med 15 kommandon. Levereras mars -82.	495:-
VIC-FORTH – Plugg-in kassett Forth är den 4:e generationens programmeringsspråk, lämpligt för den som vill ha ut lite extra av sin VIC. Språket är kompilerat, interpreterat, kompakt, mycket snabbt och innehåller macroassembler, editor och virtuellt minne. VIC-Forth kan arbeta med alla minnesstorlekar från 3K till 32K. Levereras april -82.	549:-
VIC TAPE C-12, 2-pack	19:95
VIC DISKETT, 10-pack	340:-
VIC PRINTER-PAPPER, 8x6" 10 buntar om 500 ark.	420:-
LITTERATUR	
"Lär dig VIC-20" – Användarmanual En handbok för första-gångs-användaren av datorer (svensk text).	79:-
VIC PROGRAMMERINGSHANDBOK Ger ingående information om programmering av VIC-20 (svensk text).	149:-
"BASIC PÅ VIC-20" Författare: Sune Windisch Lärobok i programmering av VIC-20 (svensk text)	89:-

Och NU Folkdatorn – VIC!

VIC 20 är den första datorn som är gjord för privatpersoner. Den är gjord för att användas i hemmet, för hobby eller för undervisning. VIC är en släkting till den välkända PET-datorn.



Hemdatorn VIC 20. VIC 20 kan ta hand om privatbudget, aktieaffärer, huskalkyler, kalkylera bilkostnader etc. Du kan lära dig programmering med VIC. Du kan lära dig hur datorer fungerar. VIC är din språngbräda in i datortekniken. Till VIC finns också många spel utvecklade. VIC ger hela familjen stort nöje och stor kunskap.

Hobbydatorn VIC 20. Du kan programmera VIC 20 i Basic och Assembler. VIC kan användas separat eller anslutas till annan utrustning för styrning, reglering, mätning etc. VIC kan kommunicera med andra datorer över telefon via ett telefonmodem. Till VIC finns ett stort utbud av tillbehör som gör att VIC kan växa i takt med dina ökade kunskaper och krav.

Undervisningsdatorn VIC 20. VIC kan användas i undervisning i programmering, datorkunskap, tekniska ämnen, fysik, kemi, biologi, matematik, geografi, samhällskunskap, språk etc.

VIC har många fördelar som skoldator. Lågt pris gör den lämplig att skaffa i klassuppsättningar. Programmerbar i BASIC – det enklaste och mest logiska programmeringsspråket, som man bör börja med, för att därefter kunna gå vidare. Stort sortiment av tillbehör gör att VIC kan byggas ut i den takt man önskar.

Faktaruta:
- 5 K RAM expanderbart till 32 K
- 20 K ROM
- BASIC och Assembler
- Färg – 24 st
- Ljud – tre tongeneratorer
- Bilden blir 22 tecken bred och 23 linjer hög
- Skrivmaskinstangentbord
- 4 programmerbara funktionstangenter med 8 möjliga funktioner
- Stort tillbehörssortiment

Det är roligt och nyttigt att skaffa en VIC 20. Gå in till närmaste handic-handlare redan idag och be honom demonstrera VIC för dig eller skicka in kupongen så får du mer information. Gör det redan idag!

VIC tillverkas av Commodore Business Machines, ett av världens ledande företag på mikrodatorer. I Commodore finns också MOS Technology, ett företag som arbetar med forskning och produktion av mikrochip. MOS ligger i den absoluta frontlinjen.
Kombinationen mellan MOS Technology och Commodore är förklaringen till att Commodore tillverkar den första folkdatorn — VIC 20.

handic electronic ab

Box 1063, 436 00 Askim / Göteborg, Tel: 031 / 28 97 90
– ett företag i Datatronicgruppen –

☐ **Skicka mig mer information om folkdatorn VIC.**

Namn _____

_____

Adress _____

_____

Postnr. / Ort _____

_____

Frankeras ej. handic betalar portot.

handic electronic ab

Svarsförsändelse
Kontonummer 2401900-2
436 00 Askim

Quality–Reliability–Safety

Tryckt på Bergslagsposten 1982.