

Eterm 6.1

Integrerat programutvecklingsverktyg Handbok

MC68000, MC68008, MC68010, MC68020, MC68030

MC68040, MC68881, MC68882, MC68851, CPU 32

är TM, *Motorola INC*

MS Windows TM, *Microsoft Corporation*

Dokument: Eterm 6.1 - Handbok

Id. nummer: 130-09

Denna handbok utgör del av dokumentationen av programvara *Eterm 6.1*. Såväl programvaran, som denna dokumentation, har noga kontrollerats med avseende på korrekthet. Allt bruk av såväl programvaran som denna dokumentation sker dock på användarens egen risk. **GMV** kan inte hållas ansvarigt för något som uppkommit direkt eller indirekt som konsekvens utav användning av programvaran eller den tillhörande dokumentationen.

©**GMV**, 1994-1999, Alla rättigheter förbehållna

"Eterm 6.1" är ett integrerat programutvecklingsverktyg för mikroprocessorer och microcontrollers från Motorola. Med texteditorn kan flera filer behandlas samtidigt i olika fönster. Texteditorn kan också enkelt konfigureras för "färgad syntax" för respektive målprocessor vilket avsevärt förenklar programutvecklingen. Den inbyggda terminalemulatorn medger att flera måldatorer ansluts samtidigt via flera samtidiga terminalfönster. Assemblern accepterar så kallad Motorola-syntax. Simulatorerna efterliknar microlf's enkortsdatorer.

Innehåll

Översikt.....	3
Installation	3
Funktioner.....	6
Redigera Källtexten.....	7
Assemblera källtexten... ..	9
Listfiler	9
Att använda terminalen.....	10
Kommunikationsinställningar	12
Simulatorn (MC68 och MD68k).....	13
Simulatorn (6809)	17
Simulatorn (MC11).....	21
Simulatorn (MC08).....	24
Underhåll av ETERM 6.1	27

Översikt

Välkommen som användare av *Eterm 6.1*, programmet som utformats som ett integrerat programutvecklingsverktyg i första hand avsett för undervisningsändamål. *Eterm 6.1* finns i olika versioner för följande mikroprocessorer/microcontrollers:

- Motorola MC 68000
- Motorola MC 68340
- Motorola MC 6809
- Motorola MC 68 HC 11
- Motorola MC 68 HC 08

Speciellt har simulatorfunktionerna i *Eterm 6.1* utformats för att efterlikna enkordatorerna *MD68k*, *MC68*, *MD09*, *MC11* och *MC08* från *microlf*.

I *Eterm 6.1* integreras följande funktioner:

- Textredigeringsverktyg, du kan arbeta med flera källtextfiler samtidigt
- Korsassembler för den Motorolabaserade mikrodatoren
- Simulator för den aktuella enkortsdatoren.

I denna handbok finner du det du behöver för att komma i gång med *Eterm 6.1*. Utöver detta häfte vägleds du med hjälp av "On-Line"-dokumentationen i respektive programversion, du använder dessa hjälp-funktioner på vanligt sätt under Windows.

Installation

Installationen av *Eterm 6.1* följer en standardprocedur.

Eterm 6.1

Placera distributionsdisketten i någon flexskiveenhet. Välj "**Start | Run X:setup.exe**" där X anger din flexskivestation.

Från diskett...

eller

Dubbelklicka på ikonen **e61_xxxx_dist.exe** för att starta det självuppackande installationsprogrammet.

Från Internet-distribution

Följ därefter installationsprogrammets anvisningar. Då installationen avslutats finner du programgruppen "GMV" under Start-Program-menyn. Alla *Eterm 6.1* versioner samlas i denna grupp av installationsprogrammet.

Då du startar *Eterm 6.1* första gången efter installationen möts du av följande dialogbox (gäller EJ 6809-varianten):

ETERM 6.1 License information

New Installation

Fill in Your user name and the software-ID (found on the media package)
Then press the OK button to create a key-file.

User/Company Name:

Software ID: - -

Upgrade

Press the 'Upgrade' button if You are upgrading a previous installation

Evaluation

Press the 'Evaluation' button if You are new to ETERM and simply want to try out this software

Dessa uppgifter fyller du i om du har en komplett distribution med diskett eller CD_ROM.

Om denna knapp är "aktiv" betyder det att du har någon tidigare version av *Eterm* som du kan uppgradera från.

Om denna knapp är aktiv kan du välja en 10-dagars utvärderingsperiod.

Beroende på hur du anskaffat *Eterm 6.1* ska du nu välja ett av alternativen:

- "New Installation" – Om du har en komplett distribution med diskett eller CD-ROM. Fyll i ditt namn (minst 6 tecken) och det programvaru ID som åtföljer distributionen. Klicka därefter "OK". Om du köpt programvaran via någon återförsäljare bör du nu också registrera din licens. Registreringen berättigar dig till 90 dagars fritt underhållsavtal (för enanvändare)

respektive 1 års fritt underhållsavtal (för platslicens).
Läs vidare på vår hemsida om de förmåner som åtföljer ett underhållsavtal.

- ”Upgrade” – Om denna knapp är aktiv betyder detta att du tidigare har en installation av Eterm 6 eller tidigare installation av Eterm 6.1 från vilken du kan uppgradera. Du ska då bara fylla i ditt namn (minst 6 tecken) och därefter klicka på ”Upgrade”.
- ”Evaluation” – Om du fått en utvärderingsversion av Eterm 6.1 klickar du på ”Evaluation”. Du har då 10 dagar på dig att skaffa dig en uppfattning om programmet, därefter måste du tillhandahålla ett programvaru ID. Om du efter perioden bestämmer dig för att fortsätta använda programmet ska du anmäla detta, se vidare på vår hemsida www.gmv.nu för information om olika typer av programvarulicenser.

Efter detta är nu *Eterm 6.1* klart att användas.

Återstoden av denna handbok beskriver de olika funktionerna i *Eterm 6.1*. För illustrationerna har vi mestadels använt *Eterm 6.1 for MC68* och detta kan innebära att vissa bilder och exempelfiler inte stämmer överens med den version du använder, arbetssättet är dock det samma för alla varianter.

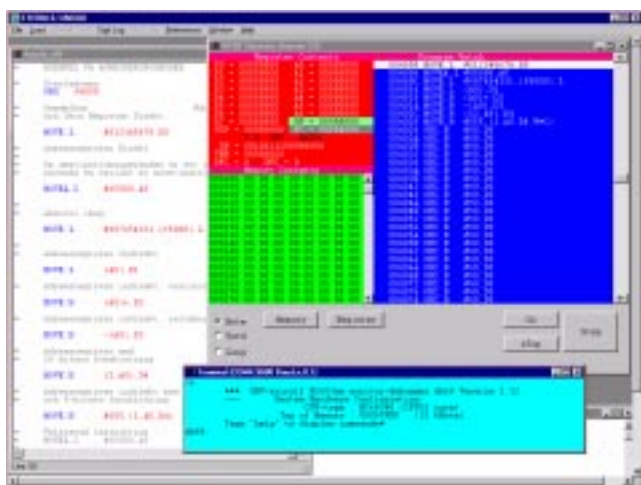
Eftersom de olika simulatorerna uppvisar större olikheter har vi dock behandlat dessa var och en för sig.

Funktioner

Eterm 6.1 har inbyggda funktioner för alla moment som ingår i programutvecklingen:

1. **Textredigering**, källtexten skrivs med hjälp av en *Editor*, filnamnet ska ha ett giltigt filnamnstilllägg (se tabell nedan) .
2. **Assemblering**, källtexten översätts av assemblern till en laddfil som innehåller maskinkoden. Laddfilen kan ha filnamnstilläggen *.S1* eller *.S2* beroende på variant. Vid assembleringen skapas också en så kallad listfil, denna har alltid filnamnstillägget *.LST*.
3. **Test**, laddfilen överförs till *Simulatorn* eller via *Terminalen* med hjälp av respektive laddningsprogram.

Version	Källtext	Laddfil
/MC68	.S68	.S2
/MD68k	.S68	.S2
/6809	.S09	.S1
/MC11	.S11	.S1
/MC08	.S08	.S1



ETERM håller ordning på förhållandena mellan de filer som skapas under programutvecklingen. Om du exempelvis försöker ladda ned en fil, vars källtext du modifierat, kommer *ETERM* automatiskt att assemblera den modifierade källtexten på nytt innan laddfilen överförs till simulatören.

I de följande avsnitten beskrivs, steg för steg, hur du arbetar med *ETERM*. I beskrivningarna används två exempelfiler, *AMODE.S68* och *PFLOW.S68*, dessa har hämtats från "*ETERM 6.1 för MC68*" där de kopierats till din hårddisk vid installationen. Även med de övriga versionerna medföljer exempel men dessa ser då annorlunda ut.

Starta nu *ETERM*

Redigera Källtexten

Textredigeringsverktyget är en standardenlig Windows-editor som dock även medger "färgad syntax". Detta betyder att textinnehållet analyseras, en korrekt symbol i första fältet färgas grön, en korrekt mnemonic i andra fältet färgas blå och ett (syntaktiskt) riktigt operandfält färgas rött. Text som uppfattas som kommentarer färgas grå. Observera att textredigeringsverktyget inte gör en fullständig assemblering, att fälten är korrekt färgade innebär bara att raden är syntaktiskt riktig.

Menyerna:

File

- **New | Source File**, skapa en ny källtextfil. Du måste ange ett nytt filnamn. (Alternativet **New | Terminal** beskrivs nedan)
- **Open**, öppna en befintlig källtextfil. Filnamnstilläggen för källtexter och listfiler är fördefinierade filter.
- **Save**, alla ändringar du gjort sparas i filen du arbetar med.
- **Save As**, spara under nytt namn.

Eterm 6.1

- **Save All**, du kan arbeta med flera filer samtidigt. Med detta val kommer samtliga dessa filer att sparas till hårddisken.
- **Close**, stänger den fil du för tillfället arbetar med. Om du inte tidigare sparat de ändringar du gjort får du en fråga om detta.
- **Print**, skicka filen till någon skrivare.
- **Preferences**,
 "Toggle Coloured Syntax" – Stäng av/Sätt på färgad syntax i textredigeringsverktyget.
 "Editor Font" – Ändra typsnitt i textredigeringsverktyget, observera att denna ändring inte påverkar de fönster du redan öppnat.
 "Output Window Font" – Ändra typsnitt i meddelandefönstret, dvs det fönster där assemblatorn skriver sina meddelanden.
 "Terminal Defaults" – beskrivs nedan.
- **Exit**, avsluta *ETERM*.

Edit

- **Cut**, eller "klipp ut" vald text. Du väljer text genom att placera markören på det första tecknet, trycka ned vänsterknappen på musen (håll den nere) dra markören till det sista tecknet du vill klippa ut och släpp vänsterknappen. Den markerade texten kan nu tas bort, tangentkombinationen för detta val är Ctrl-X. Den utklippta texten sparas internt så att du kan "klippa in" stycket någon annanstans (se "Paste" nedan). Detta är alltså en metod för att flytta textstycken.
- **Copy**, (Ctrl-C) detta är nästan samma som att klippa ut text, den markerade texten försvinner dock inte utan kopieras i stället till en intern lagringsplats. Du placerar ut den kopierade texten med "Paste" (se nedan).
- **Paste**, om du klippt ut, eller kopierat text kan du lägga till denna vid någon insättningspunkt. Placera markören på det ställe du vill skjuta in texten och tryck Ctrl-V.
- **Delete**, markerad text tas bort men sparas inte internt.
- **Select All** – märk all text i filen.
- **Find/Replace** – Standard "sök/finn/ersätt" funktion.

Assemblera källtexten...

För att assemblera en fil måste du först ha öppnat den i en editor, fönstret måste också vara aktivt, om det inte är aktivt (titelraden är då grå) placerar du markören på titelraden och klickar på fönstret.

Då ett editorfönster är aktivt kan du också välja alternativet "Assemble" från menyraden – filen i det aktiva editorfönstret assembleras nu.

Efter assembleringen aktiveras meddelandefönstret och här kan du se meddelanden från assembleratorn. Om källtexten innehöll några fel kommer varje fel att rapporteras på en egen rad i meddelandefönstret, genom att dubbelklicka på raden i meddelandefönstret aktiverar du editorfönstret med källtexten, markören placeras på början av den rad som är felaktig.

För att ytterligare underlätta felsökning och korrigering har *ETERM* flera inbyggda hjälpfunktioner. Välj från menyn:

Help | Contents

Innehållet i hjälpsystemet varierar med olika versioner av *ETERM*. Det är dock inte större utan att du snabbt kan skaffa dig en god uppfattning genom att navigera runt ett slag.

Listfiler

Vid assembleringen skapas, förutom laddfilen (.S1 eller .S2) även en listfil (.lst). Denna är avsedd som hjälp då du testat program i laborationsdatorn.

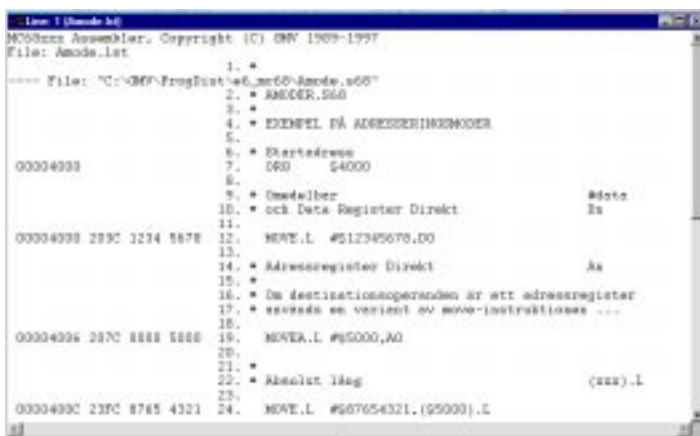
Välj:

File | Open

och därefter filtret ".lst", öppna någon listfil.

Eterm 6.1

Listfilen är en sammanställning av såväl källtextfilen som den genererade laddfilen. Längst till vänster ser du de absoluta adresserna (8 hexadecimala siffror) i måldatorns primärminne, därefter följer den kod (data) som genererats. Dessa visas som regel i grupper om 4 hexadecimala siffror. Därefter anges källtextfilens radnummer och slutligen känner vi igen källtexten.



```
1. *
2. * MASM68000 Assembler, Copyright (C) IBM 1989-1997
3. *
4. * EXEMPEL PÅ ADRESSER I PRIMÄRMINNE
5. *
6. * Startadress
7. 00004000 0000 0000 0000 0000
8. *
9. * Omedelbar och Data Register Direkt
10. 00004000 239C 3234 5678 9ABC
11. *
12. * Adressregister Direkt
13. 00004000 239C 3234 5678 9ABC
14. *
15. * Destinationsoveranden är ett adressregister
16. * avseende en variant av move-instruktionen ...
17. 00004000 239C 3234 5678 9ABC
18. *
19. * Absolut ladd
20. 00004000 239C 3234 5678 9ABC
21. *
22. *
23. *
24. *
25. *
26. *
27. *
28. *
29. *
30. *
31. *
32. *
33. *
34. *
35. *
36. *
37. *
38. *
39. *
40. *
41. *
42. *
43. *
44. *
45. *
46. *
47. *
48. *
49. *
50. *
51. *
52. *
53. *
54. *
55. *
56. *
57. *
58. *
59. *
60. *
61. *
62. *
63. *
64. *
65. *
66. *
67. *
68. *
69. *
70. *
71. *
72. *
73. *
74. *
75. *
76. *
77. *
78. *
79. *
80. *
81. *
82. *
83. *
84. *
85. *
86. *
87. *
88. *
89. *
90. *
91. *
92. *
93. *
94. *
95. *
96. *
97. *
98. *
99. *
100. *
```

Exempel på listfil

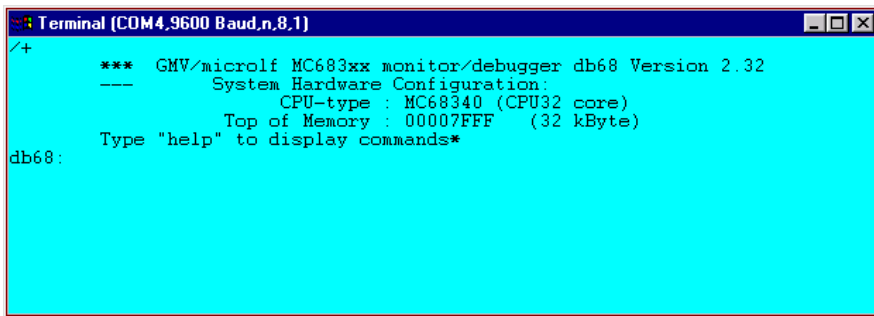
Att använda terminalen

Terminal är fönstret mot din laborationsdator. Via detta program kommunicerar du med laborationsdatorn, du kan föra över (ladda ned) ditt program för att därefter testa detta med laborationsdatorns inbyggda debugger. Du kan starta en instans av terminalen för varje ledig COM-port. Du måste själv kontrollera att du anslutit laborationsdatorn till rätt kommunikationsport på persondatorn.

Välj File | New | Terminal, och markera en tillgänglig COM-port.

Då ett terminalfönster är aktivt finns följande alternativ på meny-raden:

- **File** (har beskrivits tidigare)
- **Load**, För att ladda en fil till laborationsdatorn, du kan då välja på filer med tillägget **.S1**, **.S2** och **.S3**. Dessa förutsätts vara på Motorola S-format.
- **Cancel Load** – alternativet aktiveras då en laddning pågår. Om något går fel kan du här avbryta laddningen.
- **Start Log** – All kommunikation med mikrodatorn, dvs allt som skrivs i Terminalens fönster skrivs också till en fil (logfile.log).
- **Stop Log** – alternativet aktiveras efter ”Start Log” och används för att stänga filen ”logfile.log”.
- **Preferences** – Du kan variera terminalens bakgrundsfärg respektive textfärg, detta kan vara bra om du har flera terminalfönster öppna samtidigt, det går då snabbare att hitta rätt fönster. Med vissa versioner har du dessutom möjligheten att ställa kommunikationsparametrar för varje terminal. detta beskrivs nedan.



```
Terminal (COM4,9600 Baud,n,8,1)
/+
***  GMV/micro1f MC683xx monitor/debugger db68 Version 2.32
---   System Hardware Configuration:
      CPU-type  : MC68340 (CPU32 core)
      Top of Memory : 00007FFF (32 kByte)
      Type "help" to display commands*
db68:
```

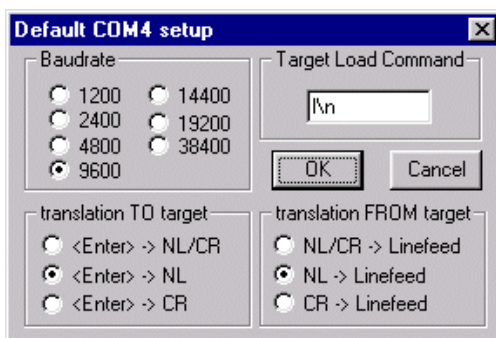
Terminalen i ETERM

Kommunikationsinställningar

I några versioner av *ETERM* har du också möjlighet att ställa in kommunikationsparametrar för terminalen. På detta sätt kan du anpassa *ETERM* till ett godtyckligt laborationssystem med den avsedda mikroprocessorn.

I fältet "Baudrate" ställer du in överföringshastigheten (8 bitar, 1 stopbit, ingen paritet gäller alltid).

I fältet "Target Load Command" skriver du in det kommando som ditt målsystem förväntar sig vid laddning. I exemplet visas laddkommandot för db68, dvs 'l' följt av *newline* (ASCII \$A), du kan även använda '\r' (för ASCII \$D).



Exempel på dialogbox

"Translation TO target" anger hur målsystemet förväntar sig att ny-rad anges, du har tre alternativ: teckenparet NL/CR tolkas som ny-rad av målsystemet, tecknet NL tolkas som ny-rad av målsystemet eller tecknet CR tolkas som ny-rad av målsystemet.

"Translation FROM target" anger hur målsystemet anger ny-rad, du har tre alternativ: teckenparet NL/CR (ASCII \$A/ASCII \$D) skickas av målsystemet; tecknet NL skickas av målsystemet eller tecknet CR skickas av målsystemet.

Simulatorn (MC68 och MD68k)

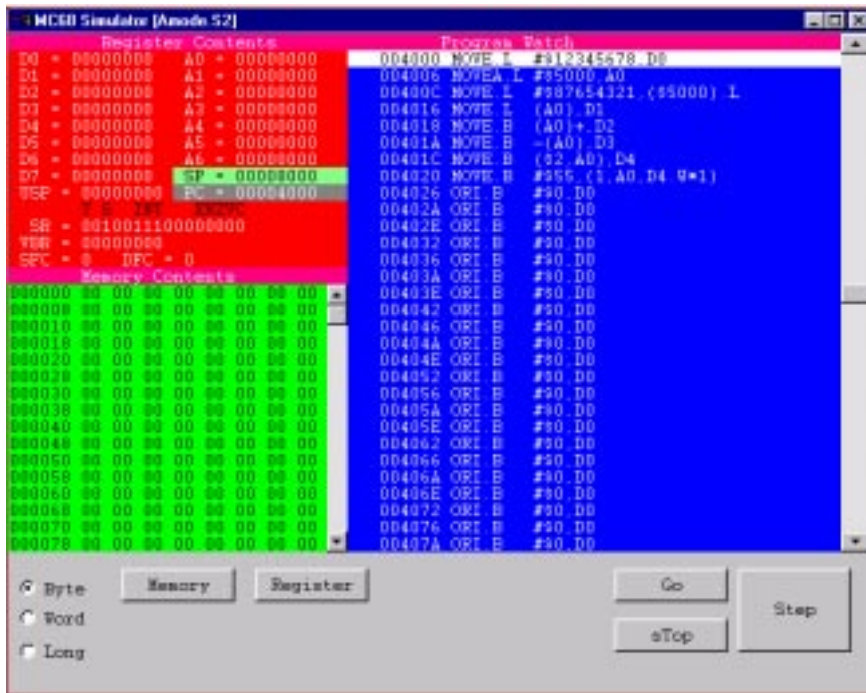
Simulatorerna för MC68 och MD68k är mycket likartade, vi nöjer oss därför här med att beskriva MC68-versionen.

Öppna filen (File | Open) AMODE.S68.

Välj från menyn:

Simulator

filen AMODE.S2 laddas nu till simulatorn.



Simulatorn delar upp bildskärmen i tre olika fält:

- "Program Watch" (*programfönster*)
- "Register Watch" (*registerfönster*)
- "Memory Contents" (*minnesfönster*)

Eterm 6.1

Programfönstret ("Program Watch") visar en dissassemblering av det program som för tillfället simuleras. Startadressen för fönstret anges som regel av innehållet i register **PC**. Den instruktion som står i tur att simuleras visas mot en vit bakgrund. Observera att oinitierat minnesinnehåll alltid är 0, detta visas som instruktionen:

```
ORI.B #$0,D0.
```

Registerfönstret ("Register Contents") visar innehåll i den simulerade maskinens register. Detta fönster uppdateras efter instruktionssimulering eller manuell ändring av registerinnehåll. Observera att register **A7 (SP)** såväl som register **SR** tilldelas speciella initiala värden, medan övriga register nollställs. Observera vidare att innehållet i **SR** ges på *binär* form medan värden i övrigt ges på *hexadecimal* form.

Minnesfönstret ("Memory Contents") visar en sekvens kontinuerligt minnesinnehåll med adressangivelse till vänster. Du kan ställa visningen på *byte*-, *word*- eller *long*-format. Med bläddringslistan ställer du enkelt in den del av minnet du vill övervaka.

Ett antal knappar används för att manövrera simulatoren:

- *Step* (eller tangentkombinationen Alt-S) används för att simulera utförandet av en instruktion.
- *Go*, (Alt-G) startar en långsam simulering. Mellan varje instruktion görs en fördröjning (c:a 0,5 sekund) osv. Du avslutar med *sTop* (Alt-T).
- *Register* (Alt-R) används för att ändra registerinnehåll manuellt.
- *Memory* (Alt-M) används för att ändra minnesinnehåll manuellt.

OBSERVERA: Begränsningar hos simulatoren; av olika skäl kan inte alla instruktioner simuleras. Via hjälpsystemet kan du utröna vilka instruktioner det gäller.

Låt nu simulatoren utföra programmets första instruktion.
Klicka på "Step"...

MC68 Simulator [Amode.S2]			
Register Contents		Program Watch	
D0 = 12345678	A0 = 00000000	004000 MOVE.L #\$12345678.D0	
D1 = 00000000	A1 = 00000000	004006 MOVEA.L #\$5000.A0	
D2 = 00000000	A2 = 00000000	00400C MOVE.L #\$87654321.(\$5000).L	
D3 = 00000000	A3 = 00000000	004016 MOVE.L (A0).D1	
D4 = 00000000	A4 = 00000000	004018 MOVE.B (A0)+.D2	
D5 = 00000000	A5 = 00000000	00401A MOVE.B -(A0).D3	
D6 = 00000000	A6 = 00000000	00401C MOVE.B (\$2.A0).D4	
D7 = 00000000	SP = 00008000	004020 MOVE.B #\$55.(1.A0.D4.W*1)	
USP = 00000000	PC = 00004006	004026 ORI.B #\$0.D0	
		00402A ORI.B #\$0.D0	

Observera instruktionens inverkan på innehållet i register D0. Observera också hur innehållet i PC ändras och nästa instruktion märks upp.

Du kan också ändra innehållet i registren manuellt...
Klicka på "Register" eller tryck "Alt-R": Ändra därefter innehållet i register D0 på följande sätt.

Modify Register

Register

D0

D0-D7, A0-A7, PC, SP, USP etc.

New Value

ABCD1234

prefix 'b' for binary

prefix 'h' for hexadecimal

no prefix for decimal

OK

Cancel

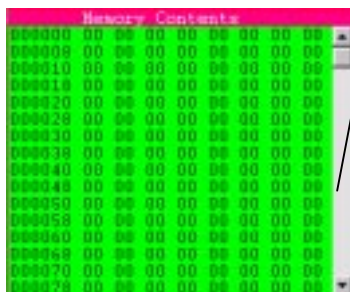
Klicka "OK"...

Register Contents			
D0 = ABCD1234	A0 = 00000000		
D1 = 00000000	A1 = 00000000		
D2 = 00000000	A2 = 00000000		
D3 = 00000000	A3 = 00000000		
D4 = 00000000	A4 = 00000000		
D5 = 00000000	A5 = 00000000		
D6 = 00000000	A6 = 00000000		
D7 = 00000000	SP = 00008000		
USP = 00000000	PC = 00004006		

Innehållet i D0 har ändrats.

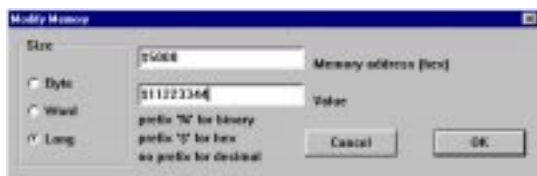
Eterm 6.1

Låt oss nu se hur vi kan övervaka en del av det simulerade minnet.

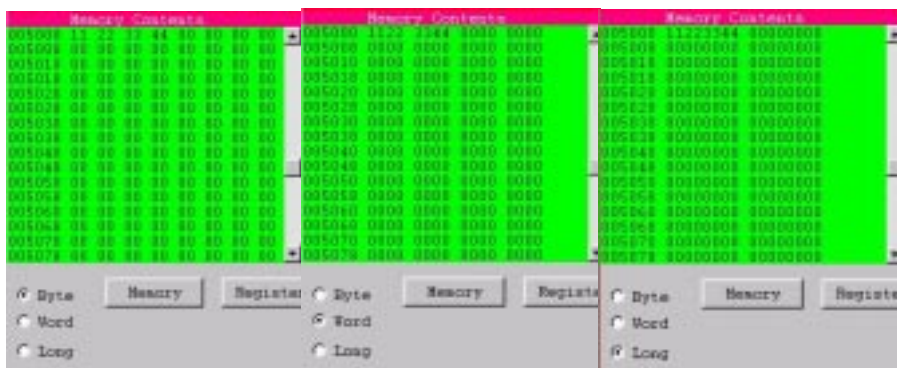


Använd rullningslisten till minnesfönstret för att ställa in minnesområdet \$5000 och framåt...

Även innehållet i *Simulatorns* minne kan ändras manuellt. Klicka på "Memory" eller tryck "Alt-M"... Du kan ändra "Byte", "Word" eller "Long". Prova med att ändra innehållet på adress \$5000 på följande sätt.



Genom att klicka på någon av knapparna "Byte", "Word" eller "Long" längst ned till vänster i *Simulatorn* kan du även ställa om visningen av minnet



Efter dessa inledande övningar bör du nu klara av att använda *Simulator*. Stega igenom återstoden av programmet. Ta det lungt och observera den inverkan instruktionsexekveringen har på register och minne.

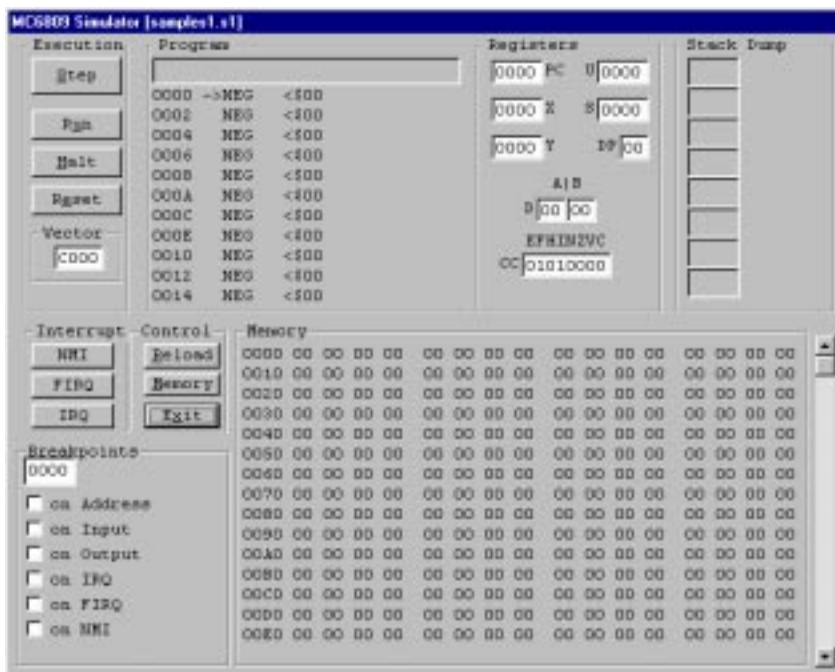
Simulatorn (6809)

Öppna filen (File | Open) SAMPLES.S09.

Välj från menyn:

Simulator

filen SAMPLES1.S1. laddas nu till simulatorn.



Simulatorns fönster delas av olika fält:

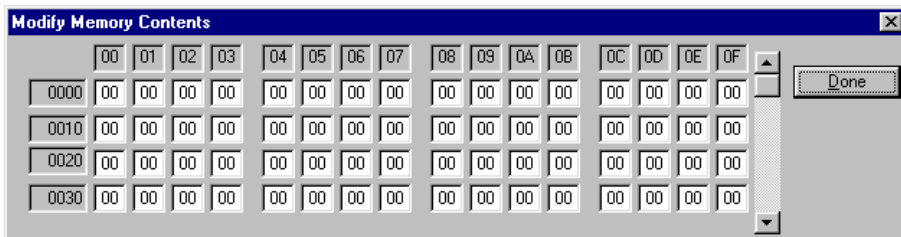
Eterm 6.1

- "Program" *programfönster*, här visas en dissassemblering av det laddade programmet. Pilen märker ut den instruktion som står i tur att utföras. Längst upp anges den sist utförda instruktionen.
- "Execution" *exekveringskontroll*, Fältet **Vector** anger en reset-vektor, dvs den adress som överförs till programräknaren vid omstart. Här ser vi att denna har värdet C000 vilket innebär att det laddade programmet startar på denna adress (första "ORG"-direktiv i programmet). Genom att klicka på **Reset** överförs denna adress till programräknaren. Med **Step** utför du en instruktion, med **Run** startar simulatören exekvering av programmet. Exekveringen kan stoppas genom att du klickar på **Halt** eller vid en *brytpunkt* (beskrivs nedan).
- "Registers" *registerinnehåll*, anger innehållet i 6809's register. Du kan ändra dessa innehåll manuellt genom att klicka på fönstret du avser och skriva in ett nytt värde. För ackumulatorer och indexregister tolkas det inskrivna värdet på hexadecimal form. För CC tolkas värdet på binär form.
- "Stack Dump" *stackinnehåll*, aktiveras av instruktioner som påverkar stackregistret. Visar stackpekarens värde och innehållet i adresser kring detta värde.
- "Memory" *minnesfönster* visar minnesinnehåll i en 256 bytes minnesarea som du kan ändra med hjälp av bläddringslistan längst ut till höger. Detta fönster kan endast *visa* minnesinnehåll. Senare visar vi hur du enkelt *ändrar* minnesinnehållet.
- "Interrupt" *avbrottshantering* här kan du få simulatören att utföra avbrottshantering. Genom att klicka en gång på **IRQ**, **FIRQ** eller **NMI** startar du simulering av motsvarande avbrottshantering hos 6809.
- "Breakpoints" *brytpunkter* används då du låter simulatören utföra **Run** men samtidigt vill avbryta programexekveringen vid någon speciell händelse. *on Address* innebär att simulatören avbryter exekveringen då den adress som anges i adressfönstret ovanför

knappen påträffas. *on Input* innebär att simuleringen avbryts om simulatören utför någon instruktion som orsakar en läsning från IO-arean (E000-E0FF). På motsvarande sätt innebär *on Output* att simulatören avbryter exekvering vid utmatning till IO-arean. Slutligen kan du sätta brytpunkter vid simulering av de olika avbrotten, exempelvis för att följa programexekveringen vid avbrott i detalj.

- "Control" övergripande kontroll, du kan ladda ner ditt program på nytt utan att avsluta simulatören med **Reload**. **Exit** avslutar simulatören och återgår till textredigering eller terminalemulering. Med **Memory** kan du ändra minnesinnehåll hos simulatören.

Klicka nu på **Memory**, följande fönster visas då:



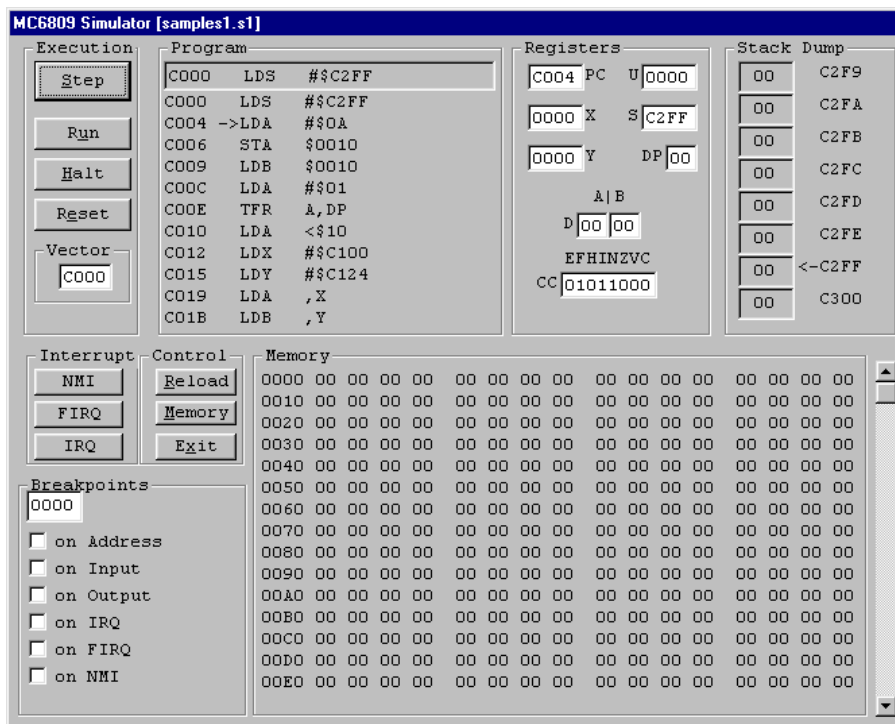
Innehållet i en 64 bytes stor minnesarea visas nu på fyra rader. Längst till vänster anges startadresser för varje rad och överst anges offseten hos raderna. Du kan enkelt ändra innehållet på enskilda minnesadresser genom att klicka på en vit ruta och skriva in det nya innehållet (tolkas på hexadecimal form). Med bläddringslistan till höger ändrar du snabbt för visning av andra delar av minnet. Klicka på **Done** när du är klar.

Testa nu programmet SAMPLES1. Klicka på **Reset** och observera ändringarna...

Eterm 6.1

Reset-vektorn (C000) överförs till programräknaren och programfönstret uppdateras. Du ser nu en dissassemblering av det laddade programmet.

Klicka på **Step**, en instruktion (LDS) utförs. Pilen flyttas till nästa instruktion, observera också ändringarna i *stackinnehåll*.



Fortsätt nu på egen hand. Du kan ju börja med de medföljande exempelfilerna:

- SAMPLES1 - illustrerar olika adresseringssätt hos 6809
- SAMPLES2 - visar hur ett subrutinanrop går till
- SAMPLES3 - illustrerar avbrottshantering
- SAMPLES4 - visar hur simulatören beter sig vid in- /utmatning.

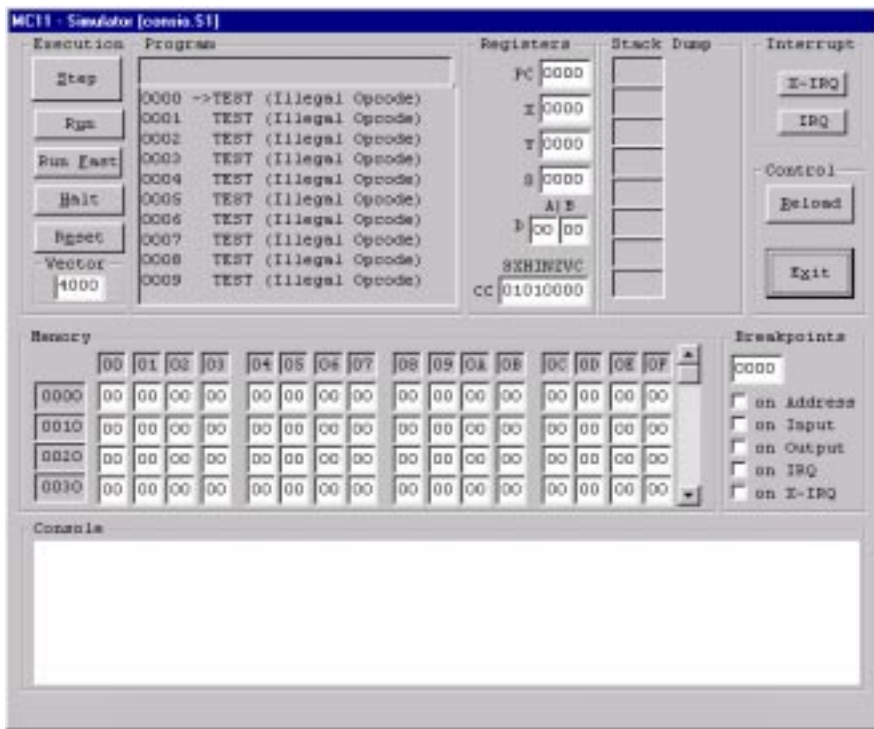
Simulatorn (MC11)

Öppna filen (File | Open) CONSIO.S11.

Välj från menyn:

Simulator

filen CONSIO.S1. laddas nu till simulatorn.



Simulatorns fönster delas av olika fält:

- "Program" *programfönster*, här visas en dissassemblering av det laddade programmet. Pilen märker ut den instruktion som står i tur att utföras. Längst upp anges den sist utförda instruktionen.
- "Execution" *exekveringskontroll*, Fältet **Vector** anger en reset-vektor, dvs den adress som överförs till programräknaren vid omstart. Här ser vi att denna har

Eterm 6.1

värdet 4000 vilket innebär att det laddade programmet startar på denna adress (första "ORG"-direktiv i programmet). Genom att klicka på **Reset** överförs denna adress till programräknaren. Med **Step** utför du en instruktion, med **Run** startar simulatören exekvering av programmet. Exekveringen kan stoppas genom att du klickar på **Halt** eller vid en *brytpunkt* (beskrivs nedan).

- "Registers" *registerinnehåll*, anger innehållet i 68HC11's register. Du kan ändra dessa innehåll manuellt genom att klicka på fönstret du avser och skriva in ett nytt värde. För ackumulatorer och indexregister tolkas det inskrivna värdet på hexadecimal form. För CC tolkas värdet på binär form.
- "Stack Dump" *stackinnehåll*, aktiveras av instruktioner som påverkar stackregistret. Visar stackpekarens värde och innehållet i adresser kring detta värde.
- "Memory" *minnesfönster* visar minnesinnehåll i en 64 bytes minnesarea som du kan ändra med hjälp av bläddringslistan längst ut till höger. Du kan här enkelt ändra minnesinnehållet.
- "Interrupt" *avbrottshantering* här kan du få simulatören att utföra avbrottshantering. Genom att klicka en gång på **IRQ**, eller **X-IRQ** startar du simulering av motsvarande avbrottshantering hos 68HC11.
- "Breakpoints" *brytpunkter* används då du låter simulatören utföra **Run** men samtidigt vill avbryta programexekveringen vid någon speciell händelse. *on Address* innebär att simulatören avbryter exekveringen då den adress som anges i adressfönstret ovanför knappen påträffas. *on Input* innebär att simuleringen avbryts om simulatören utför någon instruktion som orsakar en läsning från IO-arean. På motsvarande sätt innebär *on Output* att simulatören avbryter exekvering vid utmatning till IO-arean. Slutligen kan du sätta brytpunkter vid simulering av de olika avbrotten, exempelvis för att följa programexekveringen vid avbrott i detalj.

- "Control" övergripande kontroll, du kan ladda ner ditt program på nytt utan att avsluta simulatoren med **Reload**. **Exit** avslutar simulatoren och återgår till textredigering eller terminalemulering.
- "Console" konsolfönstret avser att illustrera en tänkt terminal ansluten till 68HC11's serieanslutning.

Simulera nu exekveringen av det laddade programmet. Börja med att klicka på **Reset** så att PC får rätt startvärde.

Klicka nu på **Run** och observera hur instruktionsexekveringen kommer igång.

Klicka nu i konsolfönstret, du får nu en markör här. Skriv in något tecken via tangentbordet och observera vad som händer. Tecknet visas snabbt och försvinner, observera nu hur programflödet ändras och tecknet så småningom dyker upp i konsolfönstret igen.

Studera även källtexten till programmet (CONSIO.S11).

Anmärkning:

Som komplement till denna beskrivning av MC11-simulatoren kan du också använda "MC11 – Hårdvarubeskrivning".

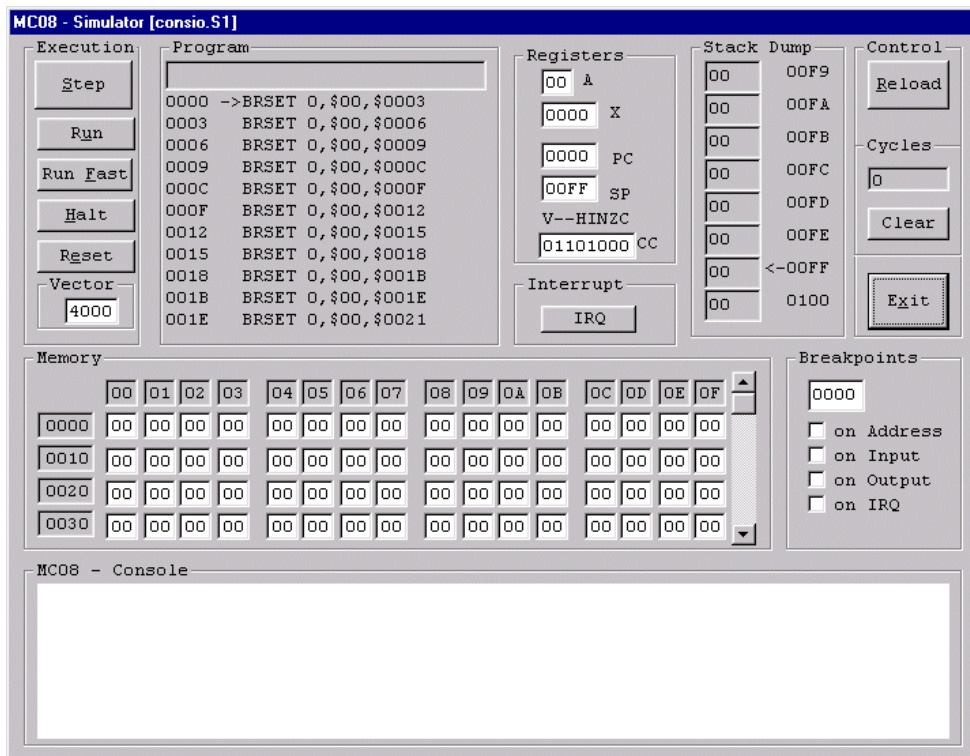
Simulatorn (MC08)

Öppna filen (File | Open) CONSIO.S08.

Välj från menyn:

Simulator

filen CONSIO.S1. laddas nu till simulatorn.



Simulatorns fönster delas av olika fält:

- "Program" *programfönster*, här visas en dissassemblering av det laddade programmet. Pilen märker ut den instruktion som står i tur att utföras. Längst upp anges den sist utförda instruktionen.

- "Execution" *exekveringskontroll*, Fältet **Vector** anger en reset-vektor, dvs den adress som överförs till programräknaren vid omstart. Här ser vi att denna har värdet 4000 vilket innebär att det laddade programmet startar på denna adress (första "ORG"-direktiv i programmet). Genom att klicka på **Reset** överförs denna adress till programräknaren. Med **Step** utför du en instruktion, med **Run** startar simulatören exekvering av programmet. Exekveringen kan stoppas genom att du klickar på **Halt** eller vid en *brytpunkt* (beskrivs nedan).
- "Registers" *registerinnehåll*, anger innehållet i 68HC08's register. Du kan ändra dessa innehåll manuellt genom att klicka på fönstret du avser och skriva in ett nytt värde. För ackumulatorer och indexregister tolkas det inskrivna värdet på hexadecimal form. För CC tolkas värdet på binär form.
- "Stack Dump" *stackinnehåll*, aktiveras av instruktioner som påverkar stackregistret. Visar stackpekarens värde och innehållet i adresser kring detta värde.
- "Memory" *minnesfönster* visar minnesinnehåll i en 64 bytes minnesarea som du kan ändra med hjälp av bläddringslistan längst ut till höger. Du kan här enkelt ändra minnesinnehållet.
- "Interrupt" *avbrottshantering* här kan du få simulatören att utföra avbrottshantering. Genom att klicka en gång på **IRQ**, startar du simulering av motsvarande avbrottshantering hos 68HC08.
- "Breakpoints" *brytpunkter* används då du låter simulatören utföra **Run** men samtidigt vill avbryta programexekveringen vid någon speciell händelse. *on Address* innebär att simulatören avbryter exekveringen då den adress som anges i adressfönstret ovanför knappen påträffas. *on Input* innebär att simuleringen avbryts om simulatören utför någon instruktion som orsakar en läsning från IO-arean. På motsvarande sätt innebär *on Output* att simulatören avbryter exekvering vid utmatning till IO-arean. Slutligen kan du sätta

Eterm 6.1

brytpunkter vid simulering av de olika avbrotten, exempelvis för att följa programexekveringen vid avbrott i detalj.

- "Control" övergripande kontroll, du kan ladda ner ditt program på nytt utan att avsluta simulatoren med **Reload**. **Exit** avslutar simulatoren och återgår till textredigering eller terminalemulering. Vid varje instruktion exekvering uppdateras denna genom att antalet utförda processorcykler adderas ("Cycles"). Du nollställer den genom att klicka på **Clear**.
- "Console" konsolfönstret avser att illustrera en tänkt terminal ansluten till 68HC08's serieanslutning.

Simulera nu exekveringen av det laddade programmet. Börja med att klicka på **Reset** så att PC får rätt startvärde.

Klicka nu på **Run** och observera hur instruktionsexekveringen kommer igång.

Klicka nu i konsolfönstret, du får nu en markör här. Skriv in något tecken via tangentbordet och observera vad som händer. Tecknet visas snabbt och försvinner, observera nu hur programflödet ändras och tecknet så småningom dyker upp i konsolfönstret igen.

Studera även källtexten till programmet (CONSIO.S08).

Anmärkning:

Som komplement till denna beskrivning av MC08-simulatoren kan du också använda "MC08 – Hårdvarubeskrivning".

Underhåll av ETERM 6.1

**Betrakta din programvarulicens som en investering.
Försäkra dig om största möjliga avkastning från denna investering.**

Med din programvarulicens följer också rätten till ett underhållsavtal.

- Enanvändarlicens, kostnadsfritt underhållsavtal i 90 dagar från registreringsdatum.
- Platslicens, kostnadsfritt underhållsavtal i 1 år från registreringsdatum.

Underhållsavtalet ger dig rätten till:

- Kostnadsfri hjälp vid installation och användning av programvaran via GMV's supportorganisation.
- Kostnadsfri uppgradering till reviderade versioner av programvaran.
- Kraftigt rabatterade priser vid uppgradering till ny (annan) version av programvaran.

Om underhållsavtalet gäller en platslicens kan du också (utan extra kostnad) beställa tjänsten "SUPPORTSIDA", läs mer om denna tjänst på GMV's hemsida.

Ytterligare kostnadsfria tjänster kan komma att erbjudas inom ramen för underhållsavtalet, dessa meddelas då i första hand via GMV's hemsida, inte nödvändigtvis via utskick.

Förutsättningar

För att underhållsavtalet ska börja gälla krävs att licensen är registrerad av GMV.

- För *enanvändarlicenser*: Med programvaran följer en programvarulicens, fyll i uppgifterna under REGISTRERING och skicka/faxa denna till GMV. Om

Eterm 6.1

du av någon anledning inte skulle ha fått denna programvarulicens med programvaran kan du rekvirera den från GMV. Du kan också hämta den (PDF-format) från vår hemsida. Observera att varje enanvändarlicens kräver registrering.

- För *platslicenser* gäller vanligtvis att denna registrering sker före leverans, ytterligare registrering krävs då inte. Om du köpt ditt program via någon av våra återförsäljare måste du dock registrera även platslicensen. Använd den blankett som medföljer programdistributionen.

Då licensen registrerats av oss skickar vi en kopia av programvarulicensen med all registreringsinformation (medföljer direktleveranser). Av denna framgår då avtalets beteckning, underhållsperiod mm. Dessa uppgifter måste du ha till hands då du utnyttjar tjänsterna i underhållsavtalet.

Giltighet, förnyelse...

Underhållsavtalet löper i perioder om 1 år. Förutsättningarna för att underhållsavtalet ska gälla är att underhållsperioden är kontinuerlig, dvs förnyelse av underhållsperiod måste ske innan föregående period löpt ut.

Förnyelse av underhållsperiod sker genom att en blankett fylls i och skickas eller faxas till GMV. Förnyelse kan också begäras via epost (register@gmv.nu).