Commodore BASIC 7.0

för Commodore 128

av Anders Hesselbom

Commodore BASIC 7.0

för Commodore 128

av Anders Hesselbom

Publicerad 2024-03-XX, revision 0, 2024-03-XX. Rättningar och kommentarer skickas till anders@winsoft.se

Innehåll

KAPITEL 1: INTRODUKTION

Introduktion

Commodore 128 är en av de mest mångsidiga och kapabla datorer som någonsin skapats. Maskinen har en avancerad BASIC (som är denna boks primära fokus), är kompatibel med Commodore 64, har två huvud-processorer som används antingen inom eller utanför diskoperativsystemet CP/M, och har en generös uppsättning av inbyggda kommandon och verktyg.

Min bok om **Commodore BASIC 2.0 second release** handlar främst om det nämnda *språket*. Om du köpte en VIC-20 eller en Commodore 64 var det den BASIC-versionen du fick inbyggd i din dator. Commodore BASIC 2.0 second release saknar kommandon för multimedia, så ska du skriva program som utnyttjar datorns kapacitet för grafik och ljud är du hänvisad till att sätta minnesadresser eller rent av välja maskinkod istället för BASIC. Den boken är neutral till ditt val av dator.

Commodore BASIC 7.0 är ett språk framtaget för just Commodore 128, och den datorn har ungefär samma multimediakapacitet som Commodore 64. Den här boken är därför knuten till en specifik dator, nämligen just Commodore 128, och vänder sig till dig som vill bemästra den datorn, främst genom att lära sig dess BASIC – Commodore BASIC 7.0 – ett språk som erbjuder avancerade kommandon för multimedia.

Den här boken ger inte någon komplett bild av Commodore BASIC 7.0, utan fokuserar på datorn Commodore 128 och nyheterna i Commodore BASIC 7.0, som introducerats sedan Commodore BASIC 2.0 second release, vilket innebär att det kan vara bra att ha det min tidigare nämnda bok som förkunskap.

För att få grunderna i Commodore BASIC, få en mer komplett terminologi och kunskap om principerna som gäller för din Commodore-dator, läs min bok om **Commodore BASIC 2.0 second release** först! Har du grundkunskaper, och vill fördjupa din kunskap om Commodore 128, är det rätt bok du håller i din hand.

Konventioner i boken

Indata som programrader eller kommandon skrivs med följande teckensnitt:

```
PRINT "HEJ"
```

Samma teckensnitt används för svaren från datorn.

Hänvisningar till tangenter på Commodore 128 skrivs med fetstil. Bilden visar till exempel **Return** till höger, **Run Stop** till vänster, och så vidare.

Figur 1: Tangentbordslayout på Commodore 128. Foto: Evan Amos

Den exakta tangentbordslayouten varierar beroende på vilken marknad du den dator du köpt är avsedd för. Bilden ovan visar en engelsk Commodore 128. För information om de olika tangenternas funktion, se din dators användarmanual.

Ibland ska du trycka ner två tangenter. Om det står till exempel **Shift+A** ska **Shift** hållas nedtryckt medan **A** trycks ner.

Bildförklaringar och kodförklaringar skrivs i *kursiv stil*, som också används för att emfasera termer eller viktiga poänger. Även namn på felmeddelanden skrivs med kursiv stil.

Commodore BASIC 7.0

Commodore BASIC 7.0 är en vidareutveckling av Commodore BASIC 2.0 second release, och innehåller ungefär samma uppsättning av kommandon som Commodore BASIC 3.6. Förutom samtliga kommandon från 2.0 och några kommandon för flödeskontroll och felsökning handlar de flesta antingen om I/O eller multimedia.

Version 3.6 togs fram till en bärbar dator, *Commodore LCD*, som aldrig nådde marknaden. Men mycket arbete som Commodore gjorde, togs med till Commodore 128, däribland BASIC, som färdigutvecklat fick versionsnumret 7.0.

Kommandot `SYS`, har utökats för att vara mer mångsidigt.

För flödeskontroll har vi fått både ett `ELSE`-kommando och några nya sätt att skapa iterationer med `DO` och `LOOP`.

För den som vill analysera eller manipulera textsträngar finns t.ex. det nya kommandot `INSTR`.

För I/O finns en rik uppsättning kommandon för att skriva och läsa data, som t.ex. `BLOAD` och `BSAVE`.

För att hantera användarinteraktioner finns kommandon för att läsa av joystick och ljuspenna.

För högupplöst grafik finns inbyggda kommandon för att rita figurer som linjer, cirklar och rektanglar.

För rörlig grafik finns en uppsättning av kommandon för att hantera sprites.

För att skapa musik finns det mycket avancerade kommandot `PLAY`, och för den som vill utveckla egna ljudeffekter finns kommandot `SOUND`.

I princip finns det kommandon för att komma åt Commodore 128:s samtliga funktioner.

Versioner

Commodore BASIC finns i de versioner som presenteras nedan.

Version 1.0 för Commodore PET 2001 som baseras på Microsoft BASIC.

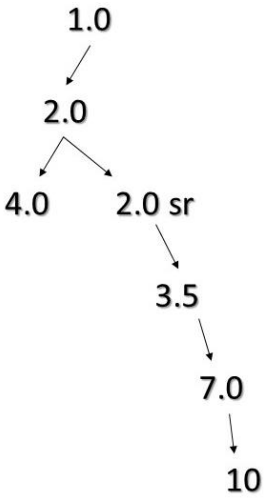
Version 2.0 för Commodore PET 2001 som är en vidareutveckling av version 1.0.

Version 4.0 för Commodore PET 4000 och CBM 8000 är den sista vidareutvecklingen av första version 2.0.

Version 2.0 second release som avhandlas i denna bok är buggrättad version av version 2.0 för PET 2001.

Version 4+ för Commodore CBM-II är en vidareutveckling på version 2.0 för PET 2001.

Version 3.5 för Commodore 16, Commodore 116 och Plus/4 är en vidareutveckling av Commodore BASIC 2.0 second release.



Figur 2: Språkets utveckling.

Version 7.0 för Commodore 128 är en vidareutveckling av version 3.5. Denna fanns även i prototypen Commodore LCD med versionsnumret 3.6.

Version 10 utvecklades för prototypen Commodore 65. Varken Commodore 65 eller Commodore LCD nådde någonsin konsumentmarknaden. Tyska *MEGA Museum of Electronic Games and Art* arbetar med att få ut en färdigställd Commodore 65-klon på marknaden.

Bokens innehåll

Den här boken innehåller, inklusive introduktionen, 13 kapitel och tre bilagor. Här följer en överblick över bokens kapitel, utöver detta första kapitel:

- Det andra kapitlet beskriver **allmänna förbättringar** i Commodore BASIC 7.0 jämfört med
- Det tredje kapitlet ger en övergripande beskrivning om datorn boken handlar om, **Commodore 128**.
- Kapitlet **Text** presenterar nya möjligheter att analysera och manipulera text.
- I kapitlet om **grafik** beskrivs hur högupplöst grafik kan skapas med Commodore BASIC 7.0.
- **Sprites** handlar om rörlig grafik och enklare animationer.
- Kapitlet om **ljud** visar hur man kan få Commodore 128 att spela upp enklare toner och effekter.
- I kapitlet om **musik** beskrivs hur melodier kan komponeras och framföras av flera röster.
- I kapitlet om **avancerade ljudeffekter** beskrivs tidigare odokumenterade funktioner för att skapa nya ljud med Commodore BASIC 7.0.
- Kapitlet om **användarinteraktion** beskriver hur man läser av tangentbordet, joysticks och ljuspennan.
- I kapitlet om **80-kolumnsläge** förklaras hur man kan dra nytta av datorns förmåga att dubblera antalet tecken som visas på skärmen.
- Därefter beskrivs de utökade möjligheterna att bevara data på disk i kapitlet om **DOS**.
- **Commodore 64-läget** beskrivs i det tolfte kapitlet.
- Det trettionde och sista kapitlet ger en introduktion till **CP/M**.

Här följer en beskrivning av bokens fyra bilagor, kallade *appendix A, B, C* och *D*:

- **Appendix A** handlar om felsökning (debugging).
- **Appendix B** förklarar de tekniska termer som används i boken.
- **Appendix C** jämför Commodore 128 med föregångarna Commodore 64 och VIC-20.
- **Appendix D** ger en introduktion till maskinkod.

KAPITEL 2: OM COMMODORE 128

Om Commodore 128

Commodore 128 introducerades på marknaden år 1985, och såldes fram till och med år 1989, när 16-bitarssystemen som Atari ST och Amiga började vinna mark. Som namnet indikerar har datorn 128 kilobyte (KB) RAM (som kunde utökas till 640 KB) vilket räcker ganska långt för många olika typer av program, men det är kanske lite i underkant för mer avancerade animationer och avancerad multimedia.

Datorn har två huvudprocessorer. MOS 8502 klarar samma instruktioner som de processorer som satt i bl.a. Commodore 64 (MOS 6510 eller MOS 8500) och VIC-20 (MOS 6502). Det är denna som normalt driver runt din Commodore 128 med en arbetshastighet på 1-2 megahertz (MHz). Den andra processorn är en Zilog Z80 på 4 MHz. Det är denna som driver runt din dator när du arbetar i CP/M-läge.

Det finns ett antal olika operativsystem för Commodore 128. Datorn kan köras med eller utan operativsystemet CP/M. CP/M (*Control Program for Microcomputers*) behöver läsas in från diskett (att ”boota” operativsystemet). Datorn levererades med version CP/M Plus version 3.0, och ger tillgång till avancerad mjukvara som t.ex. Turbo Pascal eller Microsoft Basic.

För den som inte vill köra operativsystem som likt CP/M styrs med textkommandon, finns möjligheten att köpa till det grafiska operativsystemet GEOS som kontrolleras med mus. GEOS (Graphic Environment Operating System) använder sig av rullgardinsmenyer, fönster och ikoner för att låta användaren kontrollera datorn.

Commodore 128 har samma ljudkapacitet som Commodore 64, som drivs av MOS-chippet 6581 (8580 i senare modeller). Dessa kallas kort och för SID (Sound Interface Device).

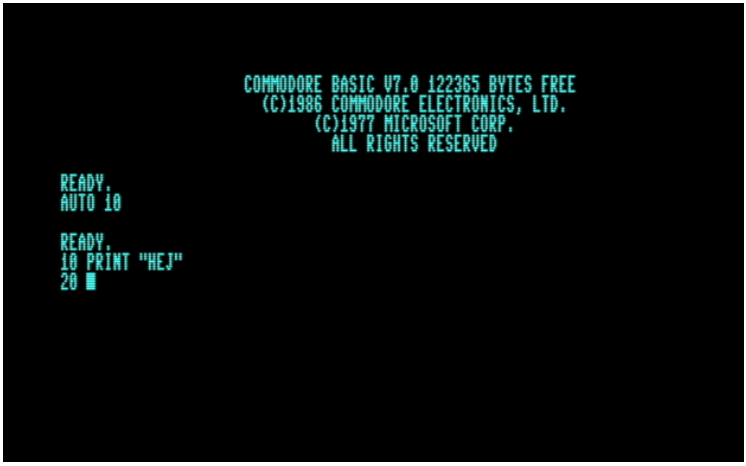
För grafik har Commodore 128 en VIC-II E med ungefär samma kapacitet som VIC-II som satt i Commodore 64, vilket innebär en skärmupplösning på 320×200 punkter (pixlar), 16 färger, 8 sprites och raster. Förutom detta har version E även stöd för blitter och ett extra grafikläge med en skärmupplösning på 640×200 punkter.

Om inget operativsystem startas, används själva Commodore BASIC 7.0 för att kontrollera datorn genom BASIC-kommandon.

För att programmera Commodore 128 används företrädelsevis just Commodore BASIC, som i utförande 7.0 är väldigt kraftfullt. Den stora nackdelen med BASIC är dess undermåliga prestanda. BASIC kan göra allt du önskar men om prestanda är en faktor måste du titta på något annat, t.ex. maskinkod. Maskinkod presenteras överskådligt i appendix C.

Inmatning av BASIC-program

Språket har två lägen. Det ena kallas *direkt*, och innebär att man skriver en instruktion utan radnummer, som exekveras direkt när man trycker på **Return**. Det andra kallas *runtime*. Instruktioner som får ett radnummer, exekveras i runtime, alltså när programmet körs med (normalt) RUN. Om inget annat anges, kan alla kommandon användas både i direktläge och i runtime-läge.För den intresserade bjuds en hel del trevliga tricks, som t.ex. `AUTO`, som visas på bilden.



Figur 3: AUTO erbjuder automatisk inskrivning av radnummer.

Kommandot `AUTO` tar ett argument, och det är avståndet från nuvarande rad till nästa. Genom att skriva `AUTO 10` så säger man till datorn att nästa radnummer ska vara nuvarande plus 10, vilket gör att datorn föreslår 20 efter att ett kommando matats in på rad 10, och så vidare.

För att stänga av automatiskt förslag på radnummer, skriv `AUTO` utan några parametrar och tryck **Return**.

Om du vill infoga ett kommando mellan, säg, rad `10` och rad `20`, kan du kalla den nya raden för `15`. Genom att skriva in dessa tre rader…

```
10 PRINT "A"
20 PRINT "B"
15 PRINT "C"
```

…så får du följande program, som kan visas med `LIST`:

```
10 PRINT "A"
15 PRINT "C"
20 PRINT "B"
```

Om du infogar tillräckligt många rader mellan rad `10` och `20`, kommer utrymmet att ta slut. Kommandot `RENUMBER` justerar avståndet mellan existerande rader så att den första raden börjar på `10` och att avståndet mellan alla rader. Exempel:

```
RENUMBER

READY.
LIST

10 PRINT "A"
20 PRINT "C"
30 PRINT "B"
```

Det finns även en möjlighet att själv ange startnummer och steg. Genom att anropa `RENUMBER 3,2` kommer programmet att starta på rad `3` och efterföljande rader kommer med ett avstånd på `2`.

```
RENUMBER 3,2

READY.
LIST

3 PRINT "A"
5 PRINT "C"
7 PRINT "B"
```

Du kan även ange `RENUMBER` endast ska agera på rader som är lika med eller större än ett visst befintligt radnummer genom att komplettera med ett startnummer. `RENUMBER 1000,10,5` programmet starta på rad `100` och efterföljande rad kommer på ett avstånd på `2`, men det gäller bara raderna som nu är större än eller lika med `5` (alltså inte rad `3`).

```
RENUMBER 1000,10,5

READY.
LIST

3 PRINT "A"
1000 PRINT "C"
1010 PRINT "B"
```

Om du använder `RENUMBER` i ett program så uppstår felet *direct mode only*, och `RENUMBER` får inte användas när programmet har en `GOTO` eller `GOSUB` som inte pekar på en existerande rad, då uppstår felet *unresolved reference*. Om programmet slår över taket (högsta tillåtna radnummer är `63999`) uppstår felet *line number too large*.

En liten varning! I Commodore BASIC definieras subrutiner av ett radnummer, vilket innebär att du behöver kontrollera vilka nya radnummer dina subrutiner har fått efter att du använt `RENUMBER`. Annars kan det bli svårt att anropa dessa på nytt.

Datatyper

Commodore BASIC 7.0 har stöd för tre datatyper. Dessa är *realtal*, *heltal* och *strängar*. Realtal använder punkt som decimalavgränsare och strängar omges av citattecken. Variablernas typ deklareras med ett postfix på variabelnamnet, där `$` (dollartecken) avser sträng och `%` (procenttecken) avser heltal. Realtal är variabler som saknar ett avslutande tecken. Apropå variabelnamn så identifieras variabler endast av de två första bokstäverna i sitt namn. En mer ingående beskrivning av datatyper och variabelnamn finns i boken *Commodore BASIC 2.0 second release*[1].

[1] Finns att ladda hem i PDF-format eller EPUB-format här: https://ahesselbom.se/pages/commodorebasic20.html

Minneshantering

Commodore 128 arbetar med s.k. minnesbanker. En minnesbank är en fördefinierad minneskonfiguration, och du bestämmer vilken minnesbank som är tillgänglig för processorn genom att använda kommandot `BANK`. Som argument tar `BANK` ett tal mellan `0` och `15`. Exempel:

```
BANK 4
```

Du kan hoppa mellan nio olika banker. Dessa har nummer `0`, `1`, `4`, `5`, `8`, `9`, `12`, `13`, `14` och `15`. `2` är samma som `0`, `3` är samma som `1`, `6` är samma som `4`, `7` är samma som `6`, `10` är samma som `8` och `11` är samma som `9`. Val av minnesbank påverkar kommandon som använder minnet direkt. Dessa kommandon är `SYS`, `PEEK`, `POKE` och `WAIT`. Minnesbank `0` är den som är förvald.

Vill du se en konsekvens av att växla mellan minnesbanker, kan du köra ett enkelt testprogram[2].

```
10 POKE 4096,75
20 PRINT PEEK(4096)
30 BANK 1
40 POKE 4096,90
50 PRINT PEEK(4096)
60 BANK 0
70 PRINT PEEK(4096)
```

Rad 10 skriver värdet 75 i aktuell minnesbank (vilket är 0 eller 3 om inget annat har sagts).
Rad 20 konstaterar det skrivna värdet.
Rad 30 växlar minnesbank till 1.
Rad 40 skriver 90 till samma adress, fast i bank 1 istället för bank 0 eller 3.
Rad 50 konstaterar det skrivna värdet.
Rad 60-70 växlar tillbaka till bank 0 (eller 3) och konstaterar att adressen fortfarande innehåller värdet 75.

Utöver detta, kommandot `STASH` och kommandot `FETCH` används för att kopiera data mellan minnesbanker och `SWAP` används för att låta två minnesbanker byta data mellan varandra.

[2] Kom ihåg att använda kommandot NEW för att radera eventuellt befintligt BASIC-program ur minnet, innan du skriver in ett nytt program.

KAPITEL 3: ALLMÄNNA FÖRBÄTTRINGAR I BASIC

Allmänna förbättringar i BASIC

Den stora skillnaden mellan Commodore BASIC 2.0 second release (programmeringsspråket som Commodore 64 och VIC-20 är utrustad med) och Commodore BASIC 7.0 (som är inbyggt i din Commodore 128) handlar om DOS och om multimedia. Men det finns några allmänna förbättringar i version 7.0 som verkligen underlättar arbetet för den som vill bygga mjukvara.

Tester

En viktig förändring i hur tester fungerar är att ett eller flera kommandon kan exekveras om ett uttryck är sant. Nyckelordet `ELSE`, som används tillsammans med `IF`, åstadkommer detta.

Låt säga att du vill att två saker ska hända om `A` är lika med `5`, men två andra saker om `A` inte är lika med `5`. På en Commodore 64 skulle detta kunna se ut så här:

```
630 IF X=5 THEN PRINT "A":PRINT "B"
640 IF X<>5 THEN PRINT "C":PRINT "D"
```

Exemplet ovan är hypotetiskt och utgör inte ett komplett program. De två ting som utförs om `X` har värdet `5` är förmodligen något annat än `PRINT "A"` och `PRINT "B"` i ett realistiskt exempel.

Det finns två problem med ovanstående kod, båda beror på att det falska uttrycket på rad `640` (i förhållande till det på rad `630`) måste formuleras uttryckligen. Dels ökar det risken för buggar, eftersom vi har två uttryck som egentligen beskriver ett uttryck och dess motsats. Och dels måste uttrycket `X=5` utvärderas två gånger – först som det är skrivet och därefter i sin motsats.

Genom att istället använda `ELSE` så löser man båda dessa problem. `ELSE` antar att om det första uttrycket är falskt, är det koden som står efter `ELSE` som ska köras. Det innebär att `X=5` endast behöver utvärderas en gång, och att det inverterade uttrycket inte behöver tillhandahållas av programmeraren. Denna programsats gör samma sak som exemplet ovan (radbytet ska inte vara med):

```
630 IF X=5 THEN PRINT "A":PRINT "B":
    ELSE PRINT "C":PRINT "D"
```

Notera att `ELSE` kräver ett kolon (`:`) före, men inte efter.

Iterationer

Prestanda

Commodore 128 är tillräckligt bra för att driva runt någorlunda avancerade spel och andra typer av datorprogram. En väldigt stor del av datorns funktionalitet görs tillgänglig för användaren genom BASIC, men BASIC är ytterst långsamt. Detta märks främst när man programmerar datorspel.

Men om en tidsödslande beräkning ska göras, så erbjuder BASIC en lösning: Man kan stänga av bilduppdateringen och därmed använda mer av processorns uppmärksamhet till att lösa en uppgift. Nackdelen är att bildskärmen är blank under tiden beräkningen görs. Kommandot `FAST` läcker skärmen och ökar prestandan i BASIC, och kommandot `SLOW` återgår till normalläge.

Följande program exekverar på 482 jiffies:

```
10 T=TI
20 SCNCLR
30 A=150000
40 A=A+1:A=A*2
50 A=A/3:A=A+A/4
60 PRINT A
70 IF A>5.01 THEN 40
80 PRINT TI-T
```

Om vi kompletterar programmet så att första raden släcker skärmen…

```
5 FAST
```

…och sista raden tänder skärmen…

```
90 SLOW
```

…så blir exekveringstiden istället 236 jiffies, vilket innebär en förbättring på ungefär 50%.

Övrigt

För att pausa ett program används `SLEEP` följt av ett önskat antal sekunder (angivet som heltal). Det innebär att kommandot `SLEEP 60` pausar datorn

under en hel minut. `SLEEP 0` pausar till nästa interrupt, vilken aldrig är längre än 16,7 miilisekunder bort. Tidigare nämnda `FAST` och `SLOW` påverkar inte beteendet för `SLEEP`.

KAPITEL 4: TEXT

Text

XXX

Text till tal

En textsträng innehållande ett decimaltal (alltså ett tal som beskrivs av tecknen `0` till och med `9`) konverteras till ett tal med funktionen `VAL`. Detta enkla program skriver värdet 40 på skärmen.

```
10 X=VAL("20")
20 PRINT X*2
```

Rad 10 konverterar en sträng innehållande en tvåa och en nolla till talet 20.
Rad 20 multiplicerar resultatet med 2 och skriver ut det (40) på skärmen.

Exemplet visar hur en sträng innehållande ett tal kan konverteras till ett riktigt tal och användas i numeriska beräkningar. Detta är ganska standard inom BASIC, och har varit med länge i Commodore BASIC.

Utöver VAL kan Commodore 128 även jobba med strängrepresentationer av det hexadecimala talsystemet. TODO DEC

Formatera ett tal

TODO PUDEF

KAPITEL 5: GRAFIK

Grafik

XXX

KAPITEL 6: SPRITES

Sprites

KAPITEL 7: LJUD

Ljud

XXX

KAPITEL 8: MUSIK

Musik

XXX

KAPITEL 9: AVANCERADE LJUDEFFEKTER

Avancerade ljudeffekter

XXX

KAPITEL 10: ANVÄNDARINTERAKTION

Användarinteraktion

XXX

KAPITEL 11: 80-KOLUMNSLÄGE

80-kolumnsläge

XXX

KAPITEL 12: COMMODORE BASIC 7.0 DOS

Commodore BASIC 7.0 DOS

XXXX

KAPITEL 13: COMMODORE 64-LÄGE

Commodore 64-läge

Din manual beskriver hur du startar din Commodore 128 i olika lägen, men du kan växla från standardläget till Commodore 64-läget med kommandot `GO64`. I direktläge frågar datorn om du verkligen vill växla till Commodore 64-läget (ditt BASIC-program går förlorat), men om `GO64` används i ett program, växlar datorn läge direkt, utan varning. Det går att infoga ett blanksteg mellan `GO` och `64`, och som kuriosa kan det vara bra att veta att `64` kan ersättas med ett uttryck som utvärderas till just `64`. Om du t.ex. tilldelar variabeln `AT` värdet `64`, går det bra att växla till 64-läge genom att skriva `GOAT`, eller om du tilldelar `HO` värdet `64`, kan du växla till 64-läge genom att skriva `GO HOME`.

KAPITEL 14: CP/M

CP/M

XXX

APPENDIX A: FELSÖKNING

Appendix A: Felsökning

Commodore 128 innehåller en del avancerade funktioner som underlättar felsökningen av ett program (debugging). Bland dessa hittar vi `EL`. Detta kapitel beskriver funktionerna och hur de kan vara till nytta.

EL

Så snart ett programfel uppstår laddas systemvariabeln `EL` med det senaste radnumret där felet uppstod. Följande program kommer att orsaka ett fel på rad `20`, eftersom Commodore 128 inte tillåter division med 0.

```
10 PRINT "EN DIVISION"
20 PRINT 10/0
```

När detta program startats med RUN ger datorn följande svar:

```
EN DIVISION

?DIVISION BY ZERO ERROR IN 20
READY
```

Om du nu läser av EL får du svaret 20, eftersom felet inträffade på svar 20.

```
PRINT EL
 20

READY.
```

Om du läser av `EL` innan något fel har inträffat, får du svaret `65535`. Du vet att inget fel har uppstått, eftersom ett giltigt radnummer på Commodore 128 är ett tal mellan `0` och `63999`. Fel som inträffar i direktläge ändrar inte värdet i `EL`. Om rättar ett fel i ett program så att inget fel uppstår, och därefter kör programmet så återställs värdet av EL till `65535`.

APPENDIX B: ORDFÖRKLARINGAR

Appendix B: Ordförklaringar

Adress

Den minsta allokeringsbara enheten i en 8-bitarsdator är en byte, varje byte har en *adress* (eller *minnesadress*). Vissa är läs- och skrivbara (RAM), andra är endast läsbara (ROM).

Användardefinierad variabel

En användardefinierad variabel är en variabel skapad av datoranvändaren. Här placeras värdet `23` i variabeln `AW`:

```
AW=23
```

En användardefinierad variabel är inte en systemvariabel.

Argument

Data som skickas till (till exempel) en funktion för att påverka dess funktionalitet. Ett argument är samma sak som en *parameter*.

Array

En array är en samling av variabler som identifieras av ett index. I BASIC har arrayen själv ett namn som följer reglerna för variabelnamn, där varje element identifieras av ett 0-baserat index som anges inom parentes, till exempel `A(31)`. Ordet *vektor* kan användas synonymt med array.

Binär logik

Binär logik är logik som är tvåställig i betydelsen att ett uttryck antingen är sant eller falskt, precis som en bit.

Binära talsystemet

Det binära talsystemet använder endast två tecken för att beskriva ett tal, till skillnad från det decimala talsystemet som använder tio eller det hexadecimala som använder sexton. Det innebär att de fem första talen (noll till fyra) skrivs 0, 1, 10, 11 och 100.

Bit

En bit är ett tal mellan 0 och 1. Åtta bitar utgör en byte. Det binära talsystemet utgörs av bitar, från engelskans "binary digits".

Bitvis

En bitvis operator agerar på värdets bitmönster, bestående av `0` eller `1`. och kan därför användas i binär aritmetik. Som exempel är `2 OR 4` lika med `6`, eftersom `00000010 OR 00000100` är lika med `00000110`. De binära bitvisa operatorerna är `AND` och `OR`, den unära är `NOT`.

Bitmaskning

Med bitmaskning avses metoder att läsa av individuella bitar i en byte, eller skriva till en eller flera bitar utan att påverka andra bitar. Bitmaskning måste behärskas av den som vill kunna sätta eller läsa av flaggor, som t.ex. vilken sprite som ska vara synlig.

Blitter

Ett chip som hanterar blitter ansvarar för snabb manipulering av data i RAM, vilket möjliggör rörlig datorgrafik.

Booleskt värde

Ett booleskt värde är ett värde som antingen är sant eller falskt. I Commodore BASIC 2.0 second release representeras 0 som falskt och icke-0 som sant vid test. Som testresultat är 0 falskt och -1 sant. Vid bit-operationer är 0 falskt och 1 sant.

Border

Ramen runt den yta som VIC-20 och Commodore 64 kan visa text och högupplöst grafik på benämns som *border*. Borderfärgen är ramens färg, medan bakgrundsfärgen avser färgen på ytan med text och grafik.

Byte

En byte är den minsta enheten med en egen adress. En byte består av åtta bitar (eller två nibbles), vilket innebär att en byte kan befinna sig i ett av 256 olika tillstånd, till exempel ett tal mellan 0 och 255.

Call stack

Varje hopp som görs med `GOSUB` lagras i datorns *call stack*, som är ett slags lista över vilka rader som anropats med `GOSUB`. När kommandot `RETURN` påträffas, plockas ett hopp från datorns call stack, och exekveringen fortsätter på raden efter. Både VIC-20 och Commodore 64 har avsatt 256 bytes för sin call stack, och klarar att hålla programhopp i 23 led i minnet.

Datatyp

Din Commodore-dator kan endast lagra ettor och nollor, vilket representerar talet `0` eller talet `1`. Dessa kallas bitar och är alltid grupperade om 8. Åtta bitar utgör en byte, och antalet tal som kan beskrivas med åtta bitar (en byte) är 256 (0-255). De individuella bitarna saknar minnesadress, men varje byte (åtta bitar) har var sin minnesadress. Så snart man vill läsa något annat än individuella tal mellan 0 och 255 (åtta bitar) måste man veta hur olika bytes ska kombineras. En datatyp är en konfiguration av kombinationer av bytes. till exempel består talet `16961` av en kombination av två bytes, nämligen `65` och `66` (66 * 256 + 65 = 16961), medan textsträngen `AB` också består av samma kombination av bytes. Om man vet vilken datatyp man hanterar, kan man avkoda informationen korrekt.

DOS

DOS står för *Disk Operating System* och med DOS avser operativsystem kapabla att serva användaren i användandet av skivminnen som sekundärminne. Commodore-maskinerna erbjuder BASIC-tolken som användarens gränssnitt för att använda skivminnen (floppydisk), och den diskdrive som är inkopplad till enheten, erbjuder ytterligare kommandon som kan skickas till enheten för utförande. Exakt vilka kommandon som står till förfogande beror på vilken modell av diskdrive som är inkopplad till datorn. Några modeller som kan förekomma är 1540, 1541 som normalt såldes till VIC-20 och Commodore 64, 1541 II, den externa 1571 och den interna 1571 som normalt såldes till Commodore 128 respektive 128D. Denna bok tar avstamp i modell 1541.

Fysisk fil

En *fysisk fil* är, till skillnad från en *logisk fil*, en faktisk fil som finns lagrad på floppydisk eller på kassettband. En fysisk fil har ett namn och en startadress eller en (förhoppningsvis) specifik struktur.

Grafikläge

En Commodore-maskin har olika grafiklägen som anger vad som kan visas på skärmen. I textläget (som används vid uppstart) kan VIC-20 visa text i 22 rader med 23 kolumner eller 25 rader och 40 kolumner för Commodore 64. I bitmapsläge hanterar man i stället individuella pixlar i två olika upplösningar, där flerfärgsläget har hälften så många individuella pixlar på skärmen som är möjligt i enfärgsläget (eller det högupplösta läget).

- Commodore 64, enfärgsläget: 320×200 pixlar

- Commodore 64, flerfärgsläget: 160×200 pixlar
- VIC-20, enfärgsläget: 176×184 pixlar
- VIC-20, flerfärgsläget: 88×184 pixlar

Hard reset

En *hard reset* innebär att datorn återställs och att hela datorns minne (BASIC-program och övrig data) raderas. Jämför med *soft reset*.

Hexadecimala talsystemet

Det hexadecimala talsystemet använder hela sexton olika tecken för att beskriva ett tal, till skillnad från det decimala talsystemet som använder tio eller det binära talsystemet som använder två. Det innebär att de tjugo första talen (noll till nitton) skrivs 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12 och 13.

Interrupten

Interrupten är en funktion för multitasking som finns inbyggd i VIC-20 och Commodore 64. Huvudprocessorn kan, med jämna mellanrum, släppa in andra uppgifter, vilket liknar det som idag kallas *time slicing*.

I/O

I/O är en förkortning av input/output och avser operationer som läser eller skriver från/till externa enheter som till exempel tangentbord, printer, skärm eller floppydisk.

Jiffy

Vad en *jiffy* (flera *jiffies*) är varierar från plattform till plattform, men i Commodore BASIC 2.0 second release är en jiffy en sextionsdels sekund. En jiffy är alltså 1/60 sekunder och 60 jiffies är en sekund. En mer samtida godtycklig tidsenhet är en *tick*, som ofta är kortare, till exempel en tusendels sekund.

Kilobyte

En *kilobyte* (kort *K*) är 1024 bytes. 16 K betyder således 16 kilobytes eller 16384 bytes.

Konsol

Konsolen (textkonsolen) är gränssnittet mellan användaren och datorn som användaren använder genom datorns tangentbord. Man känner igen konsolen på den blinkande markören.

Loader

En *loader* är ett program som laddar in ett program i minnet. I Commodore-världen är detta typiska exempel:

- Ett dataspel som ska laddas från kassett. Första programmet har till uppgift att visa en pausbild på skärmen, medan det hämtar in huvudprogrammet från kassettbandet.
- Ett BASIC-program som skriver ett maskinkodsprogram till minnet, som visas i avsnittet om `DATA`.

Logisk fil

Normalt är en fil ett stycke namngivet data på floppydisk, men en *logisk fil* kan vara precis vad som helst som kan läsas från eller skrivas till *som om* det vore en fil på en floppydisk. Förutom olika typer av filer, kan en logisk fil vara en kanal till en skrivare.

Markör

Textmarkören anger var nästa tecken från tangentbordet kommer att hamna. Markören illustreras som en blinkande rektangel.

Multitasking

Multitasking innebär att två program körs samtidigt. Det kan till exempel vara ett maskinkodsprogram som körs samtidigt som ett BASIC-program. När man programmerar en Commodore-maskin pratar man om *interrupten*, som lite slarvigt är en svensk bestämd form av engelskans *interrupt*, vilket i sammanhanget närmast kan liknas med *att bryta in*. I enkelhet kan man tänka att datorn är utrustad timers som med jämna mellanrum kan exekvera din maskinkod. Det innebär att flera maskinkodsrutiner, och även någon BASIC-rutin, kan exekvera simultant. Detta sker på bekostnad av prestandan.

Operand

En *operand* är ett värde i en ekvation. I följande exempel är `1` och `2` operander, medan `+` är *operator*:

```
X = 1 + 2
```

Operator

En *operator* är symbol eller en funktion i en matematisk operation. I följande exempel är `+` operator medan `1` och `2` är *operander*:

```
X = 1 + 2
```

PAL

PAL, Phase Alternate Line, är en standard för analog färg-tv som primärt används i Europa. I Amerika används NTSC och i Asien används SECAM.

Parameter

Data som skickas till (till exempel) en funktion för att påverka dess funktionalitet. En parameter är samma sak som ett *argument*.

Pekare

Pekare är inget som BASIC har stöd för, men man kommer ofta i kontakt med pekare när man använder datorns inbyggda funktioner. En minnesadress som syftar till att hålla reda på en minnesadress är en pekare. Olika pekare används på olika sätt. Ett par exempel som specifikt gäller Commodore 64: Adress `785`-`786` ett 16-bitarstal som är den exakta adressen för funktionen USR, medan på adress `2047` talar om var i minnet bilddata för den åttonde spriten på ligger, om man multiplicerar värdet med 64.

PETSCII

PETSCII (PET Standard Code of Information Interchange) är namnet på teckentabellen som används av bland andra VIC-20 och Commodore 64. PETSCII är Commodores version av ASCII, och teckentabellen är uppkallad efter den första datorn som använde den, Commodore PET (år 1977).

Pixel

Bilden som visas på skärmen består av punkter av olika färger, i en matris. Varje punkt (egentligen *bildelement*) som utgör bilden, kallas för *pixel*.

Primärminne

Primärminnet är datorns arbetsminne (datorns RAM-minne).

Pseudografik

Pseudografik är textbaserad grafik som syftar till att se ut som högupplöst rastergrafik. På din Commodore-dator finns massor av tecken framtagna för att kunna passa till pseudografik. Se kapitlet om text för mer information.

Radnummer

BASIC-program på Commodore 128 använder tal för att hålla reda på vilket kommando som ska utföras i vilken ordning, och framför allt, vilka kommandon som ingår i ett program, genom att numrera programsatserna.

RAM

RAM står för *random access memory* och ordet avser minne som man kan skriva till eller läsa av i valfri ordning – man kan ange värdets minnesadress. I boken menas *datorns primärminne* eller *arbetsminne*.

Ren funktion

En *ren funktion* agerar endast på inkommande data och ger ett svar utifrån det, utan att läsa från annat minne eller skriva till ett annat minne. Ett sådant exempel är `SQR` som ger roten av ett tal (beskrivs i boken Commodore BASIC 2.0 second release) eller funktionen `ERR$` som beskrivs appendix A om felsökning.

Reset

En *reset* återställer datorn, vilket innebär att BASIC-program går förlorat. Se *hard reset* och *soft reset*.

ROM

ROM står för *read only memory* (minne som endast kan läsas, inte skrivas) och avser datorminne som beskriver datorns inbyggda funktioner.

Sekundärminne

Sekundärminnet används, till skillnad från primärminnet, till långvarig lagring. Commodore använder antingen floppydisk eller datasette som sekundärminne.

Sekventiell fil

Definitionsmässigt utgörs en sekventiell datafil innehåll av data i den ordningen den skrevs till filen. En konsekvens av detta är att filen måste läsas samma ordning som den skrevs. Detta är utmärkande för programfiler, textfiler och till exempel bildfiler.

Soft reset

En *soft reset* innebär att datorn återställs och att datorns BASIC-minne raderas. Jämför med *hard reset*.

Sprite

En sprite är en liten grafisk symbol som kan röra sig fritt på skärmen, utan att störa annan grafik. Commodore 64 kan visa åtta sprites men VIC-20 saknar sprites – vill man ha tillgång till sprites på VIC-20 så får man bygga en egen sprite-rutin. Se appendix E, som jämför VIC-20, Commodore 64 och Commodore 128 med varandra för mer information.

Systemvariabel

En systemvariabel är en variabel som får sitt värde från systemet snarare än från datoranvändaren. En systemvariabel är inte en *användardefinierad* variabel. Vissa systemvariabler kan initieras eller uppdateras av användaren, andra systemvariabler är helt skrivskyddade, men alla uppdateras av systemet.

Vektor

I BASIC används ordet *vektor* synonymt med ordet *array*. En array (eller en vektor) har ett antal element och varje element har ett värde.

APPENDIX C: EN JÄMFÖRELSE MELLAN COMMODORE 128, COMMODORE 64 OCH VIC-20

Appendix C: En jämförelse mellan Commodore 128, Commodore 64 och VIC-20

Tabellen nedan visar specifikationerna för VIC-20, Commodore 64 (C64) och Commodore 128 (C128).

	VIC-20	C64	C128
Lanseringsår	1980	1982	1985
Programmeringsspråk	Commodore BASIC 2.0 second release	Commodore BASIC 2.0 second release	Commodore BASIC 7.0
Längsta programsats	88 tecken (4 rader)	80 tecken (2 rader)	160 tecken (4 rader i 40-kolumnsläge, 2 rader i 80-kolumnsläge)
Operativsystem[3]	-	GEOS (tillval)	GEOS (tillval), CP/M 3.0
Huvudprocessor	MOS 6502	MOS 6510	MOS 8502, Z80B
Klockfrekvens[4]	1,1 MHz	0,99 MHz	1-2 MHz, 4 MHz
ROM	20 KB	20 KB	72 KB
RAM	5 KB	64 KB	128 KB
Expansionsmöjlighet RAM	32 KB[5]	320 KB	512 KB
Text	22×23	40×25	40×25, 80×25
Skärmupplösning	176×184 plus border	320×200 plus border	320×200 plus border, 640×200
Monokrom grafik	176×184	320×200	320×200
Flerfärgsgrafik	88×184	160×200	160×200
Videoutgång	Analog (A/V)	Analog (RF, A/V)	Analog (RF, A/V)/Digital (RGBI)
Ljud	Tre fyrkantsvågor och ett brusljud, tre kanaler	Konfigurerbar fyrkantsvåg, triangelvåg, sinusvåg, brus, filter, med mera. Tre kanaler	Konfigurerbar fyrkantsvåg, triangelvåg, sinusvåg, brus, filter, med mera. Tre kanaler
Maskinkodsmonitor	-	-	Ja
Sprites	-	8	8
Sprite-editor	-	-	Ja

[3] Utöver BASIC.

[4] Processorns hastighet är aningen olika för PAL-anpassade datorer och NTSC-anpassade datorer. För Commodore 128 gäller 4 MHz när Z80B-processorn används.

[5] Bryter kompatibiliteten.

APPENDIX D: MASKINKOD

Appendix D: Maskinkod

XXX

INDEX

Index

BILDER

Bilder

Erkännanden

Omslagsbilden föreställande en Commodore 128 är fotograferad av Evan Amos (CC BY-SA 3.0). Bilden används även för att visa datorns tangentbordslayout på sida 4.

Böcker i denna serie

Commodore BASIC 2.0 second release (2021)
https://ahesselbom.se/pages/commodorebasic20.html

Commodore BASIC 7.0 för Commodore 128 (2024)
https://ahesselbom.se/pages/commodorebasic70.html