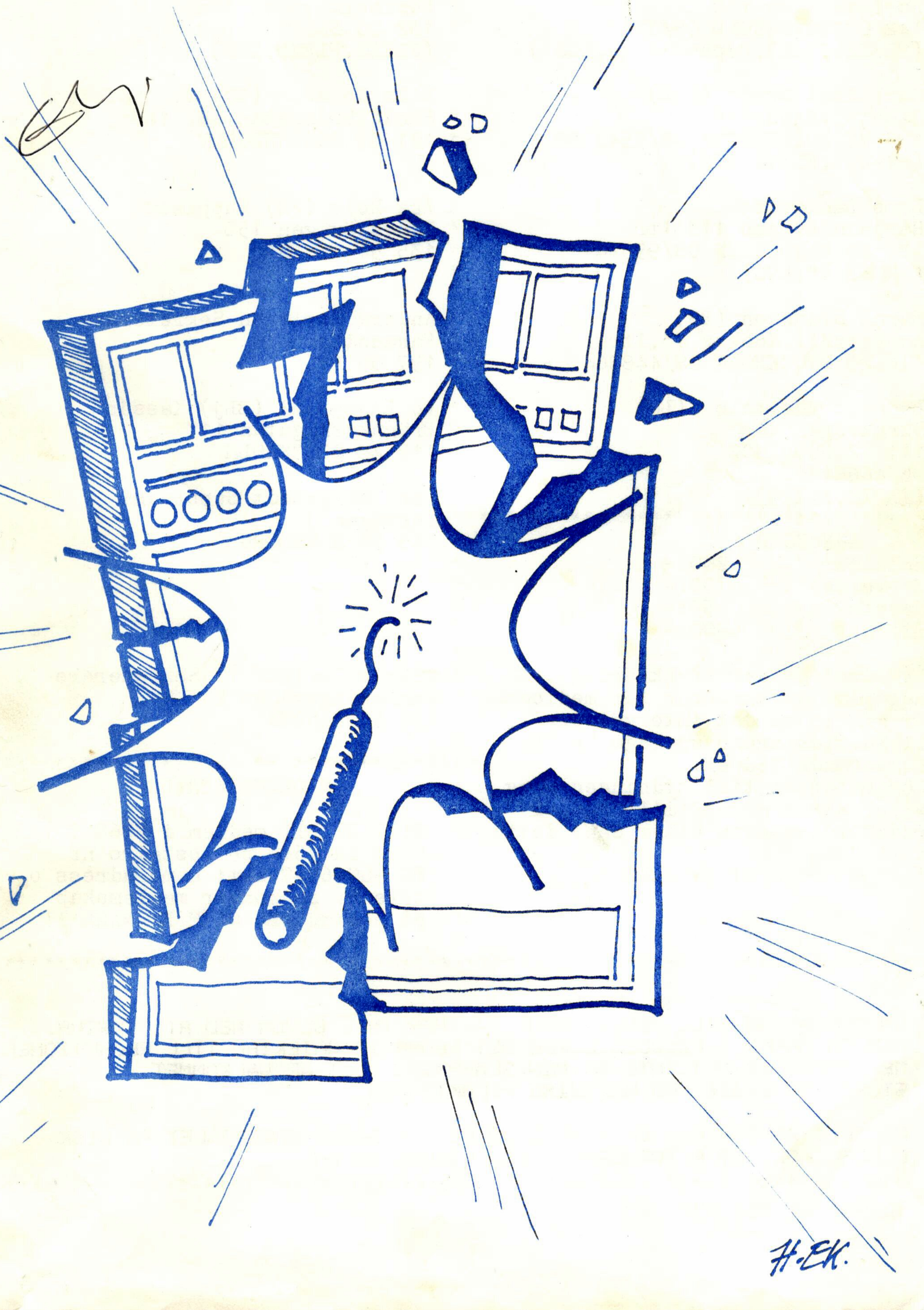
MPU-laren nr 1 1981

TIDSKRIFT FÖR PRIVATDATAKLUBBEN PD68



REDAKTIONEN

MPU-larenredaktionen består av

Roger Holmstrand (RH)
Rapsodivägen 168
142 00 TRÅNGSUND 08/7713852
(FLEX2,FLEX9,Papperstape,KCS)

Carl Axel Bruno (CAB)
Timotejvägen 116
191 77 SOLLENTUNA 08/7543788
(D2-kit,FLEX2,KCS)

Lars Landqvist
Harpsundsvägen 113 1tr
124 40 BANDHAGEN 08/995141
(SMOKE 5",KCS)

Bernt Svensson (BS)
co Petzäll,Kocksg 43,1tr
116 29 STOCKHOLM 08/448864

Lennart Uusitalo (LU)
Skrakvägen 20
141 72 HUDDINGE
08/466628
(FLEX2,FLEX9)

ANNONSER

Helsida	200:-
Halvsida	100:-
Kvartsida	50:-
Bilagor	200:-

Priserna gäller tryckfärdigt material.Radannonser för medlemmar är gratis om det inte gäller firma.Tidningens upplaga är för närvarande 350 ex.
För material till tidningen ring eller skriv gärna till någon i redaktionen ovan eller sänd detta till,PD68
BOX 98 122 21 ENSKEDE 1

STYRELSEN

PD68 nuvarande styrelse.

Tommy Bladh (TBl) Ordförande
Norrholmsvägen 106
132 00 SALTSJÖ-BOO
(FLEX2,FLEX9,KCS)

Nils Ohlsson (NO) V. Ordförande
Sandfjärdsgatan 68, 12 tr
121 69 JOHANNESHOV

Åke Holm (ÅH) Suppleant
Långsjövägen 15B
135 54 TYRESÖ

Anders Karlsson Sekreterare
Fasanstigen 40
197 00 BRO

Bo Ljungblad (BLj) Kassör
Brännkyrkagatan 117
117 28 STOCKHOLM

Lars Bergstedt Suppleant
Forvägen 31
145 51 NORSBORG

Gunnar Lovius V. Sekreterare
Ragnebergsvägen 21
136 67 HANDEN

MEDLEMSINTRÄDE

Vill DU bli medlem i PD68 ?
Sätt in 75:- på postgiro nr
PG 960468-7 samt namn adress o tele nr så kommer medlemskap på posten.OBS GLÖM EJ NAMN!!!

EN UPPMANING TILL MEDLEMMARNA!!! VAR INTE BLYGA MED ATT SKRIVA.
MED SÅ MÅNGA I KLUBBEN BORDE DET REGNA IN MATERIAL TILL REDAKTIONEN.
MEN TYVÄR ÄR DET INTE SÅ.MEN SKÄRPNING.ALLT ÄR VÄLKOMMET
STORT SOM smått.MED HÄLSNING REDAKTIONEN

PS. DET ÄR FÖR OSS EN FÖRDEL OM NI KAN SÄNDA MATERIALET PÅ DISK ELLER TAPE. VI RETURNERAR SEDAN DISKEN.TAPEN.

I HUVUDET PÅ EN ORDFÖRANDE

Återigen har vi ett nytt oskrivet år framför oss. Och vi är fler än någonsin. Det har faktiskt varit en smått fantastisk anströmning av nya medlemmar under de senaste månaderna. Det kan vi nog i stor utsträckning tacka annonsen i Radio & Televisions Bygg-själv bok för. Den utgiften har för länge sedan tjänat in sig. En blänkare i RT om att CÅ kommer att fortsätta beskrivandet av periferikort till den populära "RT"-datorn, verkar också ha slussat många till PD68. Publicitet och PR hjälper! Det är inget snack om den saken.

En fråga som dock ibland dyker upp i huvudet är denna; -Hur stora kan vi bli utan att det börjar fresta på den organisation vi har. Att få iväg en omgång MPU-lare är snart ett dagsverke. Enbart med etiketter, frimärken, nedstoppning av blaskan i kuvert etc etc. Mellan varven tillkommer registrering av nya medlemmar. Se till att information går iväg till de som är nyfikna på oss. Kopiering av program på skivor och kassetter tar sin tribut. Men det här är saker som vi kan klara av genom att hjälpas åt.

Att vara många innebär ju många fördelar. Inte minst ekonomiskt. I tryckningen av MPU-laren ligger en stor kostnad för offsetplåtarna. Utslaget på fler exemplar blir varje MPU-lare billigare. Med de större resurserna bakom ryggen kan vi även numera honorera insända bidrag. Det är inte precis någon furstlig summa som erbjuds. Men kanske tillräckligt för att stegra intresset att dela med sig på saker som man slitit sitt hår för.

Att bjuda in folk och firmor till klubbmötena skall vi fortsätta med. Den som har ideer och/eller känningar bör höra av sig. Det är en fördel om vi kan skicka ut extra kallelser med aktuell information.

Resultatet av enkäten syns delvis redan i denna MPU-lare. Jag tänker då förstäss på de olika varianterna på utskrifter av registret som finns längre bak i tidningen. De avdelningen som rörde periferikorten och MPU-laren kommer att presenteras i nästa utgåva av tidningen. Klart är att intresse finns för undervisning och utbildning i programmeringsspråk. Vi har två olika Pascal system på gång. Ett från den holländska firman Lucidata, vars system ligger under Flex. Det andra skall vara en fullvuxen USCD variant för 6809. Den förvandlar datorn till en Pascal maskin. Med egct diskoperativsystem, egen I/O etc. Rapport om våra rön kommer i MPU-laren så snart som möjligt. Låter det inte intressant? Det tycker i alla fall:

TBl

INNEHÅLLSFÖRTECKNING

Program för realtidsklockan CA 8005.	4
Försvenskning av Flex 9 assemblern.	6
Sortering. Del 2: Heapsort.	18
Brev om Pascal för det lilla systemet.	21
WOM. Nytt fantastiskt minne.	22
Patch utility för Flex filer.	22
Radannonser.	27
Disassemblers. LABDIS och DYNAMITE	28
ASCII inmatning i minnet från CBUG.	35
Exormacs och 68000 demonstration.	36
Spelhålan. GOBANG, ett 19*19 luffarschack.	40
XREF.CMD, korsreferens från Basic program.	46
Medlemsregistret. Komplett med utrustning. Postnummersorterat.	48

REALTIDSKLOCKAN CA 8005

I de flesta datorsystemen typ minidator och uppåt, brukar det finnas en realtidsklocka. Med detta menas en klocka som kan anropas av program och returnera värden på exempelvis aktuell tid på dygnet. I bland kan man även få tiden uttryckt i antalet millisekunder sedan midnatt o dyl. Givetvis kan även årtal och datum begäras från klockan. Datorerna använder dessa tidsuppgifter för en mängd olika ändamål. Vanligt är att man kan se när filer har skapats och när de senast refererades. I många datasystem tar man betalt av användaren. Då måste man noga mäta resursuttaget från datorn. Varje händelse i datorn loggas. Man registrerar exempelvis hur mycket primärminne man använt och under hur lång tid. Hur länge man belagt I/O kanaler etc. Att en klocka med upplösning ned på tusendelar av sekunder då måste användas, är lätt att inse.

Klocka för FLEX 2/9 systemen

Vad har man då för glädje av en realtidsklocka i sitt FLEX system? Datum finns ju redan från början. Systemet lägger in det i varje fils directory då filen skapas eller uppdateras. Det kan man sedan läsa ut med hjälp av DIR.CMD eller en modifierad CAT.CMD. Olyckligtvis var man på TSC inte framsynt (?) nog när man dimensionerade storleken på en fils directory. Ingen plats för klockslag finns reservarad. Av de 24 bytes som varje directory upptar, finns visserligen två icke använda bytes. Reserverade för "framtida" bruk säger manualen. Men två bytes räcker inte för att få upplösning på en-sekunders nivå. Minst tre bytes åtgår. Även med ren binär lagring går det åt mer än 16 bitar. Förmodligen skulle vi kunna knycka en bit någon annan stans i directoryt, men det verkar lite jippoartat.

Nej, filerna får vi nog skippa så länge. Utan det blir främst de egna applikations programmen som kan dra nytta av en klocka. Program för övervakning, mätning och registrering etc. Men även när man listar något på skrivaren. Själv brukar jag anropa klockan först när en assemblerlista skall ut. Då får jag datum och klockslag överst på första sidan. Det gör det lite lättare att hålla reda på olika listor. Särskilt om man glömt att stega någon revisions siffra.

CLOCK.CMD

Programmet som jag kommer att visa här är en sk utility. Dess främsta uppgift är att ställa klockkretsen på periferikortet. Därför innehåller programmet kontroller av att det man matar in är tillåtna värden. Vidare behöver man kunna ställa klockan på olika sätt. Själv brukar jag ställa uret genom att ange tiden på kommandoraden. Men om detta alltid skall ske varje gång som datorn startas upp, via STARTUP.TXT funktionen, fungerar inte den metoden. Därför kan man ge speciella options (extra parametrar) på kommandoraden. Med ett visst option kan programmet fås att ställa en fråga om tiden och ta in den från tangentbordet. Vidare kan man få klockan att ställas på önskad tid och sedan frisläppas på kommando. Bra om man vill ha noggrann ställning med Fr Ur som grund.

CLOCK.CMD är som nyss sagts främst tänkt för att ställa klockan. Men programmet kan givetvis även hämta tiden från klockan och skriva ut den. Eftersom man gissningsvis även i regel är intresserad av datumet kan detta också utmatas. Värdena hämtas från FLEX datum register. Datum eller ej styrs med option. En komplett rad kan därför få följande utseende:

DAT: 81-01-23 18:20:19

Programmet finns för både FLEX2 (6800) och FLEX9 (6809). Den senare består av en (lätt)

konverterad 6800 kod. Listan som visas är dock 6800 varianten. Alla referenser sker därför till den.

Funktionsbeskrivning

Först nollställs det dataminne som programmet använder. Detta innebär bla. att alla option flaggor får värdet "FALSE". Därefter kollar programmet om tiden är angiven på kommandoraden eller om options finns eller om det är ingendera. Är tid angiven stoppas siffrorna in i en buffert, strippade från ASCII delen. När detta är klart kollar man om options följer. I så fall tas dessa om hand vid MA1500. En flagga som motsvarar varje giltigt option sätts skilt från noll ("TRUE").

När så småningom alla tecken har tagits om hand från kommandoraden, hamnar programmet vid MA5000. Har nu option Q givits kommer programmet att begära in tiden från tangentbordet. Analysen sker därefter i samma programdel där vi var nyss. Om klockan skall ställas och vi har givit tiden som sex siffror, hamnar vi vid MA6050. Här kollas att tiden är rimlig, dvs inte överstiger 23:59:59. Efter detta ställs klockan. Detta sker i subrutinen SETTIM. Tiden förutsätts ligga i de tre ram cellerna CLBUFF. Ställningen blir lite bökig eftersom klockkretsen är släkt med den som sitter i vanliga klockradios o dyl. Man ställer den genom att trycka på tre knappar. Snabbt, långsamt och hålla stilla. Skillnaden här är att programmet "trycker" på knapparna och kollar vad det blir.

Nåväl, om option H är givet hålls klockan och startas vid tryckning på RETUr. Detta sker vid MA6750. Om vi bara ställer klockan sker ingen utskrift av tiden. Det är ju ganska onödigt då den är exakt vad vi skrev. Med option D kan vi dock alltid få ut både datum och tid (MA7000). Innan dess kan vi välja att inte ge CR+LF först (MA7050) genom att ge option C. Efter att tiden skrivits ut (MA7120) kan vi välja att ge CR+LF. Styrt av option L.

Användbara subrutiner.

Funktionerna har med avsikt isolerats i många subrutiner. Inte av orsak att spara bytes, utan av rent strukturella skäl. Det skall vara lätt att ändra funktioner och man skall kunna anropa, eller knycka, lämpliga rutiner. Därför kan vi hitta enskilda subrutiner som ställer klockan, skriver ut tid från den, skriver ut datum etc.

NEXTCH hämtar ett tecken från FLEX inbuffert. Kontrollerar om bufferten är slut.

DIGCHK strippar av ASCII delen från en siffra och anger om tecknet var siffra.

DAADIG tar två siffror från sifferbufferten och smäller i hop dem i en byte.

DAATIM hämtar två siffror från klockan. Lagrat i en byte. X-registret skall innehålla ett tabellvärde för att kunna adressera PIA:n till klockan. Se rutin DISTIM.

DISTIM hämtar all tid från klockan och ser till att den skrivs ut.

SETUP sätter upp PIA:n som interfacar mot klockkretsen.

SETTIM ställer klockan. Data skall ligga i tre ramceller (tim, min och sekunder). Först ställs sekunderna med långsam speed. När man ställer timmar och minuter med snabb speed förändras inte sekunderna. Några tester på vad klockan råkar stå på, kan snabba upp korrigeringar.

INSERT skriver ut vissa felmeddelanden där själva det felaktiga tecknet insätts (inserted) mellan två paranteser.

DISDAT skriver ut en datumsträng i formatet "ÅÅ-MM-DD". Värdena kommer från FLEX egna variabler.

BINDAA används för att enkelt omvandla ett binärt värde <99 till ett decimaljusterat

hexadecimalt värde.

Options

För att få lite möjlighet att påverka funktionen hos CLOCK.CMD kan vissa sk. options ges. Dessa består av enkla tecken och ges efter ett inledande "+" tecken. Följande options finns:

C Förhindrar att utskriften föregås av CR+LF. Bra om utskriften skall passas in efter något redan utskrivet på raden.

D Ger utskrift av både datum och klockslag.

H Håller klockan låst utan att den stegar. Bra om man vill ställa den noga på sekunden när. Startas med RETUR tangenten.

L Ger CR+LF efter utskrift. Används i de fall då man har en skrivare med radbuffert. Om klockutskriften är det sista på listan skulle den kunna bli kvar i bufferten om inget CR/LF kom.

Q Frågar efter tiden att ställa klockan med. Används vid automatisk STARTUP av FLEX. Man kan ge samtliga options även här. Använder FLEX inbuffert, varför inga fler kommandon kan följa efter på samma rad. Här kan vi se ett exempel på en kommandorad för utskrift av datum och tid både före och efter en assemblerlista. För att se hur lång tid den tog:

```
 P CLOCK,+D:P ASMB  CLOCK,+BNGS:P
CLOCK,+DL
```

Skicka diskett

Programmet finns som sagt i en version för FLEX2 och en annan för FLEX9. Det ligger i utility arean i respektive system, varför man kan tänka sig att anropa det från exempelvis Basic utan att störa något. Den som vill ha en kopia på CLOCK.CMD kan skicka en 5 1/4" skiva till mig, Tommy Bladh (Adress se sid 2). Ange version. OBS! Gratis!!

Själva periferikortet med klockkretsen kommer från CÅ-Elektronik AB, BOX 2010, 135 02 TYRESÖ, tel 08-742 34 01. Kan även köpas på TELKO i Stockholm, Göteborg och Malmö. Pris 249 kr exkl moms. Kortet var beskrivet i Radio & Television nr 10 1980.

TB1

The Don't Gate

You don't get no output no matter what's at the inputs. It is believed that the don't gate was the breakthrough that made the LSI write only memory possible.



DON'T GATE

FÖRSVENSKNING AV FLEX 9 ASSEMBLERN

Det har visat sig att 6809-MPUn. har en indirekt addresserings-mod. Denna mod använder sig utav stora paranteser (brackest) för att markera indirect addessering. det kan se ut så här:

```
LABEL   LDA   ÄB,XÅ
```

Som ni ser så ser det ganska konstigt ut. Det beror på att ANSI ASCII-standarden inte innehåller ÅÄÖ. Det hade varit bättre om man använde små paranteser, tex så här:

```
LABEL   LDA   (B,X)
```

Det är ganska enkelt att ändra detta och jag föreslår att det standardiseras. I flex är det bara 3-bytar du behöver ändra på, dessa är:

ADRESS	URSPRUNG	NYTT-VÄRDE
$0AAC	$5B	$28
$0B12	$5D	$29
$0C62	$5D	$29

Gör så här:

```
+++GET ASMB.CMD
+++MON    Gå till monitor och
          ändra enligt tabellen.
*G CD03   Varmstart utav flex
+++SAVE ASM.CMD,0,2C64,0
```

Jag hopas nu att du får det att fungera lika bra som jag fick det att göra. Eventuella synpunkter på detta motages gärna.

LU

```
DAT: 81-01-25   20:03:45
                 *
                 * CLOCK.CMD
                 *
                 * REV A/2 T Bladh
                 * 1981-01-25
                 *
                 * This utility is written for the
                 * digital clock CA-8005
                 *
                 * SYNTAX: CLOCK <hhmmss><,+OPTIONS>
                 *
                 * WHERE hh = hours
                 *       mm = minutes
                 *       ss = seconds
                 *
                 *       OPTIONS: Q Query about time.
                 *                H Hold clock until released.
                 *                D Display Date and time.
                 *                C Do not preced printout with CR+LF
                 *                L Send CR+LF after printout.
                 * NOTE: If option Q is used, then
                 * this destroys the remaining command buffer.
                 * No more multiple commands may continue on the
                 * same line in this case.
                 * Since option Q uses the line buffer,
                 * more options may be specified. Q is best used
                 * in a STARTUP file with CLOCK.
                 * In this case, obtaining the time from the command line
                 * does not make sense.
                 * If no time and no options are argumented,
                 * then only time is printed.
                 * Options given from the command line
                 * is not destroyed by Q.
                 * Option L is useful on a buffered lineprinter.
                 * If CLOCK sends the last line without L option,
                 * then nothing will be printed.
                 *
                 * FORMAT: DAT 80-12-13   15:23:35
                 *
                 * EXTERNAL REFERENCES
                 *
                 *
                 * LIB REFLIB.SYS
                 * ----------------------
                 ************************
                 *                      *
                 * REFERENCES IN        *
                 * FLEX 2.0             *
                 *                      *
                 * REV C 1980-01-06     *
                 *                      *
                 ************************
```

```
A840            FCB     EQU     $A840       SYSTEM FCB
B406            FMS     EQU     $B406       FILE MANAGEMENT
B403            FMSCLS  EQU     $B403       CLOSE ALL FILES
AD03            WARMS   EQU     $AD03       RE-ENTER FLEX
AD2D            GETFIL  EQU     $AD2D       GETS FILE SPECS
AD33            SETEXT  EQU     $AD33       SETS FILE EXT
AD3F            RPTERR  EQU     $AD3F       REPORTS ERROR
AD15            GETCHR  EQU     $AD15       INPUT CHAR
AD18            PUTCHR  EQU     $AD18       OUTPUTS CHAR
AD1E            PSTRNG  EQU     $AD1E       PRINTS STRING
AC0E            DATERG  EQU     $AC0E       3 BYTES(M,D,Y)
AD0F            OUTCH   EQU     $AD0F       OUTPUT CHAR
AD27            NXTCH   EQU     $AD27       NEXT CHAR FROM I/O BUFFER

AD1B            INBUFF  EQU     $AD1B       FILL LINEBUFFER
A080            LBUFF   EQU     $A080       LINE BUFFER
AC14            LBPOIN  EQU     $AC14       POINTER IN LINEBUFFER
                *
AC2B            EOM     EQU     $AC2B       CONTAINS LAST USER RAM
AD24            PCRLF   EQU     $AD24       CR+LF
AD39            OUTDEC  EQU     $AD39       B=0, NO LEADING ZEROS
AD3C            OUTHEX  EQU     $AD3C       2 HEX CHAR
A100            UTILAR  EQU     $A100       START OF UTILITY AREA
                *
                * END REFLIB.SYS
                * ----------------
                *
                *
                * LOCAL REFERENCES
                *
0002            VERS    EQU     2
0002            SLOW    EQU     2           SETTING SPEED IS SLOW
0001            FAST    EQU     1           FAST SPEED
0004            HOLD    EQU     4           HOLD CLOCK STILL
0000            RUN     EQU     0           LET IT TICK
8050            PIAURD  EQU     $8050       DATA REG OF CLOCK
8051            PIAURC  EQU     $8051       CONTROL REG
8052            PIAURS  EQU     $8052       SELECT DIGIT
AC02            EOLCH   EQU     $AC02       FLEX EOL CHAR
                *
                * START OF PROCESSING
                * -------------------
                *
A100                    ORG     UTILAR
                *
A100 20 01      START   BRA     STRT        FLEX CONVENSION
A102 02                 FCB     VERS
                *
A103            STRT    EQU     *
A103 CE A4 9A           LDX     #DBANK
A106 6F 00      CL100   CLR     X
A108 08                 INX
A109 8C A4 AE           CPX     #DBEND
A10C 26 F8              BNE     CL100
A10E BD A2 D0           JSR     SETUP       FIX PIA
```

```
A111 7F A4 9C  MAIN    CLR     TFLAG       TIME GIVEN = .FALSE.
A114 BD A2 6D  MA1000  JSR     NEXTCH
A117 26 03             BNE     *+5
A119 7E A1 AF          JMP     MA6000      NO PARAMETERS
A11C 81 2C             CMP A   #',         OPTIONS
A11E 27 32             BEQ     MA1100
A120 81 2B             CMP A   #'+
A122 27 3D             BEQ     MA1500
A124 CE A4 9F          LDX     #TIMBUF     INPUT TIME
A127 BD A2 78  MA1010  JSR     DIGCHK      CHECK IF 0-9
A12A 25 03             BCS     *+5
A12C 7E A2 42          JMP     ER1000      JUMP IF BAD
A12F A7 00             STA A   X
A131 08                INX
A132 8C A4 A5          CPX     #TIMBUF+6   ALL 6 DIGITS
A135 27 06             BEQ     MA1050
A137 BD A2 6D          JSR     NEXTCH
A13A 26 EB             BNE     MA1010
A13C 7E A2 5C          JMP     ER4000      6 DIGITS
                       *
A13F 7C A4 9C  MA1050  INC     TFLAG       TIME IS GIVEN
A142 BD A2 6D          JSR     NEXTCH      OPTIONS
A145 27 74             BEQ     MA6050      VALIDATE TIME
A147 81 2C             CMP A   #',         STILL HOPE FOR OPT
A149 27 07             BEQ     MA1100
A14B 81 20             CMP A   #$20        SP ALSO VALID SEPARATOR
A14D 27 03             BEQ     MA1100
A14F 7E A2 4C          JMP     ER2000
A152 BD A2 6D  MA1100  JSR     NEXTCH
A155 26 03             BNE     *+5
A157 7E A2 54          JMP     ER3000      SHOULD BE + HERE
A15A 81 2B             CMP A   #'+
A15C 27 03             BEQ     *+5
A15E 7E A2 4C          JMP     ER2000      DARN FOOL
                       *
A161 BD A2 6D  MA1500  JSR     NEXTCH      OPTION LETTER
A164 27 35             BEQ     MA5000      NO (MORE) OPTIONS
A166 81 48             CMP A   #'H         H(OLD)
A168 26 05             BNE     MA2000
A16A 7C A4 9B          INC     HFLAG       H .TRUE.
A16D 20 F2             BRA     MA1500
                       *
A16F 81 44     MA2000  CMP A   #'D         D(ATE)
A171 26 05             BNE     MA3000
A173 7C A4 9A          INC     DFLAG       .TRUE.
A176 20 E9             BRA     MA1500
                       *
A178 81 51     MA3000  CMP A   #'Q         Q(UERY)
A17A 26 05             BNE     MA3100
A17C 7C A4 9D          INC     QFLAG       .TRUE.
A17F 20 E0             BRA     MA1500
                       *
A181 81 43     MA3100  CMP A   #'C
A183 26 05             BNE     MA3200
```

```
A185 7C A4 9E            INC     CFLAG
A188 20 D7               BRA     MA1500      COMBINATIONS
                         *
A18A 81 4C      MA3200   CMP A   #'L         L OPTION
A18C 26 05               BNE     MA4000
A18E 7C A4 AD            INC     LFLAG
A191 20 CE               BRA     MA1500
                         *
A193 CE A3 8B   MA4000   LDX     #TEXT10     ILLEGAL OPT
A196 BD A3 32            JSR     INSERT      ERR-MSG
A199 20 C6               BRA     MA1500
                         *
A19B 7D A4 9D   MA5000   TST     QFLAG       QUERY FOR TIME
A19E 27 0F               BEQ     MA6000
A1A0 7F A4 9D            CLR     QFLAG       .FALSE.
A1A3 CE A3 A8            LDX     #TEXT20     TIME (TTMMSS)
A1A6 BD AD 1E            JSR     PSTRNG
A1A9 BD AD 1B            JSR     INBUFF
A1AC 7E A1 11            JMP     MAIN        TAKE NORMAL COURSE
                         *
A1AF 7D A4 9C   MA6000   TST     TFLAG       TIME GIVEN
A1B2 26 07               BNE     MA6050
A1B4 7D A4 9A            TST     DFLAG       DISPLAY DATE
A1B7 26 5A               BNE     MA7050
A1B9 20 6C               BRA     MA7100
                         *
                         * CHECK IF GIVEN TIME IS REASONABLE
                         *
                         * HOURS  : 0-23
                         * MIN    : 0-59
                         * SEC    : 0-59
                         *
A1BB CE A4 9F   MA6050   LDX     #TIMBUF
A1BE BD A2 7D            JSR     DAADIG      10+1 HOURS DIGITS
A1C1 81 24               CMP A   #$24
A1C3 25 05               BCS     MA6100
A1C5 CE A3 BE            LDX     #TEXT30     ILLEGAL HOUR SPEC
A1C8 20 1C               BRA     ERT100
                         *
A1CA B7 A4 A5   MA6100   STA A   CLBUFF
A1CD BD A2 7D            JSR     DAADIG      10+1 MINUTES
A1D0 81 5A               CMP A   #$5A
A1D2 25 05               BCS     MA6200
A1D4 CE A3 DA            LDX     #TEXT40     ILLEGAL MIN SPEC
A1D7 20 0D               BRA     ERT100
                         *
A1D9 B7 A4 A6   MA6200   STA A   CLBUFF+1
A1DC BD A2 7D            JSR     DAADIG      10+1 SEC
A1DF 81 5A               CMP A   #$5A
A1E1 25 09               BCS     MA6300
A1E3 CE A3 F8            LDX     #TEXT50     ILLEGAL SEC SPEC
A1E6 BD AD 1E   ERT100   JSR     PSTRNG
A1E9 7E A2 64            JMP     FATERR
                         *
```

```
A1EC B7 A4 A7  MA6300  STA A  CLBUFF+2
A1EF BD A2 E9          JSR    SETTIM
A1F2 86 04             LDA A  #HOLD
A1F4 B7 80 52          STA A  PIAURS     HOLD CLOCK
A1F7 7D A4 9B          TST    HFLAG      SHALL WE HOLD IT
A1FA 27 0D             BEQ    MA6800
A1FC CE A4 16          LDX    #TEXT60    PRESS RETURN..
A1FF BD AD 1E          JSR    PSTRNG
A202 BD AD 15  MA6750  JSR    GETCHR
A205 81 0D             CMP A  #$D        CR
A207 26 F9             BNE    MA6750
                       *
A209 86 00     MA6800  LDA A  #RUN
A20B B7 80 52          STA A  PIAURS     TICK..TICK
                       *
A20E 7D A4 9A  MA7000  TST     DFLAG      DATE
A211 27 2C             BEQ     MA8000
A213 7D A4 9E  MA7050  TST     CFLAG      IGNORE CR
A216 26 06             BNE     MA7060
A218 BD AD 24          JSR     PCRLF
A21B 7C A4 AC          INC     CRLFFG     CRLF IS PRINTED
A21E BD A3 4C  MA7060  JSR     DISDAT
A221 CE A4 97          LDX     #TEXTD0    2 SP
A224 BD A3 44          JSR     INS300
A227 7D A4 AC  MA7100  TST     CRLFFG
A22A 26 06             BNE     MA7120
A22C 7D A4 9E          TST     CFLAG
A22F 26 03             BNE     MA7120
A231 BD AD 24          JSR     PCRLF
A234 BD A2 A4  MA7120  JSR     DISTIM     DISPALY TIME
A237 7D A4 AD          TST     LFLAG      ISSUE CR+LF ?
A23A 27 03             BEQ     MA8000
A23C BD AD 24          JSR     PCRLF
A23F 7E AD 03  MA8000  JMP     WARMS
                       *
A242 8B 30     ER1000  ADD A  #$30        ASCII
A244 CE A4 4A          LDX     #TEXT90
A247 BD A3 32          JSR     INSERT
A24A 20 18             BRA     FATERR
                       *
A24C CE A4 6E  ER2000  LDX     #TEXTB0    SYNTAX ERR
A24F BD A3 32          JSR     INSERT
A252 20 10             BRA     FATERR
                       *
A254 CE A4 5C  ER3000  LDX     #TEXTA0
A257 BD AD 1E          JSR     PSTRNG
A25A 20 08             BRA     FATERR
                       *
A25C CE A4 82  ER4000  LDX     #TEXTC0    6 DIGITS
A25F BD AD 1E          JSR     PSTRNG
A262 20 00             BRA     FATERR
                       *
A264 CE A4 39  FATERR  LDX     #TEXT80
A267 BD AD 1E          JSR     PSTRNG
```

```
A26A 7E AD 03            JMP     WARMS
                         *
                         *
                         * SUB ( NEXTCH )
                         *
A26D                     NEXTCH  EQU     *
A26D BD AD 27                    JSR     NXTCH
A270 81 0D                       CMP A   #$0D
A272 27 03                       BEQ     NEXRET
A274 B1 AC 02                    CMP A   EOLCH       FLEX EOL
A277 39                  NEXRET  RTS
                         *
                         * SUB ( DIGCHK )
                         *
                         * CHECK IF DIGIT IS 0-9
                         *
                         * INPUT: A-ACC DIGIT ( ASCII )
                         * OUT:    CARRY=1 IF NO DIG
                         *
A278                     DIGCHK  EQU     *
A278 80 30                       SUB A   #$30        STRIP ASCII
A27A 81 0A                       CMP A   #$A         <9
A27C 39                          RTS
                         *
                         * SUB ( DAADIG )
                         *
                         * TAKE TWO BYTES FROM TIME BUFFER
                         * AND DAA ADJUST THEM TO ONE BYTE.
                         *
                         * INPUT : X-REG POINTS TO LOCATION
                         * OUTPUT: X-REG TWICE INCREMENTED
                         *         A-ACC TWO DEC DIGITS
                         *
A27D                     DAADIG  EQU     *
A27D A6 00                       LDA A   X
A27F 48                          ASL A
A280 48                          ASL A
A281 48                          ASL A
A282 48                          ASL A
A283 AA 01                       ORA A   1,X
A285 08                          INX
A286 08                          INX
A287 39                          RTS
                         *
                         * SUB ( DAATIM )
                         *
                         * FETCH TWO DIGITS FROM CLOCK.
                         *
                         * INPUT : X-REG PIONTS TO UNITS TABLE
                         * OUTP  : A-ACC TWO DIGITS ( IE. HOURS)
                         *
A288                     DAATIM  EQU     *
A288 E6 00                       LDA B   X           FETCH 10'S SELECTION CODE

A28A F7 80 50                    STA B   PIAURD
```

```
A28D 8D 14          BSR     DAARET      DELAY TIME
A28F B6 80 50       LDA A   PIAURD      10'S DIGIT
A292 48             ASL A
A293 48             ASL A
A294 48             ASL A
A295 48             ASL A
A296 E6 01          LDA B   1,X         1'S SELECTION CODE
A298 F7 80 50       STA B   PIAURD
A29B 8D 06          BSR     DAARET
A29D F6 80 50       LDA B   PIAURD
A2A0 C4 0F          AND B   #$0F        MASK
A2A2 1B             ABA
A2A3 39     DAARET  RTS
            *
            * SUB ( DISTIM )
            *
            * DISPLAY TIME FROM CLOCK AS: TT:MM:SS
            *
A2A4        DISTIM  EQU     *
A2A4 CE A2 CA       LDX     #HOURD      SELECTION CODE HOURS
A2A7 8D 15          BSR     DISSUB
A2A9 86 3A          LDA A   #':
A2AB BD AD 18       JSR     PUTCHR
A2AE CE A2 CC       LDX     #MIND       SELECTION CODE MINUTES
A2B1 8D 0B          BSR     DISSUB
A2B3 86 3A          LDA A   #':
A2B5 BD AD 18       JSR     PUTCHR
A2B8 CE A2 CE       LDX     #SECD       SELECTION CODE SECONDS
A2BB 8D 01          BSR     DISSUB
A2BD 39             RTS
            *
A2BE 8D C8  DISSUB  BSR     DAATIM
A2C0 B7 A4 AB       STA A   SLASK
A2C3 CE A4 AB       LDX     #SLASK
A2C6 BD AD 3C       JSR     OUTHEX
A2C9 39             RTS
            *
            * PIA SELECT CODES
            *
A2CA 00     HOURD   FCB     $00,$10     10,1 HOURS
A2CC 20     MIND    FCB     $20,$60     10,1 MINUTES
A2CE 50     SECD    FCB     $50,$40     10,1 SECONDS
            *
            * SUB ( SETUP )
            *
            * INITIALIZE THE CLOCK-PIA
            *
A2D0        SETUP   EQU     *
A2D0 7F 80 51       CLR     PIAURC
A2D3 7F 80 53       CLR     PIAURS+1    CONTROL
A2D6 86 F0          LDA A   #$F0        0-3 IN, 4-7 OUT
A2D8 B7 80 50       STA A   PIAURD
A2DB 86 07          LDA A   #7          B 0-7 OUT
A2DD B7 80 52       STA A   PIAURS
```

```
A2E0 86 04              LDA A  #4
A2E2 B7 80 51           STA A  PIAURC     CLOSE
A2E5 B7 80 53           STA A  PIAURS+1
A2E8 39                 RTS
                *
                * SUB ( SETTIM )
                *
                * SET THE CLOCK.
                *
                * INPUT: CLBUFF HOURS
                *        CLBUFF+1 MINUTES
                *        CLBUFF+2 SECONDS
                *
A2E9            SETTIM  EQU    *
A2E9 86 02              LDA A  #SLOW
A2EB B7 80 52           STA A  PIAURS
                *
                * SET SECONDS RIGHT FIRST
                *
A2EE CE A2 CE   SETT10  LDX    #SECD
A2F1 8D 95              BSR    DAATIM     FETCH TIME
A2F3 B1 A4 A7           CMP A  CLBUFF+2
A2F6 26 F6              BNE    SETT10
A2F8 86 01              LDA A  #FAST      SET MINUTES TO ZERO
A2FA B7 80 52           STA A  PIAURS
                *
                * IF HOUR = TIME THEN
                *     IF MIN <= TIME THEN
                *       SET MIN.
                * ELSE
                *     SET MIN = 0
                *     SET HOUR
                *     SET MIN
                * ENDIF
                *
A2FD CE A2 CA           LDX    #HOURD
A300 8D 86              BSR    DAATIM
A302 B1 A4 A5           CMP A  CLBUFF     EQUAL
A305 26 0B              BNE    SETT20
                *
A307 CE A2 CC           LDX    #MIND
A30A BD A2 88           JSR    DAATIM
A30D B1 A4 A6           CMP A  CLBUFF+1
A310 23 1A              BLS    SETT45
                *
A312 CE A2 CC   SETT20  LDX    #MIND      MINUTES SEL CODES
A315 BD A2 88           JSR    DAATIM
A318 4D                 TST A
A319 26 F7              BNE    SETT20
                *
A31B CE A2 CA   SETT30  LDX    #HOURD     SEL CODE
A31E BD A2 88           JSR    DAATIM
A321 B1 A4 A5           CMP A  CLBUFF
A324 26 F5              BNE    SETT30
```

```
                        *
A326 CE A2 CC   SETT40  LDX     #MIND       SEL CODE MIN
A329 BD A2 88           JSR     DAATIM
A32C B1 A4 A6   SETT45  CMP A   CLBUFF+1
A32F 26 F5              BNE     SETT40
                        *
A331 39                 RTS
                        *
                        * SUB ( INSERT )
                        *
                        * PRINT A MESSAGE WITH AN INSERT BETWEEN
                        * '()'.
                        *
                        * INPUT : X-REG STRING
                        *          A-ACC INSERT CHARACTER
                        *
A332                    INSERT  EQU     *
A332 36                 PSH A
A333 BD AD 24           JSR     PCRLF
A336 A6 00      INS100  LDA A   X
A338 BD AD 18           JSR     PUTCHR      PRINT
A33B 08                 INX
A33C 81 28              CMP A   #'(
A33E 26 F6              BNE     INS100
A340 32                 PUL A
A341 BD AD 18   INS200  JSR     PUTCHR
A344 A6 00      INS300  LDA A   X           NEXT CHAR
A346 08                 INX
A347 81 04              CMP A   #4          EOT
A349 26 F6              BNE     INS200
A34B 39                 RTS
                        *
                        *
                        * SUB ( DISDAT )
                        *
                        * PRINT FLEX DATE AS DAT: YY-MM-DD
                        *
                        * INPUT : DATERG   MM
                        *          DATERG+1 DD
                        *          DATERG+2 YY
                        *
A34C                    DISDAT  EQU     *
A34C CE A4 33           LDX     #TEXT70     DAT
A34F BD A3 44           JSR     INS300
A352 F6 AC 10           LDA B   DATERG+2    YEAR
A355 8D 2C              BSR     BINDAA      CONVERT
A357 B7 A4 A8           STA A   YEAR
                        *
A35A F6 AC 0E           LDA B   DATERG
A35D 8D 24              BSR     BINDAA
A35F B7 A4 A9           STA A   MONTH
                        *
A362 F6 AC 0F           LDA B   DATERG+1
A365 8D 1C              BSR     BINDAA
```

```
A367 B7 A4 AA            STA A  DAY
                    *
A36A CE A4 A8            LDX    #YEAR
A36D BD AD 3C            JSR    OUTHEX
A370 86 2D               LDA A  #'-
A372 BD AD 18            JSR    PUTCHR
A375 08                  INX
A376 BD AD 3C            JSR    OUTHEX
A379 86 2D               LDA A  #'-
A37B BD AD 18            JSR    PUTCHR
A37E 08                  INX
A37F BD AD 3C            JSR    OUTHEX
A382 39                  RTS
                    *
                    * SUB ( BINDAA )
                    *
A383                BINDAA  EQU    *
A383 4F                     CLR A
A384 8B 01          BINDA5  ADD A  #1
A386 19                     DAA
A387 5A                     DEC B
A388 26 FA                  BNE    BINDA5
A38A 39                     RTS
                    *
                    * T E X T  S T R I N G S
                    *
A38B 49             TEXT10  FCC    'Illegal option () specified.'
A3A7 04                     FCB    4
                    *
A3A8 47             TEXT20  FCC    'Give time (hhmmss) : '
A3BD 04                     FCB    4
                    *
A3BE 49             TEXT30  FCC    'Illegal hour specification.'
A3D9 04                     FCB    4
                    *
A3DA 49             TEXT40  FCC    'Illegal minute specification.'
A3F7 04                     FCB    4
                    *
A3F8 49             TEXT50  FCC    'Illegal second specification.'
A415 04                     FCB    4
                    *
A416 50             TEXT60  FCC    'Press RETURN to start clock.'
A432 04                     FCB    4
                    *
A433 44             TEXT70  FCC    'DAT: '
A438 04                     FCB    4
                    *
A439 43             TEXT80  FCC    'Command ignored.'
A449 04                     FCB    4
                    *
A44A 49             TEXT90  FCC    'Illegal digit ().'
A45B 04                     FCB    4
                    *
A45C 4F             TEXTA0  FCC    'Options expected.'
```

```
A46D 04                         FCB     4
                        *
A46E 53             TEXTB0      FCC     'Syntax error at ().'
A481 04                         FCB     4
                        *
A482 53             TEXTC0      FCC     'Six digits required.'
A496 04                         FCB     4
                        *
A497 20             TEXTD0      FCC     '  '
A499 04                         FCB     4
                        *
                        * LOCAL VARIABLES
                        * ---------------
                        *
A49A                DBANK       EQU     *
A49A                DFLAG       RMB     1       OPTION D
A49B                HFLAG       RMB     1       OPTION H
A49C                TFLAG       RMB     1       TIME GIVEN
A49D                QFLAG       RMB     1       OPTION Q
A49E                CFLAG       RMB     1       CRLF OPTION
A49F                TIMBUF      RMB     6       INPUT TIME
A4A5                CLBUFF      RMB     3       CLOCK TIME
A4A8                YEAR        RMB     1       YEAR IN DEC
A4A9                MONTH       RMB     1       MONTH IN DEC
A4AA                DAY         RMB     1       DAY
A4AB                SLASK       RMB     1       GARBAGE
A4AC                CRLFFG      RMB     1       CR PRINTED
A4AD                LFLAG       RMB     1       OPTION L
A4AE                DBEND       EQU     *
                        *
                                END     START
```

NO ERROR(S) DETECTED

1.UNDERTECKNAD ÖNSKAR KOMMA I KONTAKT MED NÅGON SOM HAR TID OCH INTRESSE AV ATT UPPRÄTTA OCH UNDERHÅLLA EN DATABAS FÖR TRAVHÄSTAR/KÖRSVENNER/RESULTAT.
INMATNING VIA 5-TUMS FLOPPY (TILL 1oMB WINCHESTER 480:-).
DATABASEN KAN ANVÄNDAS FÖR ENSKILT ELLER GEMENSAMT DELTAGANDE I VAS-SPEL.

ARTHUR WESSBLAD TELEFON 08/630200 ARB 08/611323 POST

2.FINNS NÅGON SOM ÄR INTRESSERAD AV ATT "KNAPPA IN" STATISTIKUNDERLAG MM FÖR DATREGISTER VIA 5-TUMS FLOPPY MED HJÄLP AV SPECIELLA INMATNINGSPROGRAM ? ERSÄTTNING EFTER ÖVERENSKOMMELSE.

ARTHUR WESSBLAD TELEFON 08/630200 ARB 08/611323/POST

FORTSATT SORTERING

Här kommer nu del två av den artikelserie om sortering som gått i Creative Computing. Författaren är Albert Nijenhuis. Shoot, Albert!..

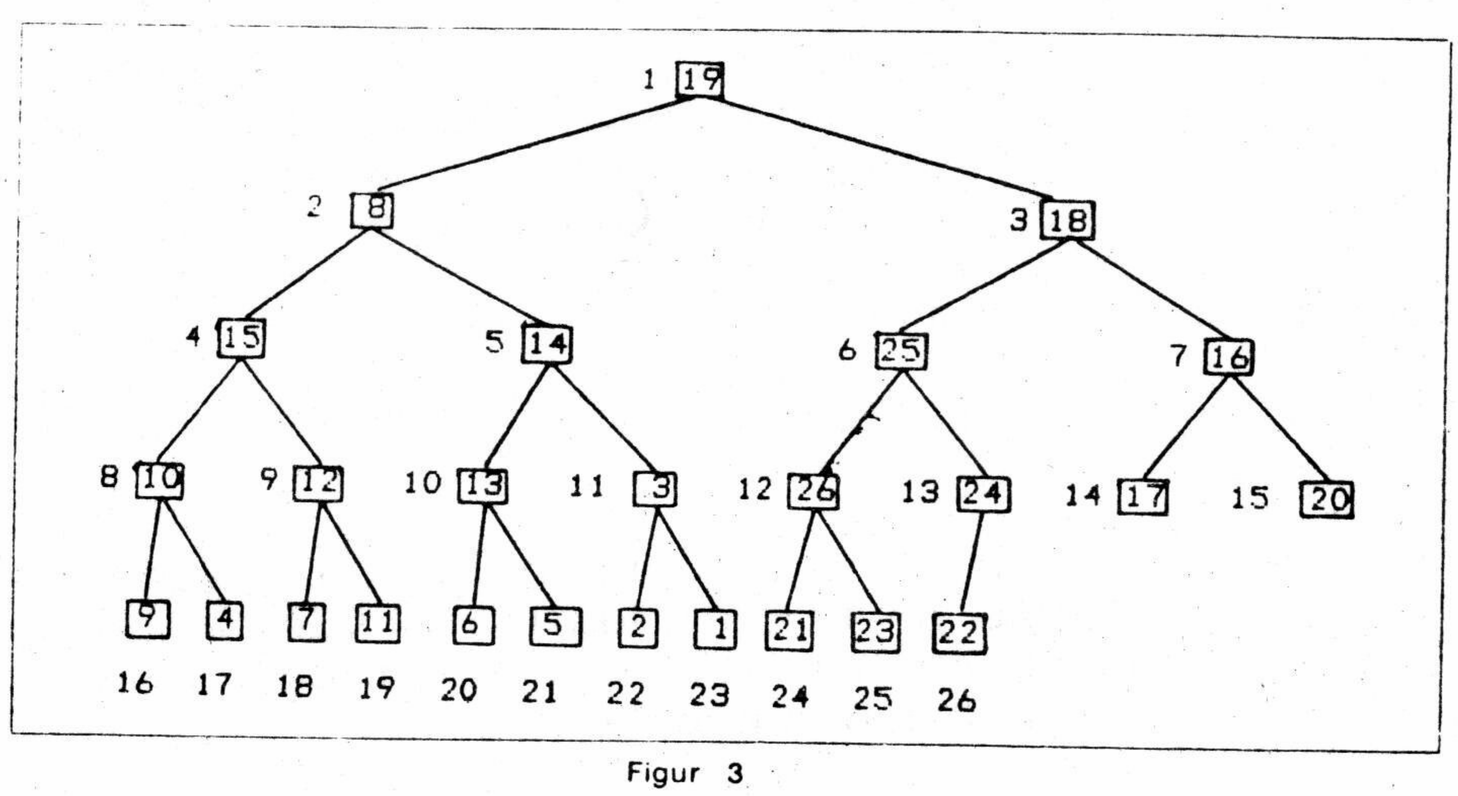
Den första delen i den här artikelserien behandlade "insertion sort" såsom en excellent metod för att sortera korta listor. I den här delen skall vi diskutera den enastående "Heapsort" metoden. Dess program är kort, inget extra lagringsutrymme behövs, men dess sätt är tilltalande smart, fastän vid första anblicken inte direkt enkelt. Den upptäcktes av Williams och Floyd 1964. Vår beskrivning följer nära den i "Combinatorial Algorithms" av författaren och H.S. Wilf (Acad. Press), vari läsaren också finner ett komplett program i Fortran.

Vad är en "Heap" ?

Det engelska ordet heap betyder hög, hop, samling. I fortsättningen använder vi orginalordet.

Ämnet för dagen är således heapsort. Vi åskådliggör därför en array a(1)...a(n) enligt figur 3. Det är den vi skall sortera. Första radens box motsvarar a(1), nästa rad blir a(2) och a(3) etc. Varje rad innehåller dubbelt så många array medlemmar som föregående. Ända tills de tar slut (n = 26 i fig 3).

Varje box ("fader") är ansluten till två boxar (hans "söner"), så länge som förrådet räcker. (Terminologin uppkom före Women's Lib). Figur 3 är ett exempel på ett så kallat binärträd. Dess rot är i box nr 1. Notera vidare att varje box i sin tur är roten till ett mindre binärträd. Exempelvis är box 5 roten till ett träd bestående av boxarna 5, 20, 21, 22, 23. Box 14 är däremot ett binärträd på egen hand.



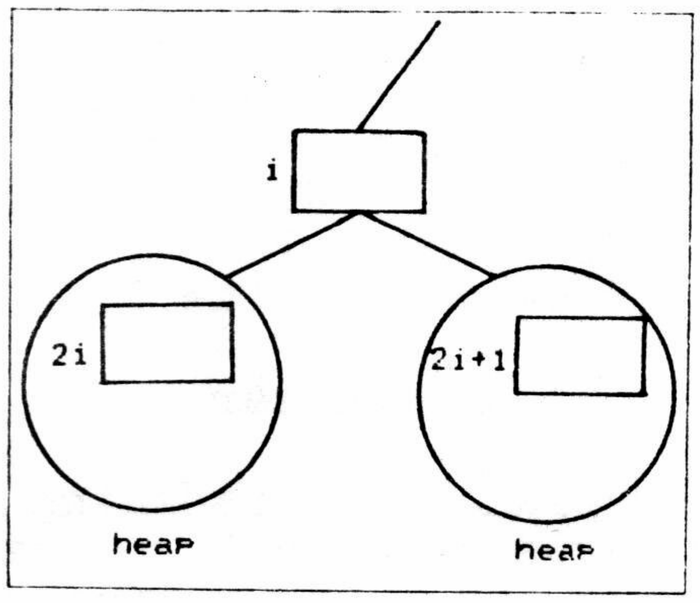
Figur 3

Det är lätt att bestämma strukturen hos ett binärträd med n boxar utan användandet av en bild: box i är fader till boxarna 2i och 2i+1. Detta gäller så länge dessa inte överskrider n (maxima).

Ett binärträd kallas en heap om elementen som lagras i boxarna har följande egenskaper: Värdet lagrat i varje fader är större än, eller lika med det i varje sons. Om 2i<n betyder detta att a(i)>=a(2i) och a(i)>=a(2i+1), medan om 2i=n betyder det endast det första av villkoren. Om 2i>n föreligger inte (son - mindre) villkoret för faderskapet i box i. Till exempel, i figur 3 är underträden rotade vid 4 och 5 heaps. Däremot är trädet rotat i 6 inte en heap.

Att skapa en Heap

Första steget i heapsort består i att omvandla binärträdet till en heap. Detta uppnår man genom en noggrant konstruerad sekvens av utbyten mellan far och son. I analogi med del 1, förutsätter vi redan delar av den önskade heap strukturen närvarande. Och vi skall utvidga den ända tills hela binärträdet är en heap. Om ett utbyte mellan en fader och en son övervägs, är binärträden rotade vid sönerna redan heaps. Initialt är alla underträd rotade vid "sonlösa" fäder heaps.



Figur 4

Låt oss anta att strategin pockar på ett möjligt utbyte vid fader i; se figur 4. Först jämför vi sönerna 2i och 2i+1 för att hitta den störste av dem. Låt j=2i eller j=2i+1 som fallet kan bli (om 2i=n, har vi alltid j=2i). Därefter jämförs a(i) och a(j). Om a(i)>=a(j), håller heap villkoret vid box i. Eftersom faderns värde är större eller lika med sönernas. Är så inte fallet utbyter vi a(i) och a(j). Därmed försäkrar vi oss om att det nya värdet uppfyller heap villkoret vid box i. Observera dock att ett heap villkor nu kanske inte längre finns för box j, eftersom det nya värdet är mindre än det gamla. Vi måste kanske justera situationen vid box j. Med tanke på det faktum att binärträden som är rotade vid box j är heaps, har vi samma problem igen. Fast i mindre skala. Det kan kanske vara svårt att tro, men denna enda operation av fader-son utbyten, är byggstenen i hela heapsort metoden.

```
1000 'SUBR: HEAPSORT OF  A(1),
     ...,A(N)

1010 'PHASE 1

1020 LET M=N

1030 FOR L=INT(N/2) TO  1 STEP -1

1040 LET B=A(L)

1050 GOSUB  1150

1060 NEXT L

1070 'PHASE 2

1080 LET L=1

1090 FOR M=N-1 TO 1 STEP -1

1100 LET B=A(M+1)

1110 LET A(M+1)=A(1)

1120 GOSUB 1150

1130 NEXT M

1140 RETURN

1150 'TOHEAP

1160 LET I=L

1170 LET J=I+I

1180 IF J>M GOTO 1250

1190 IF J=M GOTO 1210

1200 IF A(J+1)>A(J) THEN LET J=J+1

1210 IF B>=A(J) GOTO 1250

1220 LET A(I)=A(J)

1230 LET I=J

1240 GOTO 1170

1250 LET A(I)=B

1260 RETURN
```

Figur 5

Låt oss med referens till figur 3 göra en heap. För det första är binärträden rotade vid boxarna 13 till och med 26 redan heaps. Först gör vi en heap av subträdet rotat vid 13, därefter 12 osv. Efter detta fortsätter vi med träden rotade vid 7, 6 etc hela vägen upp till 1. I denna sista grupp kan naturligtvis ett byte vid roten leda till ytterligare byten nedanför.

Vi följer i detalj vad som händer vid box 6. Observera att binärträden rotade vid 12 och 13 är heaps. Nu är a(12) > a(13) så vi jämför a(6) och a(12). Detta medför ett utbyte. Jämför nu a(24) och a(25) för att finna den större. Dvs den nya a(12)=25 med a(25). Emedan den senare är mindre behövs inga ytterligare utbyten. I annat fall skulle vi bytt den nya a(12) och a(25).

Algoritm TOHEAP

Vi förutsätter att binärträdet med roten vid box L skall göras till en heap, medan binärträden rotade vid sönerna (om några) till box L är heaps. Den högst numrerade boxen får inte överskrida m (detta var n i sektion 6; vi inser senare varför).

Steg 1. (Initiera) Sätt b=a(L); sätt i=L1

Steg 2. (Leta rätt på en son om det finns någon). Sätt j=2i; om j>m gå till steg 5

Steg 3. (Om det finns en annan son, låt j vara den större). Om j=m, gå till steg 4, annars om a(j+1) > a(j) sätt j=j+1.

Steg 4. (Jämför b med a(j)). Om b<=a(j) gå till steg 5, annars (flytta a(j) uppåt) sätt a(i) = a(j), sedan gå till steg 2.

Steg 5. (Insätt b). Sätt a(i= = b; EXIT

Steg 1 (att skapa en heap) är utförd genom att genomlöpa TOHEAP med m=n, och med L gående baklänges från (n/2) till 1. (Vi låter (x) vara det högsta heltalet i x, t ex. (7,5) = 7, (6) = 6).

Heapsort, del 2

Ok - nu när vi har en heap, vad skall vi då göra? Vi ville ju sortera listan, eller hur? För vad vi får ut med heap är en trave, tämligen egendomligt organiserad. Med det största elementet på toppen. Det är inte ens klart vart det näst största elementet ligger. För att inte tala om det tredje största... Oaktat detta behövs bara ett par rader för skingra molnen.

Tag a(1) som är det största elementet och byt ut det med a(n). Nu är det största elementet där det skall vara (rör det aldrig igen!), och vi har ett binärträd som består av boxarna 1 till och med n-1. Detta är en heap utom vid roten. Ett anrop av TOHEAP (1,n-1) återupprättar heap strukturen. Det andra största elementet är nu a(1). Utbyt a(1) med a(n-1) och släpp lös TOHEAP igen. Osv..osv.. Det är allt!!

Fas 1. För L = (n/2) till 1 i steg -1 utför TOHEAP (L,n); next L

Fas 2. För m = n-1 till 1 i steg -1 utbyt a(m+1), a(1), utför TOHEAP (1,m), next m; EXIT

I figur 5 ger vi ett Basic program för heapsort. Subrutinen TOHEAP startar vid 1150. Den skiljer sig från algoritmen i sektion 7 bara i att b (=B) definieras strax före anropet. I instruktionerna 1040 och 1100. Loopen 1030-1060 är Fas 1, medan slingan 1090-1130 är Fas 2.

Slutkommentar

En noggran undersökning av TOHEAP visar att varje applicering av denna subrutin är en insertion (se MPU-larnen 4/80, Sort del 1). Den lista som denna insertion utförs på är dock extremt kort. Andra metoder, t ex. med merge-sort (MPU-laren 2/81) behöver ännu färre jämförelser. De kräver dock ett

annat pris, bl.a. extra arbetsutrymme.. Därutöver finns andra metoder som drar nytta av enventuell existerande ordning i indatat. Men med hänsyn till denna sista egenskap är till och med insertion sort överlägsen.

Detta om detta. Heapsort är en verkligt kvalificerad kandidat för snabb sortering, utan onödigt besvär, av listor på praktiskt taget vilken längd som helst. Speciellt då indata (vanligtvis) är i ett tillstånd av betydande oordning. (Nästa nummer: Merge-sort av länkade listor).

TBL

Dynasoft Pascal för 6800

Vi fick ett brev från Curt Söderberg som berättade om ett intressant program. Det rör sig om ett Pascal system för den lilla dataanläggningen. Där man inte har så våldsamt mycket minne och ingen disk enhet. Så här skriver Curt:

- Detta är en Pascal kompilator som genererar p-kod vilken exekveras av en intepretator skriven i 6800 kod. Systemet består förutom av intepretatorn av en supervisor, editor och kompilator. Den levereras på en kassett i KCS och S1 format. En välgjord manual medföljer.

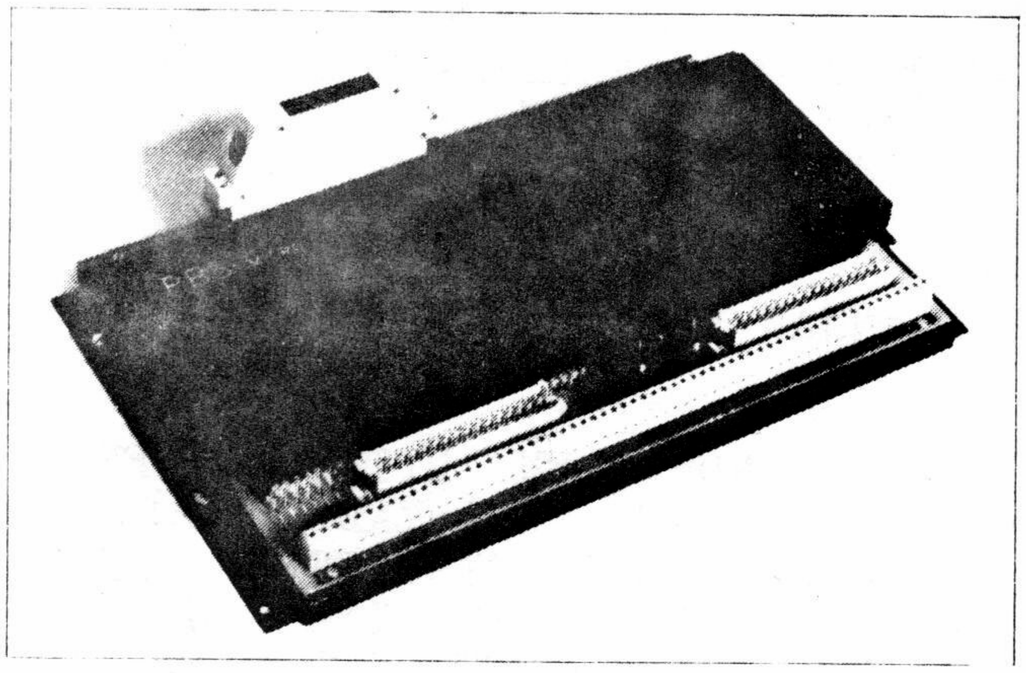
Av det egna systemet krävs minst 12k RAM från adress 0 och uppåt. Kompilatorn tar självt cirka 7k. För I/O hantering används rutiner i monitorn SWTBUG, men manualen talar om hur man ändrar till annan monitor. Då man skrivit kompilatorn för att användas på små system, har den inte en del av de mer avancerade datastrukturerna. Den har inte heller flyttal. Men den lämpar sig väl för den som vill lära sig Pascal. Programmet kan köpas från:

Dynasoft Systems
P.O. Box 51
Windsor Junction
Nova Scotia
CANADA BON 2V0

Priset är 38 dollar.

TB1

PBS-Micro har kommit — nu kan Du spola alla små 68system!



Med vår nya succé, PBS-Micro, får Du ett programmerbart binärt styrsystem i Micro-format, 360x210x40 mm, och till Micropris med följande prestanda:

- in/utgångar 32 resp. 16, 24 V DC 0.3 A, med lysdiodindikeringar
- kan utökas med hela tre st tillsatsenheter
- anslutning direkt på skruvplint eller kontaktdon
- programmet lagras i PROM alt. CMOS RAM med max. 2 k ord
- programspråk helt lika PBS-Mini*)

Kontakta närmaste AEG-kontor!

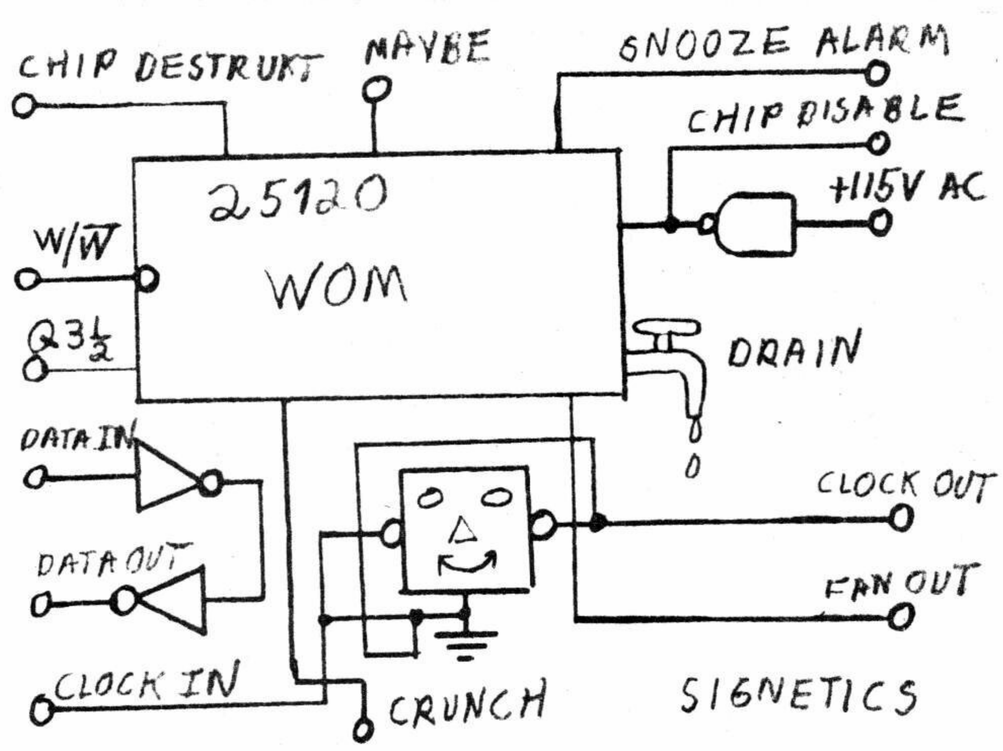
*) PBS från SATT-Elektronlund omfattar även Mini, C 08, Maxi och Jumbo — Du kan få just det styrsystem som Du behöver!

Elektriska Aktiebolaget AEG Division Anläggningsteknik Ett AEG-TELEFUNKENföretag
Stockholm 08-29 00 80 • Malmö 040-711 40 • Göteborg 031-80 00 20 • Karlstad 054-18 51 40 •
Norrköping 011-10 66 40 • Sundsvall 060-12 16 00 • Skellefteå 0910-362 10

AEG

NYTT SENSATIONELLT MINNE
9046 * N WOM

Med anledning av januari månads insänd har Signetics introducerat ett smått sensationellt halvledarminne av write-only typ. Minneskapaciteten är inte mindre än 9046*n bit och det har möjliggjorts tack vare den unika SEX-processen (enl fotnot på databladet: "en skopa av det ena och en skopa av det andra", som patenterades redan 1946). En speciell fördel med det nya minnet är att det är hel-fullt komatibelt med TTL (Trös Tämkt Logik) förutsatt att man har lämpliga anpassningskretsar. Konstruktionen är givetvis mer eller mindre SOS, och vad gäller ingångarna så är de skyddade med latex-överdrag - undantagna är givetvis PILL (plenty in less locations) -förpackade kretsar. Beträffande minnets hastighet så garanterar man 5011 ggr högre hastighet än vad som är möjligt, och detta tillsammans med en "refräsch tajm" på en kaffe- eller en lunchpaus ger intressanta aspecter på olika typer av applikationer. Har man bara en 6 fots fläkt, som placeras omkr. 15 cm från kåpan, så är kylnings-problemet löst. Signetics säger dock försiktigtvis att om minnet skulle klappa ihop så är det bara att friska på med mera luft. Utöver detta vill vi på MPU-laren rekommendera ett insående studium av det genialiska blockschemat. Och de som efter detta vill veta mera kan lämpligen rekvirera det utförliga databladet från den norska generalagenten.



PATCH UTAV TRACK/SECTOR UTILITY

SUNE JOHANSSON IFRÅN HÄSSLEHOLM HAR SÄNT OSS PÅ REDAKTIONEN ETT ANTAL TREVLIGA PROGRAM.
DET HÄR ÄR ETT PROGRAM FÖR DEN SOM VILL REPARERA PÅ DISKETTEN. DET FINNS VISSERLIGEN REPAIR SOM FLEX UTILITY MEN DET KAN VÄL VARA KUL ATT SE HUR DET HÄR ARBETAR.

PROGRAMMET FUNGERAR SÅ ATT DET FRÅGAR EFTER SPÅR SAMT SEKTOR OCH JÄMFÖR DETTA MED MAX ANTAL SPÅR (TRACKS) OCH SEKTORER SOM DISKETTEN INNEHÅLLER. OM DU ANGIVIT FEL SÅ KNORRAR PROGRAMMET.
SEDAN DUMPAR DET UT INNEHÅLLET PÅ SEKTORN I DET FORMATET SOM PASSAR SKÄRMEN. DEN TITTAR DÅ PÅ VILKEN WIDD-PARAMETER SOM DU HAR ANGIVIT I FLEX (TTYSET WD=XX).

DU KAN SEDAN ÄNRA EN BYTE PÅ EN VISS ANGIVEN OFFSET I SEKTORN. PROGRAMMET ÄR GJORT SÅ ATT DU KAN AVSLUTA NÄR SOM HELST GENOM ATT ANGE 'X'. PROGRAMMET ARBETAR ALLTID PÅ DIN ARBETS-DRIVE. OM DU VILL KÖRA DEN PÅ EN ANNAN DRIVE SKRIVER DU:
+++ASN W=X
+++PATCHTS
DÄR X=ANNAN DRIVE

EN VARNING DOCK!!!
DU BÖR EJ ÄNDRA EN BYTE PÅ EN SEKTOR UTAN ATT VETA VAD DU GÖR. ANNARS KAN HELA DISKETTEN BLI FÖRSTÖRD. EN LISTNING UTAV PROGRAMMET SAMT ETT KÖR-EXEPEL SER DU HÄR BREDVID. PROGRAMMET KAN FÅS OM DU SKICKAR IN EN DISKETT TILL SUNE ELLER MIG.

LU

```
                        NAM     PATCHTS
                        OPT     PAG
                *PATCH  AV EN BYTE I EN SPECIFICERAD TRACK O SEC
                *SUNE   JOHANSSON 1981.1.30
                * LENNART UUSITALO
                *
                *   FLEX-SYSTEM REFERENSER:
                *
AC04            WIDCNT  EQU     $AC04       TERMINAL BREDDS-PARAMETER
AC0C            WORKDR  EQU     $AC0C       ARBETS-DRIVE NR.
AC14            LIBUPO  EQU     $AC14       LINE BUFFER POINTER
AD03            WARMS   EQU     $AD03       WARMSTART ENTRY POINT
AD15            GETCHR  EQU     $AD15       GET CHARACTER
AD18            PUTCHR  EQU     $AD18       PUT CHARACTER
AD1B            INBUFF  EQU     $AD1B       INPUT INTO LINE BUFFER
AD1E            PSTRNG  EQU     $AD1E       PRINT STRING
AD21            CLASS   EQU     $AD21       CLASSIFY CHARACTER
AD24            PCRLF   EQU     $AD24       PRINT CR,LF
AD36            ADDBX   EQU     $AD36       ADD B-REG TO X-REG
AD3C            OUTHEX  EQU     $AD3C       OUT 1 BYTE I HEX = 2 ASCII
AD3F            RPTERR  EQU     $AD3F       REPORT ERROR
AD42            GETHEX  EQU     $AD42       GET HEXADECIMAL NUMBER
B406            FMS     EQU     $B406       FMS CALL

A840            FCB     EQU     $A840       FCB BASE ADRESS


A100                    ORG     $A100
A100 20 39      SECLI   BRA     START2
A102 03         VN      FCB     3
A103 00         TRACK   FCB     0
A104 00         SECTOR  FCB     0
A105            BNRDAT  RMB     1           DATA I EN VISS BYTE
A106            BYTNRP  RMB     2           BYTE POINTER
A108            TEMPX   RMB     2
A10A            BYTENR  RMB     1           BYTENUMMER I SEKTORN
A10B            WIDD    RMB     1           TERMINAL BREDD 8/16
A10C            ADRESS  RMB     1
A10D 22         MAXTRA  FCB     $22
A10E 0A         MAXSEC  FCB     $A
                *
                * TAG IN EN BYTE -SUBRUTIN
                *
                *
A10F BD AD 1E   UTPINP  JSR     PSTRNG
A112 BD AD 1B   INPBYT  JSR     INBUFF
A115 FE AC 14           LDX     LIBUPO
A118 A6 00              LDA A   X
A11A 81 58              CMP A   #'X'        EXIT
A11C 27 1A              BEQ     EXWAR1
A11E BD AD 42           JSR     GETHEX
A121 25 10              BCS     MARKF
A123 5D                 TST B
A124 27 0D              BEQ     MARKF
```

```
A126 FF A1 00            STX     TEMPX
A129 7D A1 00            TST     TEMPX
A12C 26 05               BNE     MARKF
A12E B6 A1 09            LDA A   TEMPX+1
A131 0C                  CLC
A132 39                  RTS
A133 0D          MARKF   SEC
A134 39                  RTS

A135 BD AD 3F    ERROR1  JSR     RPTERR
A138 7E AD 03    EXWAR1  JMP     WARMS

A13B C6 10       START2  LDA B   #16
A13D B6 AC 04            LDA A   WIDCNT      TESTA OM BRED ELLER SMAL DUMP
A140 27 06               BEQ     BREDD
A142 81 40               CMP A   #64
A144 24 02               BHS     BREDD
A146 C6 08               LDA B   #$08
A148 F7 A1 0B    BREDD   STA B   WIDD

A14B CE A8 40            LDX     #FCB       TAG REDA Pa MAX DISK-ADRESS
A14E B6 AC 0C            LDA A   WORKDR
A151 A7 03               STA A   3,X
A153 86 03               LDA A   #3          MARKERA SECTOR 0.3
A155 6F 1E               CLR     30,X
A157 A7 1F               STA A   31,X
A159 86 09               LDA A   #9
A15B A7 00               STA A   0,X
A15D BD B4 06            JSR     FMS
A160 A6 66               LDA A   102,X
A162 B7 A1 0D            STA A   MAXTRA
A165 A6 67               LDA A   103,X
A167 B7 A1 0E            STA A   MAXSEC

A16A CE A2 C9    SECLI1  LDX     #VILTRA    VILKET SPAR
A16D 8D A0               BSR     UTPINP
A16F 25 F9               BCS     SECLI1
A171 B1 A1 0D            CMP A   MAXTRA
A174 22 F4               BHI     SECLI1
A176 B7 A1 03            STA A   TRACK

A179 CE A2 F3    SECLI2  LDX     #VILSEC    VILKEN SEKTOR ?
A17C 8D 91               BSR     UTPINP
A17E 25 F9               BCS     SECLI2
A180 B1 A1 0E            CMP A   MAXSEC
A183 22 F4               BHI     SECLI2
A185 B7 A1 04            STA A   SECTOR

A188 CE A8 40            LDX     #FCB       LAS AKTUELL SEKTOR
A18B B6 AC 0C            LDA A   WORKDR
A18E A7 03               STA A   3,X
A190 B6 A1 03            LDA A   TRACK
A193 A7 1E               STA A   30,X
A195 B6 A1 04            LDA A   SECTOR
```

```
A198 A7 1F          STA A   31,X
A19A 86 09          LDA A   #9
A19C A7 00          STA A   X
A19E BD B4 06       JSR     FMS
A1A1 26 92          BNE     ERROR1

A1A3 BD AD 24 PRINT1 JSR    PCRLF       SKRIV UT DEN
A1A6 CE A3 41       LDX     #TRA
A1A9 BD AD 1E       JSR     PSTRNG
A1AC B6 A1 03       LDA A   TRACK
A1AF CE A1 03       LDX     #TRACK
A1B2 BD AD 3C       JSR     OUTHEX
A1B5 CE A3 48       LDX     #SEC
A1B8 BD AD 1E       JSR     PSTRNG
A1BB CE A1 04       LDX     #SECTOR
A1BE BD AD 3C       JSR     OUTHEX

A1C1 BD AD 24       JSR     PCRLF       SKRIV UT HUVUD
A1C4 CE A2 69       LDX     #HUVUD1
A1C7 B6 A1 0B       LDA A   WIDD
A1CA 81 08          CMP A   #8
A1CC 22 03          BHI     WRITH
A1CE CE A2 A5       LDX     #HUVUD2
A1D1 BD AD 1E WRITH JSR     PSTRNG

A1D4 BD AD 24       JSR     PCRLF       PRINTA UT SECTORN
A1D7 CE A8 80       LDX     #FCB+$40    PEKA PÅ DATA
A1DA 7F A1 0C       CLR     ADRESS      NOLLA ADRESS RÄKNARE
A1DD BD A2 35 PRINT4 JSR    PRINT5      SKRIV UT EN RAD
A1E0 7D A1 0C       TST     ADRESS
A1E3 26 F8          BNE     PRINT4

A1E5 CE A3 1D BYTE2 LDX     #VILBYT     VILKEN BYTE
A1E8 BD A1 0F       JSR     UTPINP
A1EB 25 F8          BCS     BYTE2
A1ED B7 A1 0A       STA A   BYTENR

A1F0 CE A8 80       LDX     #FCB+$40    TAG REDA PÅ POINTER
A1F3 16             TAB
A1F4 BD AD 36       JSR     ADDBX
A1F7 FF A1 06       STX     BYTNRP      SPARA POINTERN T.V.
A1FA A6 00          LDA A   X           HÄMTA DATA BYTE
A1FC B7 A1 05       STA A   BNRDAT
A1FF BD AD 24 BYTE3 JSR     PCRLF
A202 CE A1 05       LDX     #BNRDAT
A205 BD AD 3C       JSR     OUTHEX
A208 8D 26          BSR     OUTS
A20A 86 2F          LDA A   #'/
A20C BD AD 18       JSR     PUTCHR
A20F 8D 1F          BSR     OUTS
A211 BD A1 12       JSR     INPBYT
A214 25 E9          BCS     BYTE3
A216 FE A1 06       LDX     BYTNRP
A219 A7 00          STA A   X           SKRIV IN DET NYA VÄRDET
```

```
A21B CE A8 40          LDX    #FCB
A21E 86 0A             LDA A  #10        LAGRA DET PA SKIVAN
A220 A7 00             STA A  X
A222 BD B4 06          JSR    FMS
A225 26 03             BNE    ERR011
A227 7E A1 6A          JMP    SECLI1
A22A 7E A1 35 ERR011   JMP    ERROR1

A22D BD AD 3C OUT2HS   JSR    OUTHEX     SKRIV UT 2 HEX+ SPACE
A230 86 20    OUTS     LDA A  #' 
A232 7E AD 18          JMP    PUTCHR
                       *
                       * SUBR. FöR ATT SKRIVA UT EN RAD
                       *
A235 FF A1 08 PRINT5   STX    TEMPX
A238 CE A1 0C          LDX    #ADRESS    SKRIV UT ADRESSEN
A23B 8D F0             BSR    OUT2HS
A23D 8D F1             BSR    OUTS

A23F FE A1 08          LDX    TEMPX      SKRIV UT DATA
A242 F6 A1 0B          LDA B  WIDD
A245 8D E6    PRINT6   BSR    OUT2HS
A247 08                INX
A248 7C A1 0C          INC    ADRESS
A24B 27 03             BEQ    PASCII
A24D 5A                DEC B
A24E 26 F5             BNE    PRINT6

A250 FE A1 08 PASCII   LDX    TEMPX      SKRIV UT ASCII
A253 F6 A1 0B          LDA B  WIDD
A256 A6 00    PRASC    LDA A  X
A258 BD AD 21          JSR    CLASS
A25B 24 02             BCC    PRASC1
A25D 86 2E             LDA A  #'.
A25F BD AD 18 PRASC1   JSR    PUTCHR
A262 08                INX
A263 5A                DEC B
A264 26 F0             BNE    PRASC
A266 7E AD 24          JMP    PCRLF      LF+RETURN

A269 41       HUVUD1   FCC    'ADR 0  1  2  3  4  5  6  7  8  9  A  B  C
                              D  E  F    ASCII'
A2A4 04                FCB    4
A2A5 41       HUVUD2   FCC    'ADR 0  1  2  3  4  5  6  7    ASCII'
A2C8 04                FCB    4
A2C9 56       VILTRA   FCC    'VILKEN TRACK, HEX, (0-22), EL EXIT (X) ? '
A2F2 04                FCB    4
A2F3 56       VILSEC   FCC    'VILKEN SECTOR, HEX, (0-A), EL EXIT (X) ? '
A31C 04                FCB    4
A31D 56       VILBYT   FCC    'VILKET BYTE,HEX , ELLER EXIT (X) ? '
A340 04                FCB    4
A341 54       TRA      FCC    'TRACK '
A347 04                FCB    4
```

```
PATCHTS                                  1-12-81  TSC ASSEMBLER  PAGE     4

 A348 53          SEC      FCC     'SECTOR '
 A34F 04                   FCB     4

                           END     SECLI

NO ERROR(S) DETECTED

+++PATCHTS

VILKEN SECTOR, HEX, (0-A), EL EXIT (X) ? 0A

TRACK 22
SECTOR 0A

ADR  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    ASCII
00  02 02 00 09 09 02 58 0D 09 08 4A 53 52 09 03 46  ......X...JSR..F
10  4D 53 0D 09 08 42 45 51 09 03 50 52 49 4E 54 31  MS...BEQ..PRINT1
20  0D 09 08 4A 4D 50 09 03 45 52 52 4F 52 31 0D 0D  ...JMP..ERROR1..
30  50 52 49 4E 54 31 09 02 4A 53 52 09 03 50 43 52  PRINT1..JSR..PCR
40  4C 46 09 04 53 4B 52 49 56 20 55 54 20 44 45 4E  LF..SKRIV.UT.DEN
50  0D 09 08 4C 44 58 09 03 23 54 52 41 0D 09 08 4A  ...LDX...TRA...J
60  53 52 09 03 50 53 54 52 4E 47 0D 09 08 4C 44 41  SR..PSTRNG...LDA
70  41 09 02 54 52 41 43 4B 0D 09 08 4C 44 58 09 03  A..TRACK...LDX..
80  23 54 52 41 43 4B 0D 09 08 4A 53 52 09 03 4F 55  .TRACK...JSR..OU
90  54 48 45 58 0D 09 08 4C 44 58 09 03 23 53 45 43  THEX...LDX...SEC
A0  0D 09 08 4A 53 52 09 03 50 53 54 52 4E 47 0D 09  ...JSR..PSTRNG..
B0  08 4C 44 58 09 03 23 53 45 43 54 4F 52 0D 09 08  .LDX...SECTOR...
C0  4A 53 52 09 03 4F 55 54 48 45 58 0D 0D 09 08 4A  JSR..OUTHEX....J
D0  53 52 09 03 50 43 52 4C 46 20 53 4B 52 49 56 20  SR..PCRLF.SKRIV.
E0  55 54 20 48 55 56 55 44 0D 09 08 4C 44 58 09 03  UT.HUVUD...LDX..
F0  23 48 55 56 55 44 31 0D 09 08 4C 44 41 41 09 02  .HUVUD1...LDAA..

VILKET BYTE,HEX , ELLER EXIT (X) ? FF
02 / FF

VILKEN TRACK, HEX, (0-22), EL EXIT (X) ? X
+++
```

RT-dator (D2).
16k RAM,32k RAM utan minnen.
18k EPROM med 11k mont.
Kassett interface.
Int. terminal,tangentbord.
BASIC & PASCAL.

Curt Söderberg
TEL. 08/7654667

SÄLJES

1 st 8k Statiskt RAM-
kort för MOTOROLA-bussen.
Hänvändelse till:
Carl Axel Bruno
08/ 754 37 88

DISASSEMBLERS

Har du kanske någon gång råkat ut för det här problemet: Du har skaffat ett program som verkar intressant. Men det är bara det att det inte går att köra på din maskin. Därför att din I/O ligger på andra adresser. Eller programmet använder minne där du inte har något. Kanske vill du fixa om någon utskrift, lägga till några extra "kromlister" e dyl.

När du suttit och rivit i en hex-dump några timmar, när du försökt hitta alla referenser till deras OUTEEE rutin, börjar du kanske fantisera, ja rent av drömma. Drömma om ett program som traskar igenom bytarna ett par varv. Och som ger ifrån sig en läsbar kodlista. En lista med symboliska lägesnamn, med operationskoderna omvandlade till mnemoniks. Med FCB, FDB, FCC, EQU och allt som de vitrockade grabbarna/tjejerna fyller sina listor med bara för att jäklas (Att alla programmerare har vita rockar och tjocka brillor har ju TV lärt oss. Så det måste ju vara sant!). I ett fjärran töcken ser du en komplett textfil som du editerar in dina ändringar i, kastar in i assemblerns käftar och därefter exekverar... Men, suckar du skeptiskt, sådana godsaker finns ju inte. Eller... gör de det?

Javisst. Sådana program finns faktiskt. De kallas med ett gemensamt namn för disassemblers. De här programmen finns i olika duktighetsgrad. Från det enklaste utförandet som klarar att skriva ut operationskoder i mnemoniks (dvs 86 blir LDAA) men med operanden i absolut form, till disaassemblers som jobbar i flera pass. Som har fördefinierade symbol/adress referenser lagrade i externa filer. Som klarar datafält, strängar, adresstabeller etc.

LABDIS och DYNAMITE

Två program som jag känner till och som jobbar under FLEX, heter LABDIS respektive DYNAMITE. LABDIS är en gamling medan DYNAMITE är alldeles ny. LABDIS klarar 6800 kod medan "pang-pang" klarar både 6800 och 6809 kod.

Ingen disassembler kan göra ett alltigenom perfekt jobb. Det går helt enkelt inte att återskapa orginaltextfilen inklusive kommentarer, precis som den ursprunglige programmeraren en gång gjorde den. Frågan är bara hur bra slutresultatet ut från disassemblern skall bli, och hur mycket möda man själv måste bidra med. I slutändan kan vi ta till editorn för att fixa till kvarvarande oegentligheter.

Här kan man se en skillnad i hur de båda programmen är tänkt att användas. I princip är det så att LABDIS kör man en gång, sen är det färdigt. Strukturen i den genererade koden kan inte påverkas. Med DYNAMITE är det annorlunda. Man kan köra den med speciellt option för att upptäcka datablock, strängar odyl. Sedan kan man köra den igen och tala om för disassemblern att adressområdet x-y skall behandlas som ASCII data, ett annat som byte data osv. Tack vare denna samverkan mellan människan (Du!) och programmet, blir resultatet mycket bättre än om disassemblern på egen hand måste försöka avgöra vad som är data och vad som är programkod.

Låt oss ta en titt i manualerna och se vad som utlovas. LABDIS manual motsvarar c:a 4 A4 sidor, medan DYNAMITE kommer med c:a 19 A4 sidor. Detta kanske förklarar varför genomgången av den senare kommer att bli något fylligare?

LABDIS

LABDIS klarar alltså enbart 6800 kod. Programmet har följande egenskaper:

1. Assembler direktiv såsom ORG, NAM och END sätts in där det behövs.
2. Transfer adressen (om sådan

finns) ges ett läge och matas ut efter END direktivet.
3. Data block detekteras och reduceras till antingen FCC (i fall av 3 eller flera konsekutiva alfanumeriska tecken), eller FCB direktiv.
4. Utmatning kan ske antingen mot systemets terminal eller till en diskfil.
5. En diskfil kan disassembleras mellan specifika adressgränser.
6. Användaren kan välja mellan olika utformat.
7. En speciell diskfil som innehåller parvisa adresser och symbolnamn kan anropas om så önskas. Om någon av dessa adresser refereras i det disassemblerade programmet, kommer den motsvarande symbolen att insättas i den utmatade texten för att förbättra läsbarheten.

Options för styrning

För styrning av disassembleringens förlopp kan man ge att antal sk. options. Dessa består av enkla bokstäver. Följande möjligheter finns:
C. Denna komprimerar alla tabulatorer till ett mellanslag. Sparar buffertutrymme vid en senare editering.
E. Expandera alla lägen (labels) med ledande nollor. Det betyder att i stället för defaultmodens 'L127' kommer 'L00127' att matas ut. Därigenom blir varje label unikt identifierbar för en editor.
H. Denna option instruerar disassemblern att mata ut de hexadecimala byteparen som utgör en instruktion och dess operand (om någon) före lägesfältet i uttexten. Får resultatet att likna en assemblerlista.
K. Om assemblern inte lyckas hitta en referens i EQUATES.DSM filen, genererar den ett eget namn. Exempelvis L127. Med K option får man möjlighet att i stället besvara olösta referenser från terminalen. Mycket användbart om man har tillgång till en assemblerlista men inte en källkodsfil. Med listan i ena nypan och den andra handen på tangenbordet, kan man få samma namn i den disassemblerade texten som i orginalet.
N. Mata ut adressen på instruktionens byte vid varje rads början. Bra tillsammans med H option.
P. Paginera uttexten. Printar källfilens namn, en heading och sidnummer överst på varje sida.
S. Mata ut symbolerna efter disassembleringens slut. Sorterade i adressordning. Alla egna definierade EQU värden kommer också med.

EQUATES.DSM filens uppbyggnad

Man kan som tidigare sagts skapa en fil som innehåller de adresser och namn som man känner till. Exempelvis samtliga FLEX referenser. Varje rad i textfilen, vars namn måste vara EQUATES.DSM har följande utseende:
XXXX LABEL (CR)
där XXXX är en fyrställig hexadecimal adress med (ev) inledande nollor. 'LABEL' är en sträng på upp till sex alfanumeriska tecken. Exempel:
AD03 WARMS

I figur 3 ser vi hur LABDIS behandlar det testexempel som medföljde DYNAMITE disketten.

DYNAMITE

I manualen för DYNAMITE beskriver man programmets egenskaper så här:
1. Körs under FLEX 9.0
2. Kan disassemblera både 6800 och 6809 kod.
3. Producerar källkod som när den ånyo assembleras blir identisk med orginalets binärfil.
4. Lägen genereras automatiskt för alla minnesreferenser inom ett program. Lägets namn visar minnesadressen där det ligger. Exempelvis L65A0. Den labeln låg i programmet på adress $65A0.
5. Varje läge kan ges ett standardnamn som används i

stället för ett genererat namn. I default konfigurationen kommer referenser till standard FLEX adresser att få sina vanliga namn, exempelvis 'WARMS'. Man kan definiera egna standardnamn, eller använda olika lägen i stället för default tilldelningen i speciella program.

6. Vilket som helst block i ett program kan behandlas som data. En sådan area kan definieras som varande någon av följande varianter:

- ASCII data områden, vilka producerar FCC direktiv. Strängar avgränsas automatiskt och ASCII kontrolltecken döps till sina standardnamn.
- Byte data områden producerar FCB:n med hexadecimala data bytes.
- Label data områden producerar FDB:n med lägesreferenser till program. Många program har datatabeller som används för indirekt adressering eller för inhopp i programmet. Sådana tabeller disassembleras bäst som en serie av lägen.
- Word data områden producerar FDB:n med hexadecimala dubbla bytes.

7. Man kan specificera gränserna för dataområdena inom ett program på två sätt. Data adresserna kan matas in interaktivt i början på disassembleringen, eller läsas från en preparerad textfil.

8. Utmatningen från DYNAMITE kan styras mot utskriftsenheten eller till en diskfil.

9. Listan kan pagineras och radnumreras.

10. För att underlätta sökningen av dataareor i ett program kan den översatta koden även printas som ASCII tecken.

Filanvändning

DYNAMITE använder ett flertal filer när den jobbar. Vi har först naturligtvis INPUT filen med den object kod som skall disassembleras. OUTPUT filen (optionell) kommer att innehålla den disassemblerade koden. Sedan kan vi ha en COMMAND fil innehållande textrader med specifikationer över adressområden och datatyper inom programmet. Däri kan egna lägesfiler anges, liksom de options man även kan ge via FLEX kommandorad. Slutligen finns LABEL filen som i DYNAMITE är en binär fil. I den kan sambandet ges mellan minnesadresser och symboliska namn. DYNAMITE använder som default filerna DISLBL09.BIN för 6809 kod och DISLBL00.BIN för 6800 kod. Man kan specificera en egen lägesfil för användning jämsides med defaultfilerna. Eller i stället för dessa.

Options för DYNAMITE

Även DYNAMITE har en hel rad med options som man kan styra förloppet med:

A. Printa de ASCII tecken som programkoden motsvarar på källraden. Dessa tecken skrivs mellan kodens hexadecimala bytes och lägesfältet.

B. Promta (fråga) användaren efter dataområdenas gränser. Optionet får disassemblern att gå in i en interaktiv mod.

D. Skapa inte en textfil på disken. Inte ens om ett filnamn har givits.

G. Generera inga extra rader med kodbytes när mer än fyra bytes finns i en FCC, FCB eller FDB. Sparar papper.

L. Skicka inte en lista till utmatningsenheten.

N. Generera radnummer i listan.

P. Slå på paginering. Man blir tillfrågad om en titel vilken skall skrivas överst på varje sida. Ger även formatering rader/sida, heading, blankrader, 54 kodrader samt form feed vid sidslut.

S. Promta för ett segment som skall disassembleras. Gör att man kan disassemblera portioner av binärfilen. Programmet frågar efter adressgränser. Bra om den resulterande textfilen skulle bli så stor att den inte ryms på en disk.

Z. Disassemblera programmet för 6800 och inte 6809 som är

default.
0-3. Tala om vilken drive som default lägesfilerna finns på (DISLBL00 eller DISLBL09). Normalt hämtas de från samma drive som DYNAMITE laddades från.

Kommando filen

Vi skall inte gå in i detalj hur kommandofilen byggs upp. Utan låt oss se på ett exempel:

```
+ZG
* Data area inom koden
B A102-A104
W A105-A108
L A233-A240
A A241-A27F
* Egen referens fil
T LBLFIL.BIN
```

Första kommandoraden deklarerar att programmet som skall disassembleras är 6800 kod. Vidare skall extra kodrader för FCC etc undertryckas. Nästa rad som startar med en asterisk är endast en kommentarrad. Rad tre till och med sex deklarerar fyra dataområden i programmet. En av varje typ. Kod från A102 till och med A104 skall disassembleras som bytes med FCB direktiv. Nästa rad ger två stycken FDB:n. Mellan A233-A240 skall sju lägesreferenser genereras i form av FDB:n. Månntro en adresstabell? Slutligen skall A241-A27F disassembleras som ASCII data. I strängar eller som konstanter. Rad sju är åter en kommentar. Åttonde raden specificerar en egen labelfil att användas tillsammans med default filen.

Labelfilen

En labelfil är av typen .BIN. Den editeras och assembleras med de vanliga FLEX programmen. Exempel på utseende:

```
 FCC 'VN    '
 FDB $C102
 FCC 'WARMS '
 FDB $CD03
```

Varje namn skall vara sex tecken långt. Eventuellt får blanka tillskjutas. På disken som DYNAMITE kom på finns flera färdiga lägesfiler i textformat. Där finns:

```
DISLBL09.TXT 6809 FLEX ref
DISLBL00.TXT 6800 FLEX ref
MFLEXLBL.TXT 6800 MiniFlex
SWTBGLBL.TXT SWTBUG referenser
SBUGLBL .TXT S-BUG (6809) ref
```

Hur arbetar DYNAMITE

Ja vanligtvis flyger den i luften... Skämt åsido. Det här är en tre pass disassembler. Under första passet söker programmet efter minnesreferenser som skall bli lägen. Även immediate instruktioner får lägesreferenser. Detta säkrar att vi inte missar några lägen. Olyckligtvis får vi fall där detta är felaktigt. Tag fallet då vi sätter X-registret med en konstant för att ha som loopräknare. Då kommer ett läge som motsvarar en minnesreferens att skapas. Vars adress är konstanten. Detta får man fixa i efterhand om koden skall relokeras. Ett undantag är instruktion av type ADDD #. Eftersom denna nästan enbart används för datamanipulation (ex ADDD #1), får programmet känna till detta. Ingen label där således.

Pass nr två gör inget disassembleringsförsök. Man flaggar bara samtliga lägen funna i pass ett som antingen interna eller externa. Tack vare detta kan de externa läggas ut som EQU direktiv i början.

I tredje passet skapar DYNAMITE den verkliga disassemblerade koden. Både diskfil och utmatningslistor. Utmatningen består av fyra huvuddelar. Först grupperar programmet samman alla standardlägen som hämtats ur lägesfilerna. Därefter grupperas alla ASCII EQU satser samman. Exempelvis kontrolltecken som BELL EQU 7. Som tredje grupp kommer externa referenser utan standardnamn. Till slut kommer som fjärde avdelning själva programmet i form av återgenererade källkodsrader.

Var kan man köpa't

LABDIS kan man köpa från:
Swedish Electronic hb
Box 2065
750 02 Uppsala
tel 018-10 01 90

DYNAMITE köptes från:
Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
USA

Programet kostar $60. När man har det i handen (tull moms mm), har kalaset gått på c:a 300 Skr.

TBl

```
1                       * DISASSEMBLY BY DYNAMITE OF 0.TESTFILE.BIN
2
3
4                       * STANDARD PRE-NAMED LABEL EQUATES
5
6              AD03  WARMS   EQU     $AD03
7              AD1E  PSTRNG  EQU     $AD1E
8
9                       * EXTERNAL LABEL EQUATES
10
11             A16E  LA16E   EQU     $A16E
12
13  A100                      ORG     $A100
14
15  A100 20  04       .      LA100   BRA     LA106
16  A102 01           .              NOP
17  A103 00           .      LA103   FCB     $00
18  A104 00           .      LA104   FCB     $00
19  A105 00           .              FCB     $00
20  A106 B7  A103     ...    LA106   STAA    LA103
21  A109 FF  A104     ...            STX     LA104
22  A10C CE  A115     ...            LDX     #LA115
23  A10F BD  AD1E     ...            JSR     PSTRNG
24  A112 7E  AD03     G..            JMP     WARMS
25  A115 54           T      LA115   LSRB
26  A116 45           E              FCB     $45
27  A117 53           S              COMB
28  A118 54           T              LSRB
29  A119 20  53        S             BRA     LA16E
30  A11B 54           T              LSRB
31  A11C 52           R              FCB     $52
32  A11D 49           I              ROLA
33  A11E 4E           N              FCB     $4E
34  A11F 47           G              ASRA
35  A120 04           .              FCB     $04
36
37                     A100          END     LA100
```

Fig 1. Här är testfilen disassemblerad med options DAN2. Observera hur man med ASCII optionet A lätt ser var textsträngen ligger.

```
                        * DISASSEMBLY BY DYNAMITE OF 0.TESTFILE.BIN
                        * STANDARD PRE-NAMED LABEL EQUATES
                AD03    WARMS   EQU     $AD03
                AD1E    PSTRNG  EQU     $AD1E
                        * ASCII CODE EQUATES
                0004    EOT     EQU     $04

A100                            ORG     $A100

A100 20 04              LA100   BRA     LA106
A102 01                         FCB     $01
A103 00                 LA103   FCB     $00
A104 0000               LA104   FDB     $0000
A106 B7  A103           LA106   STAA    LA103
A109 FF  A104                   STX     LA104
A10C CE  A115                   LDX     #LA115
A10F BD  AD1E                   JSR     PSTRNG
A112 7E  AD03                   JMP     WARMS
A115 54 45 53 54        LA115   FCC     "TEST STRING"
A120 04                         FCB     EOT

                A100            END     LA100
```

Fig 2. Här har vi kört en sång till, men nu med options BNGZ. Därigenom frågade DYNAMITE efter dataområdena. Vi matade in B A102-A103, W A104-A105 samt A A115-A120. Den resulterande listan är så välgjord man kan begära.

```
                        NAM     TESTFILE


                        ORG     $A100
A100 20 04      L00001  BRA     L00002
A102                    FCB     1,0,0,0
A106            L00002  FCB     $B7,$A1,3,$FF,$A1,4,$CE,$A1,$15
A10F                    FCB     $BD,$AD,$1E,$7E,$AD,3
A115                    FCC     /TEST STRING/
A120                    FCB     4
                        END     L00001
```

Fig 3. LABDIS trivdes inte alls med testexemplet. Programmet lyckades aldrig upptäcka var programkoden började, utan genererade FCB satser i stället. Textsträngen har dock upptäckts och taggits om hand på ett snyggt sätt.

```
                        NAM     PRINT
                ACIAC   EQU     $800C
                ACIAD   EQU     $800D


                        ORG     $ACC0
ACC0  86 03             LDA A   #3
ACC2  B7 800C           STA A   ACIAC
ACC5  86 15             LDA A   #$15
ACC7  B7 800C           STA A   ACIAC
ACCA  39                RTS

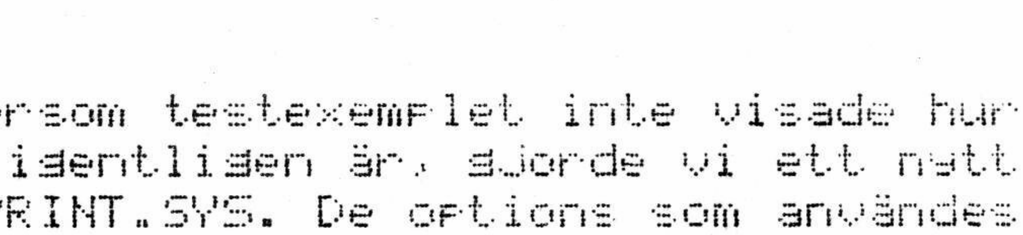

                        ORG     $ACD8
ACD8  37        PREADY  PSH B
ACD9  F6 800C           LDA B   ACIAC
ACDC  54                LSR B
ACDD  54                LSR B
ACDE  56                ROR B
ACDF  33                PUL B
ACE0  39                RTS


                        ORG     $ACE4
ACE4  8D F2     POUTCH  BSR     PREADY
ACE6  2A FC             BPL     POUTCH
ACE8  B7 800D           STA A   ACIAD
ACEB  39                RTS

                        END
```

Fig 4. Eftersom testexemplet inte visade hur bra LABDIS egentligen är, gjorde vi ett nytt försök på PRINT.SYS. De options som användes var EHKN. Tack vare option K kunde vi se namnen ACIAC och ACIAD för $800C resp. $800D. Övriga namn fanns uppenbarligen i EQUATES.DSM.

Svar på frågan om de arabiska siffrorna i nr 4 1980:



Tony Månsson
Tegv. 82
175 45 JÄRFÄLLA.
0758 / 165 44

Bidrag till Mpularen, samt några kommentarer.

Jag tackar och bockar flerfaldigt för Åke Holms CBUG, som jag knappat in och funnit alldeles förskräckligt förträffligt. Bock inte förträffligare än att jag hittat på en förbättring! Inför uppgiften att knappa in C G Nilssons 6800 Disassembler (som jag modifierat för att köras på 6809), greps jag av olust inför utsikten att sitta och slå upp HEX-koden för alla ASCII-bokstäver och därefter trycka in dem hexadecimalt.

Raskt konstruerade jag då denna patch, som möjliggör inmatning av ASCII-tecken medelst CBUG:s M-kommando. Vill man skriva t.ex LDA, så trycker man $L $D $A, varvid 4C 44 41 genast hälkar in i minnet.

Så här gör man:

```
FF9F:   16 05 6E     BRA PATCHAREA  ;(Det var ledigt på FF70)

PATCHAREA:
FF70:   81 24        CMPA  '$'
        26 06        BNE OUT        ; Jaså inte det, återvänd då.
        17 F9 04     LBSR INCHE     ; Hämta Ascii-tecknet.
        16 FA 8F     LBRA MEM       ; Återvänd in på FA09.
OUT:    16 FA 85     LBRA BACK      ; Återvänd in på FA02.
```

Ni som har DOS och Basic och Assembler och sådant, är naturligtvis ointresserade av det här, men för en fattig Hex-knappare kan det vara guld värt.

Jag använder MicroDaSy's S-100 kort med 6809, som ni säkert sett annonser om i BYTE. Om det föreligger intresse för en testrapport, så slå bara en signal.

Hälsningar,

Tony Månsson

INTERELKO VISAR EXORMACS OCH 68000

På klubbmötet 26 sept var Lars Jonasson från INTERELKO med och visade MOTOROLAS nya EXORMACS-system.

MOTOROLA började 1976 att undersöka 16-bitsmaskiner, och i december-77 hade man en spec framme.Tidigare har man tagit fram nya kretsar som kunnat användas ihop med de gamla kretsarna och utan att man behövt skrota den gamla programvaran.Man har varit bakåtkompatibel.

I och med 68000 har man infört ett antal nya filosofier:

1. I och med att CPU-n betjänar flera användare (users) skall den klara av att skifta mellan dessa på ett säkert sätt,så att inte en användare kan störa en annan. Detta gör att man har infört en överliggande övervakningsnivå (Supervisor). Vid felupptäckt (Intern i CPU eller externt via BUSERROR) skall processorn styras över till övervakningsnivån(via vektorer),som återstartar verksamheten.
2. I mer komplicerade system skall man kunna få tilldening av minne och kontroll av att varje användare bara använder sitt minne.
3. Man vill ha en processor kombinerad med programutvecklingshjälpmedel som är inriktade mot högnivåspråk.
4. Processorn skall vara avpassad mot operativsystem av minidatortyp (Timesharing) med speciell inriktning mot säkerhet.

Pinning och inkoppling av 68000

Vad skiljer nu 68000 från 6800?Vi börjar med de pinnar som vi känner igen:

Matning +5V på pin 14 och 49 (Max 300 mA)
jord på pin 16 och 53
Utgångarna A1-A23 på pin 29 till 52 (A0 saknas!)
Utgång R/W på pin 9
Ut/ingångarna D0-D15 på pin 5 till 1 och pin 64 till 54
Ingang CLOCK huvudklocka på pin 15 (Normalt 8 MHz)
Ut/ingång RESET på pin 18 kan numera återställa periferikretsar.

Nu har vi klarat av 46 pinnar,och 18 nya återstår:
68000:s bussystem är en asynkron buss,där alla enheter skall lämna en klarsignal (DTACK pin 10). CPUn väntar obegränsat på denna signal varför det är lämpligt att ha en yttre timer som övervakar bussen och ger BUSERROR (pin 22) om en enhet somnat. Alla andra bussfel,liksom om en användare använder någon annans minne eller liknande felfall signaleras via BUSERROR (pin 22). Beroende på om HALT (pin 17) är 0V eller 5V får man ett nytt försök att klara busscykeln (Rerun) eller att processorn styrs over till övervakningsmod via vektorerna i minnets lägsta del.Dessutom fungerar HALT som på 6800 (dvs vid icke buserr.)

Det asynkrona bussystemet arbetar med ADRESSSTROBE (pin 6) och eftersom A0 saknas har man dessutom en UPPER och en LOWER DATA STROBE (pin 7 resp 8). Upper styr D8-D15 och lower D0-D7. Varje ord (16 bit) ligger sedan bara på de jämna adresserna.

Om man skulle försöka ett hopp till en udda adress uptäcker CPUn detta, och man hamnar i övervakningsmod.På samma sätt upptäcker CPUn om man försöker skriva relativt programräknaren, om en användare försöker använda någon instruktion, som är avsedd enbart för övervakningsmod.

CPUn talar kontinuerligt om vad den sysslar med via en tre bitars funktionskod (FC0 - FC2,pin 26,27 ocn 28) från 0-7.

0	-
1	Dataacess från användare
2	Program (OPkod)acess från användare
3	-
4	-
5	Dataacess från övervakaren
6	Programacess från övervakaren
7	Interrupt acknowledge

Interrupt

Det finns tre interrupingångar (IPL1-IPL2,pin 23,24 och 25) som medger sju interruptnivåer,varav den högsta (7) inte är maskbar. Inkommande interrupt jämförs med en interruptmask i CPUn. Lågt prioriterade interrupt avbryter ej en högre (eller samma) nivå.

Interrupt vektoreras lisom felfallen via en area på 1kbyte i botten (00 00 00 - 00 03 FE) av minnet:

Adress	Vektor
0	Stackpekare som nämtas vid reset
4	Startadress vid reset
8	Vektor för Buserror
C	Vektor vid felaktig adress
10	Vektor vid felaktig instruktion
14	Vektor om Du har försökt dividera med noll
18	Vektor vid fel vid CHK-instruktionen
1C	Vektor för TRAPV-instruktion (motsv SWI) vid overflow.
20	Vektor om Du har försökt använda instruktion eller adressering som är avsedd bara för övervakningsmod.
24	Vektor för TRACE
28	Vektor för instruktioner som börjar på HEX $A ($AXXX).
2C	Vektor för instruktioner som börjar på HEX $F ($FXXX).
30- --	
-5C	Reserverade för framtiden
60	Buserror vid INT ACK.
64- --	
-7C	7 vektorer för ickeintelligenta kringkretsar (6800 fam.)
80- --	
-BC	Vektorer vid TRAP-instruktion (motsv SWI)
C0- --	
-FC	Reserverade
100- --	
-3FC	Användarinterrupt

Ickeintelligenta kringkretsar

De flesta nya kringkretsarna runt 68000 kommer att vara intelligenta, dvs de kommer att kunna svara på interrupt med ett vektornummer, de är anpassade mot den asynkrona bussen och överför data blockvis med DMA.

För att möjliggöra inkoppling av gamla synkrona PIA,ACIA osv har man försett 68000 med VMA och E (enable,dvs gamla Fitvå). När en sådan relik adresseras får 68000 en signal VPA(pin 21),som gör att denlägger ut VMA och E (pin 19 ocn 20).

DMA
==
Dessutom innehåller 68000 tre pinnar för DMA,BUS REQUEST (pin 13), BUS GRANT (pin 11) och BUS GRANT ACKNOWLEDGE (pin 12). Den sistnämnda talar om att någon annan krets är Bus Master. (Den som bestämmer över bussen).

Den som vill läsa mer om 68000-chipet ,se ref 1 och 2.

EXORMACS
=======

Programutveckling för 68000 sker på EXORMACS,som är ett timesharingsystem för ca 8 (Max 32) samtidiga användare. Varje användare har en egen EXORTERM 155 och en fjärdedels seriekommunikationskort (9,6 kbaud). Som sekundärminne kan man välja mellan dubbel dubbelsidig floppy (= 1Mbyte) eller 16M + 16M hårddisk.
Grundkorten är en MPU-modul och en Debug-modul,som är placerade i en låda med MOTOROLAS nya VERSABUS (120 + 140 stift) i botten.Numera kopplar man in I/O i botten på lådan, och 50 busskontakter är reserverade för I/O.

På MPU-modulen sitter ytterligare en länk i säkerhetskedjan, MMU (Minneskontrollenheten) som ger CPU-n felindikering om en användare försöker acessa minne som han ej först fått sig tilldelat.Dessutom kopplar MMU-n om bland adressledningarna,så att CPU-n acessar olika minne beroende på vilken användare som exekverar.På detta sätt har alla användare adress 00 00 00 i "sitt" minne. MPU-modulen innehåller också en Watch-Dog timer för bussöverföringarna.

När man startar EXORMACS så körs först en självtest och resultatet av de denna visas på två hexdisplayer på framsidan. EXORMACS-korten (Ej MPU-modul och Debug-modul) har egna inbyggda testprogram med LED-indikering på korten och möjlighet till RS232-anslutning av terminal.

Diskarna är inkopplade via en diskcontroller,som har en egen MPU (6801). Alla disköverföringar sker med DMA mellan controllerns minne och EXORMACS minne,sedan sköter MPU-n på controllern resten.

Programvaran i EXORMACS består dels av MACSBUG,som monitor och dels av VERSADOS som diskoperativsystem. I VERSADOS har varje användare ungefär samma kommandon som i MDOS. Man arbetar dock överallt med symboliska adresser (även i monitorn). Dessutom har man infört Printersoooling.

Till VERSADOS finns CRT-editor,Macroassembler,PASCAL,FORTRAN och fler lär komma.

Till EXORMACS kan man koppla en USE.SU-modul för hårdvaruutveckling. Denna metod är bäst i enanvändarsystem (Systemet måste ju stängas av vid omkopplingar i systemet) men i fleranvändarsystem kan särskilda hårdvarutvecklingssystem med egen 68000 MPU kopplas in.De kommunicerar med EXORMACS över ett RS232 ((V24)interface (serieinterface).

Den som vill läsa mer om EXORMACS hänvisas till ref 3.

Lars Jonasson gick igenom hela EXORMACS och visade hur den är

uppbyggd. Han visade också hur den kan plockas isär (jodå , han fick ihop den igen).Därefter hade han en genomgång av MEX 68000 KDM,som är ett kort med EXORCISER-busskontakt (för kraft) och en serieutgång fö en terminal.Den är försedd med en egen 68000 MPU, MACSBUG,32 kbyte RAM,2st PIA,1st 6840 och en extra serieport.

MEX 68000 KDM är beskriven i ref 4.
Till sist avslutade Lars med att tala om att flera nya kringkretsar var att vänta till 68000-serien (MMU,Intelligenta Perifericontrollers o.s.v.).

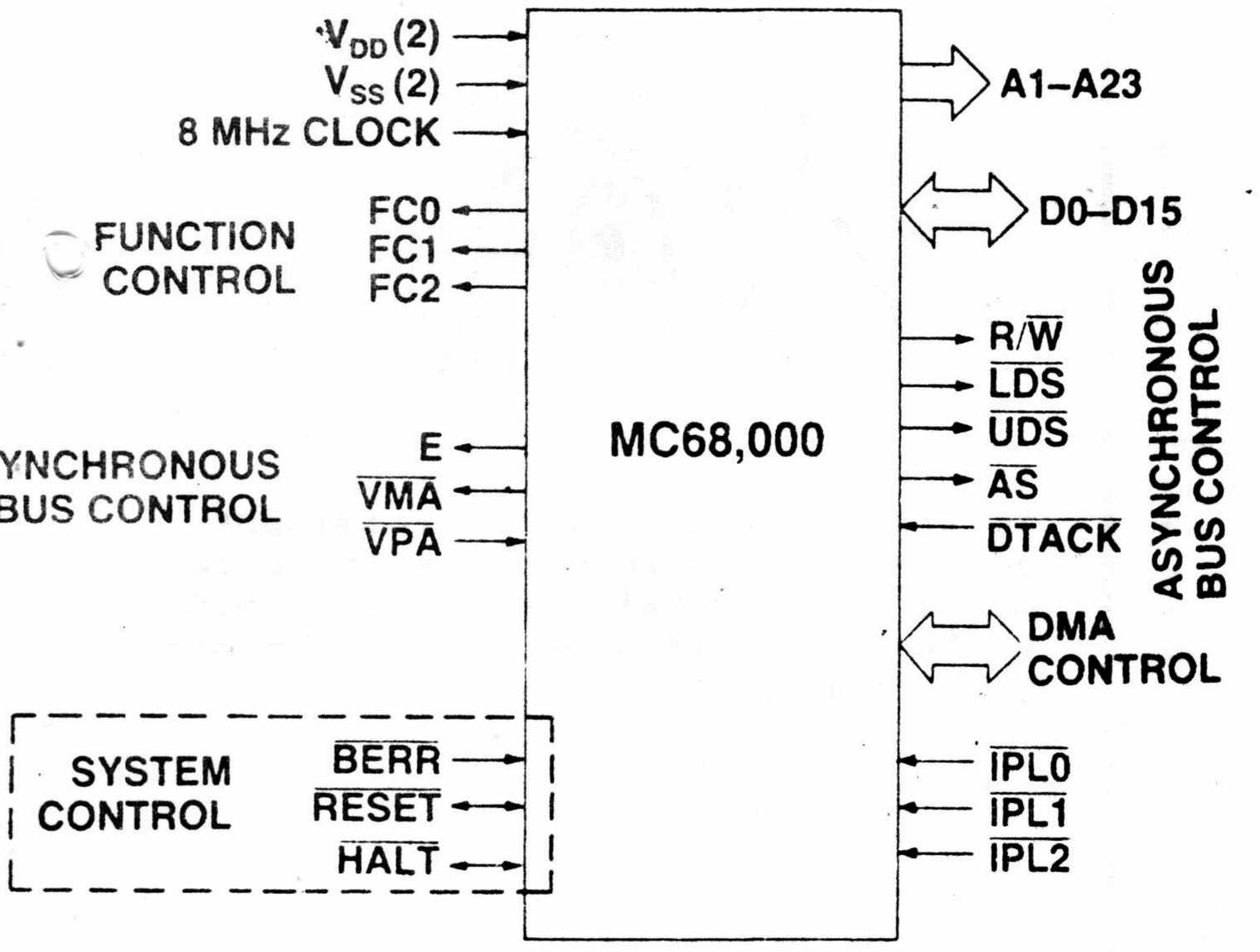
En genomgång av dessa finns i ref 5.

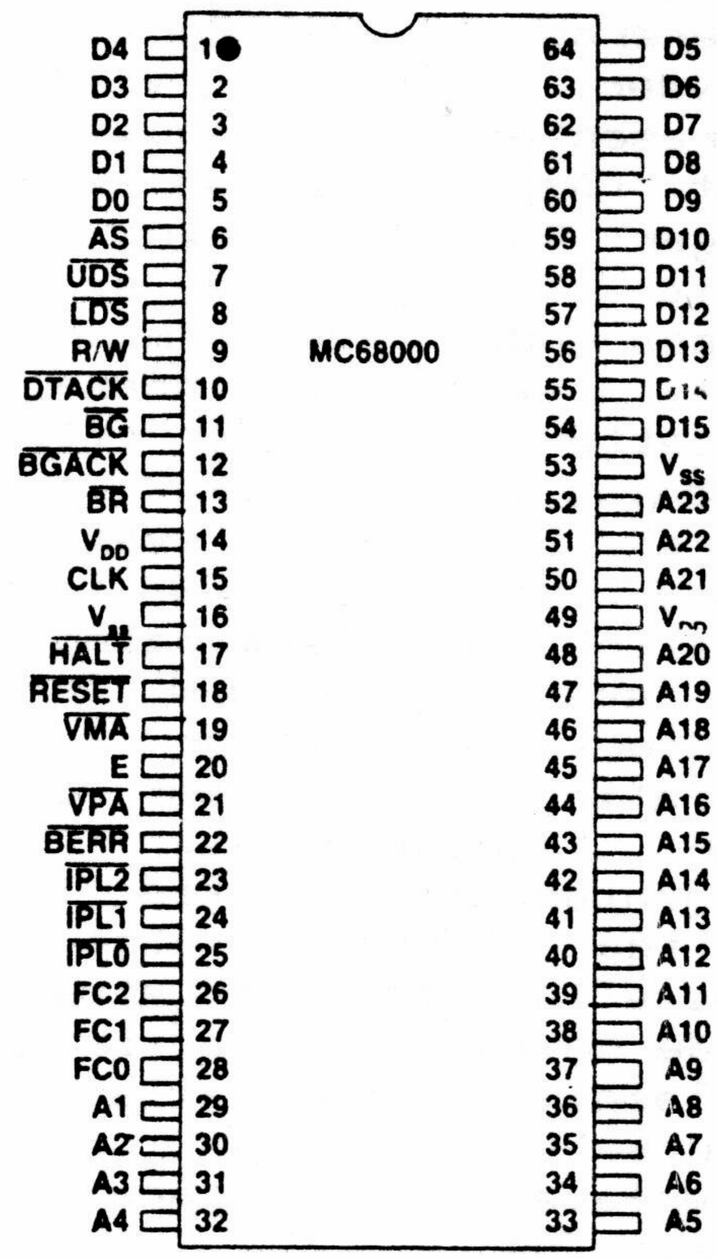
68000 används alltså både i mycket omfattande och i mindre omfattande system. Vem i klubben blir först med att skriva om MIKBUG för 68000? Vem bygger första "minisystemet" med 68000?

CAB

REFERENSER:
==========

1. Advance Information MC68000 (ADI-814)
2. MC68000 16-bit microprocessor,users manual.
3. Broshyr EXORMACS System,The advanced microcomputer development system that takes you into the future.
4. Advance information,MEX68KDM, MC68000 design module.
5. M68KFM, M68000 Family.





SPELHÅLAN

Här kommer ånyo ett litet bidrag till spelbiblioteket. Denna gång är det en variant av det gamla hederliga luffarschacket. Man är två som spelar, och det gäller att försöka placera fem av ens symboler i rad. Vågrätt, lodrätt eller diagonalt. Genom att placera sitt tecken i vägen för motspelaren kan man blockera varandra.

Spelet är ganska vanligt på mikrodatorer. Kallas ofta "TIC-TAC-TOE". Våran variant består av 19 ggr 19 rutor. Och det bör ju räcka en stund.

Den som vill ha ett exemplar av programmet skickar en 51/4" diskett eller audio kassett. Vi kopierar utan kostnd.

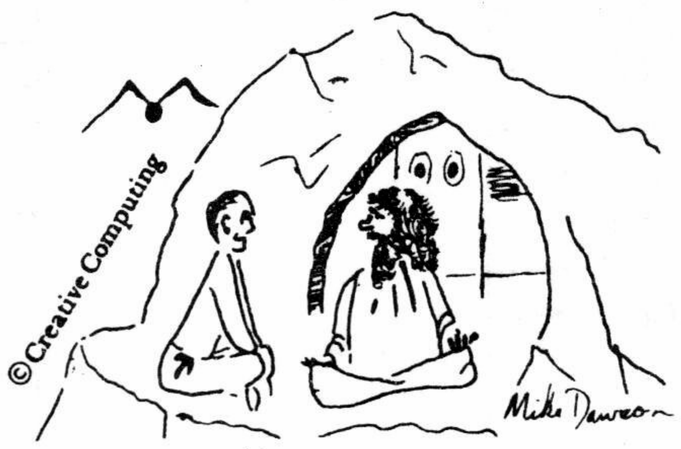
En sak bara som rör lagringsmetoden för KCS kassetter. Programmet kan endast levereras i källformat. Det betyder att varje rad representeras av radnummer, sats samt CR+LF. Mellan varje rad ligger en fördröjning för att Basic intepretatorn skall hinna stoppa undan raden i minnet. På detta sätt är innehållet på kassetten inte bundet till något laddningsformat. För att kunna "lura" Basic att svälja detta måste man ha KCS interfacet parallellt med terminalen. Genom att sedan göra NEW och slå på bandspelaren, kommer Basic att uppfatta raderna som om de kom från tangentbordet. Eftersom tangentbordet alltid levererar vanliga tecken utan några speciella start och stopp koder, borde detta sätt att ladda vara tillämpligt på alla kärror. Förutsatt alltså att KCS interfacet sitter parallellt med terminalen.

Efter programlistan av GOBANG finner vi ett exempel på hur körningen tar sig ut. Utskriften av brädet är anpassad till en 40 tecken/rad terminal. Den som har bredare skärm kan modifiera brädesutskriften.

Allra sist har 6809 programmet XREF.CMD körts på GOBANG. XREF producerar en lista med korsreferenser på ett Basic program. Man kan se var variabler har refererats, varifrån man hoppat till vissa lägen etc. Ett riktigt trevligt och i vissa sammanhang mycket användbart program.

Förresten, medan vi ändå är inne på spelprogrammen; Alla ni som med stor möda knappat in ett program, hur vore det låta andra få del av mödan. Skicka en rad till PD68 om du har något bra spelprogram. Eller varför inte en diskett eler kassett. Vi returnerar materialet om du talat om vad du heter.

TB1



"Livets Mening?" Vi har frisläppt den som ett Basic program för alla mikro system. Det kostar 60.95 och finns i alla affärer. Hav en angenäm tillbakaresa.

GOBANG.BAS

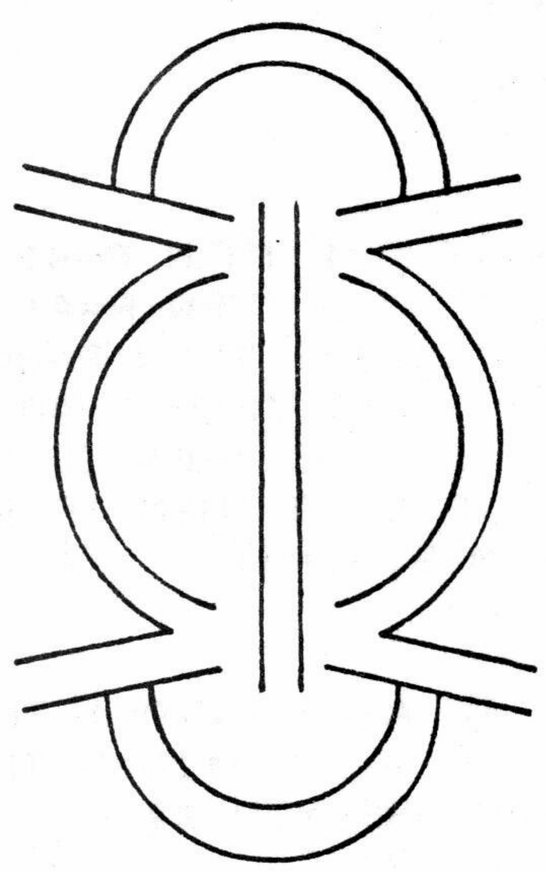
```
DAT: 81-02-07  15:54:26
10 REM ** GOBANG (BYTE NOV 1979)
20 REM ** INKNAPPAD AV TBL 79-12-09 **
30 REM ** RÄTTAD 1981-02-07 TBl
40 GOSUB 1750
50 DIM M(19,19),T(27,27),S(81)
60 FOR I=1 TO 81
70  S(I)=0
80 NEXT I
90 S(20)=1
100 S(10)=40
110 S(12)=30
120 S(13)=47
130 S(27)=15
140 S(28)=20
150 S(29)=10
160 S(30)=40
170 S(31)=50
180 S(32)=30
190 S(24)=1
200 S(36)=39
210 S(37)=65
220 S(38)=40
230 S(39)=70
240 S(40)=100
250 S(41)=60
260 S(42)=30
270 S(43)=30
280 S(44)=30
290 S(62)=41
300 S(72)=31
310 S(73)=11
320 S(74)=41
330 S(78)=51
340 S(80)=90
350 S(26)=21
360 S(79)=40
370 S(60)=21
380 S(61)=11
390 REM ** NOLLA BRÄDET OCH BÄSTA DRAG ARRAYEN **
400 FOR I=1 TO 27
410  FOR J=1 TO 27
420   IF I<19 THEN IF J<19 THEN M(I,J)=0
430 IF I<19 THEN IF J<19 THEN T(I,J)=0
440  NEXT J
450 NEXT I
460 C=-1
470 W=14
480 N=14
490 O=14
500 X=14
510 GOTO 870
520 GOSUB 930
530 REM ** BEGÄR DRAG OCH KOLLA GILTIGHET **
540 PRINT "DITT DRAG (X,Y) ";
541 ON ERROR GOTO 2000
550 INPUT Z,Y
560 Y=Y+4
570 Z=Z+4
```

```
580 IF Y>23 THEN 540
590 IF Z>23 THEN 540
600 IF Y<5 THEN 540
610 IF Z<5 THEN 540
620 IF T(Y,Z)>0 THEN 540
630 T(Y,Z)=2
640 I=Y
650 J=Z
660 REM ** STUDERA SISTA TVÅ DRAGEN **
670 GOSUB 1100
680 IF C<>-1 THEN 900
690 REM ** OM C=0 HAR DATORN FÖRLORAT **
700 I=W
710 J=X
720 GOSUB 1100
730 REM ** SCANNA BRÄDET FÖR BÄSTA DRAG **
740 REM ** NOTERA BEGRÄNSNINGARNA FÖR UPPSNABBNING **
750 Q=-1
760 FOR I=N-1 TO O+1
770  FOR J=5 TO 23
780   IF T(I,J)>0 THEN 840
790   A=M(I-4,J-4)
800   IF A<Q THEN 840
810   W=I
820   X=J
830   Q=A
840  NEXT J
850 NEXT I
860 PRINT "MITT DRAG ";X-4,",";W-4
870 T(W,X)=1
880 IF M(W-4,X-4)<100 THEN 520
890 PRINT CHR$(7);"JAG **VINNER**";CHR$(7)
900 IF C=0 THEN PRINT "DU VINNER"
910 GOTO 390
920 REM ** SUB FÖR ATT VISA BRÄDET **
930 PRINT "    12345678901234567890"
940 FOR I=5 TO 23
950  IF I-4<10 THEN PRINT I-4;" ";
960  IF I-4>9 THEN PRINTI-4;
970  FOR J=5 TO 23
980   IF T(I,J)=0 THEN PRINT ":";
990   IF T(I,J)=1 THEN PRINT "X";
1000   IF T(I,J)=2 THEN PRINT "O";
1010  NEXT J
1020 PRINT
1030 NEXT I
1040 RETURN
1050 REM ** SUB BERÄKNA BÄSTA DRAG **
1060 REM ** SCANNA GENOM DRAG VID I,J **
1070 REM ** FÖR F KAVDRATER VID VARJE SIDA AV DRAGET **
1080 REM ** I ÅTTA RIKTNINGAR OCH **
1090 REM ** UPPDATERA BÄSTA DRAG ARRAYEN **
1100 K=1
1110 L=-1
1120 IF I<N THEN IF I>5 THEN N=I
1130 IF I>O THEN IF I<23 THEN O=I
1140 REM ** UPPDATERA SCAN GRÄNSERNA **
1150 U=I
```

```
1160 V=J
1170 REM ** I,J ÄR DRAG ATT KOLLA, D ÄR LOOP RÄKNARE **
1180 REM ** K,L ÄR X,Y-RIKTNINGAR GENOM DRAGET **
1190 D=0
1200 D=D+1
1210 P=81
1220 REM ** KOLLA OM FORTFARANDE PÅ BRÄDET **
1230 IF U>23 THEN 1660
1240 IF V>23 THEN 1660
1250 IF U<5 THEN 1660
1260 IF V<5 THEN 1660
1270 E=U-4
1280 G=V-4
1290 A=M(E,G)
1300 Q=T(U+K,V+L)
1310 REM ** BERÄKNA POSITIONENS PRIORITET **
1320 R=T(U-K,V-L)*27+T(U-2*K,V-2*L)*9
1330 R=R+T(U-3*K,V-3*L)*3+T(U-4*K,V-4*L)
1340 B=Q*27+T(U+2*K,V+2*L)*9+T(U+3*K,V+3*L)*3
1350 IF R=80 THEN IF T(U,V)=2 THEN C=0
1360 IF T(U,V)<>0 THEN 1610
1370 REM ** S(R) ÄR PRIORITET. FöLJANDE ÄR UNDANTAG **
1380 REM ** SE TABELL 2 **
1390 IF R<14 THEN IF R>11 THEN IF Q=1 THEN P=37
1400 IF R>71 THEN IF B>53 THEN IF B<63 THEN P=80
1410 IF R>71 THEN IF B>71 THEN P=80
1420 IF R>53 THEN IF R<63 THEN IF Q=2 THEN P=72
1430 IF P=72 THEN IF R=60 THEN P=31
1440 IF Q<>2 THEN 1480
1450 IF R=78 THEN P=80
1460 IF R=79 THEN P=80
1470 IF R=41 THEN R=81
1480 IF R<42 THEN IF R>35 THEN IF Q=1 THEN P=41
1490 IF R<33 THEN IF R>29 THEN IF Q=1 THEN P=41
1500 IF R>53 THEN IF R<63 THEN IF B>71 THEN P=80
1510 IF R>38 THEN IF R<42 THEN IF Q=1 THEN R=40
1520 IF R>35 THEN IF R<45 THEN IF B>35 THEN IF B<45 THEN R=40
1530 IF R>27 THEN IF R<54 THEN IF B>38 THEN IF B<42 THEN R=40
1540 IF R=79 THEN IF A=51 THEN M(E,G)=41
1550 IF R=0 THEN R=81
1560 IF S(P)>S(R) THEN R=P
1570 IF S(R)-S(R)/10*10=1 THEN IF A-A/10*10=1 THEN IF S(R)<41 THEN R=74
1580 IF S(R)-S(R)/10*10=9 THEN IF A-A/10*10=9 THEN IF S(R)<65 THEN R=37
1590 REM ** UPPDATERA BÄSTA DRAG ARRAYEN **
1600 IF S(R)>M(E,G) THEN M(E,G)=S(R)
1610 IF D>4 THEN 1660
1620 U=U+K
1630 V=V+L
1640 GOTO 1200
1650 REM ** ÄNDRA RIKTNING **
1660 IF K=0 THEN IF L=-1 THEN RETURN
1670 IF K=-1 THEN IF L=-1 THEN K=0
1680 IF K=-1 THEN IF L=0 THEN L=-1
1690 IF K=-1 THEN IF L=1 THEN L=0
1700 IF K=0 THEN IF L=1 THEN K=-1
1710 IF K=1 THEN IF L=1 THEN K=0
1720 IF K=1 THEN IF L=0 THEN L=1
1730 IF K=1 THEN IF L=-1 THEN L=0
```

```
1740 GOTO 1150
1750 PRINT
1760 PRINT "* * *   G O B A N G   * * *"
1770 PRINT
1780 PRINT "GOBANG ÄR ETT LUFFARSCHACK MED"
1790 PRINT "19*19 RUTOR. DET GÄLLER SOM"
1800 PRINT "VANLIGT ATT FÅ 5 I FÖLJD. HORI-"
1810 PRINT "SONTELLT,VERTIKALT ELLER DIAGO-"
1820 PRINT "NALT. DATORN HAR 'X' DU 'O'"
1830 PRINT
1840 PRINT "DU GER ETT DRAG MED X,Y KOORDINATER."
1850 PRINT "EXEMPELVIS: 9,9"
1860 PRINT
1870 PRINT "SKRIV SIFFRAN 1 NÄR DU VILL SPELA "
1880 INPUT Z
1890 RETURN
2000 PRINT CHR$(7);"SKRIV SIFFROR"
2010 RESUME 540
```

EFTERLYSES: SYMBOL FÖR PD68



Någonting som vi saknat i klubben är en symbol för PD68. Det jag syftar på är en figur som man kan ha på brev och andra officiella papper. Som man till och med skulle kunna göra en nål med. Att ha på tröjan e.dyl. Kanske även göra en självhäftande dekal att fästa på bilen. Då skulle man kunna känna igen likasinnade och utbyta en hälsning.

Geniala förslag mottages tacksamt. Här ovan ser ni ett sådant. Det är en symetrisk, stiliserad symbol som innehåller alla tecknen i ordet PD68. Vad tycks? Man måste kanske ha lite fantasi för att se det direkt. Vi låter förslagställaren tills vidare vara anonym.

TBl

```
RUN


* * *   G O B A N G   * * *

GOBANG ÄR ETT LUFFARSCHACK MED
19*19 RUTOR. DET GÄLLER SOM
VANLIGT ATT FÅ 5 I FöLJD. HORI-
SONTELLT,VERTIKALT ELLER DIAGO-
NALT. DATORN HAR 'X' DU 'O'

DU GER ETT DRAG MED X,Y KOORDINATER.
EXEMPELVIS: 9,9

SKRIV SIFFRAN 1 NÄR DU VILL SPELA
? 1
    12345678901234567890
 1  ....................
 2  ....................
 3  ....................
 4  ....................
 5  ....................
 6  ....................
 7  ....................
 8  ....................
 9  ....................
 10 ..........X.........
 11 ....................
 12 ....................
 13 ....................
 14 ....................
 15 ....................
 16 ....................
 17 ....................
 18 ....................
 19 ....................
DITT DRAG (X,Y) ? 10,9
MITT DRAG  11   , 11
    12345678901234567890
 1  ....................
 2  ....................
 3  ....................
 4  ....................
 5  ....................
 6  ....................
 7  ....................
 8  ....................
 9  ..........O.........
 10 ..........X.........
 11 ..........X.........
 12 ....................
 13 ....................
 14 ....................
 15 ....................
 16 ....................
 17 ....................
 18 ....................
 19 ....................
DITT DRAG (X,Y) ? 10,8
```

```
MITT DRAG  12   , 12
    12345678901234567890
 1  ....................
 2  ....................
 3  ....................
 4  ....................
 5  ....................
 6  ....................
 7  ....................
 8  ..........O.........
 9  ..........O.........
 10 ..........X.........
 11 ..........X.........
 12 ..........X.........
 13 ....................
 14 ....................
 15 ....................
 16 ....................
 17 ....................
 18 ....................
 19 ....................
DITT DRAG (X,Y) ? 9,9
MITT DRAG  13   , 13
    12345678901234567890
 1  ....................
 2  ....................
 3  ....................
 4  ....................
 5  ....................
 6  ....................
 7  ....................
 8  .........O..........
 9  .........OO.........
 10 ..........X.........
 11 ..........X.........
 12 ..........X.........
 13 ..........X.........
 14 ....................
 15 ....................
 16 ....................
 17 ....................
 18 ....................
 19 ....................
DITT DRAG (X,Y) ? 14,14
```

XREF.CMD

```
DAT: 81-02-07  16:58:05
XREF OF PROGRAM GOBANG   CREATED 81-02-07

"     12345                    930
" "                            950
"* * *   G                     1760
","                            860
"19*19 RUT                     1790
":"                            980
"DITT DRAG                     540
"DU GER ET                     1840
"DU VINNER                     900
"EXEMPELVI                     1850
"GOBANG ÄR                     1780
"JAG **VIN                     890
"MITT DRAG                     860
"NALT. DAT                     1820
"O"                            1000
"SKRIV SIF                     1870 2000
"SONTELLT,                     1810
"VANLIGT A                     1800
"X"                            990
A                              790 800 830 1290 1540 1570 1580
B                              1340 1400 1410 1500 1520 1530
C                              460 680 900 1350
D                              1190 1200 1610
E                              1270 1290 1540 1600
G                              1280 1290 1540 1600
I                              60 70 80 400 420 430 450 640 700 760 780 790
                               810 850 940 950 960 980 990 1000 1030 1120
                               1130 1150
J                              410 420 430 440 650 710 770 780 790 820 840
                               970 980 990 1000 1010 1160
K                              1100 1300 1320 1330 1340 1620 1660 1670 1680
                               1690 1700 1710 1720 1730
L                              1110 1300 1320 1330 1340 1630 1660 1670 1680
                               1690 1700 1710 1720 1730
M                              50 420 790 880 1290 1540 1600
N                              480 760 1120
O                              490 760 1130
P                              1210 1390 1400 1410 1420 1430 1450 1460 1480
                               1490 1500 1560
Q                              750 800 830 1300 1340 1390 1420 1440 1480
                               1490 1510
R                              1320 1330 1350 1390 1400 1410 1420 1430 1450
                               1460 1470 1480 1490 1500 1510 1520 1530 1540
                               1550 1560 1570 1580 1600
S                              50 70 90 100 110 120 130 140 150 160 170 180
                               190 200 210 220 230 240 250 260 270 280 290
                               300 310 320 330 340 350 360 370 380 1560
                               1570 1580 1600
T                              50 430 620 630 780 870 980 990 1000 1300
                               1320 1330 1340 1350 1360
U                              1150 1230 1250 1270 1300 1320 1330 1340 1350
                               1360 1620
V                              1160 1240 1260 1280 1300 1320 1330 1340 1350
                               1360 1630
W                              470 700 810 860 870 880
X                              500 710 820 860 870 880
```

```
Y                                   550 560 580 600 620 630 640
Z                                   550 570 590 610 620 630 650 1880
390 CALLED BY                       910
520 CALLED BY                       880
540 CALLED BY                       580 590 600 610 620 2010
840 CALLED BY                       780 800
870 CALLED BY                       510
900 CALLED BY                       680
930 CALLED BY                       520
1100 CALLED BY                      670 720
1150 CALLED BY                      1740
1200 CALLED BY                      1640
1480 CALLED BY                      1440
1610 CALLED BY                      1360
1660 CALLED BY                      1230 1240 1250 1260 1610
1750 CALLED BY                      40
2000 CALLED BY                      541
```



"Jag skulle vilja återanställa er Figsby...maskinen som vi köpte för att ersätta er arbetar inget vidare den heller."

PD68:S MEDLEMSREGISTER DEN 9 FEBRUARI 1981

Här har vi nu registret igen. Åtskilliga har tillkommit sedan vi visade det förra gången. En nyhet i det alfabetiska registret är att också telefonnumren är upptagna. De som via enkäten eller på annat sätt önskat ha viss sekretess på telefonnumret har bönhörts.

Vidare har registret kompletterats med uppgifter om datautrustning. Det är tack vare enkäten som dessa kunnat insamlats. Av utrymmesskäl är registrets utskrifter uppdelade i tre avdelningar. Först samtliga medlemmar i bokstavsordning. Med adress och telefonnummer. Därefter i alfabetisk ordning de som har lämnat uppgift om utrustning. I den tredje listan har adressregistret sorterats på postnummret. Härigenom kan kontaktskapandet underlättas. Nu är det ingen match att hitta närbelägna medlemmar. Eftersom den fullständiga adresslistan redan finns, har bara namn och postadress medtagits. Sorteringen och den editerade utskriften är gjord med TSC:s SORT/MERGE paket. Ett fantastiskt flexibelt sorteringspaket. Otroligt snabbt dessutom. En eloge åt TSC! Verkligen!..

I utrustningslistan har en del nödvändiga förkortningar använts. SWTPc 0 betyder exempelvis att det rör sig om en SWTPc 6800 maskin. En CA 0/9 anger den dubbelt CPU bestyckade 6800 (08)/6809 datorn.

Frågorna om terminal var inte riktigt tydliga. Därför har svaren ibland blivit onödigt luddiga. T e x "12 tums TV". Här betyder CA-790 den sk intelligenta terminalen. CT-82 är en annan intelligent terminal. Av fabrikat SWTPc. TTY står för Tele-TYpewriter.

I kolumnen för massminne anges 5" resp 8" för floppy system med de små resp stora skivorna. KCS står för audio kassett interface enligt Kansas City Standard. Remsa ock digital kassett förekommer också. Vissa hårddiskar kan också spåras.

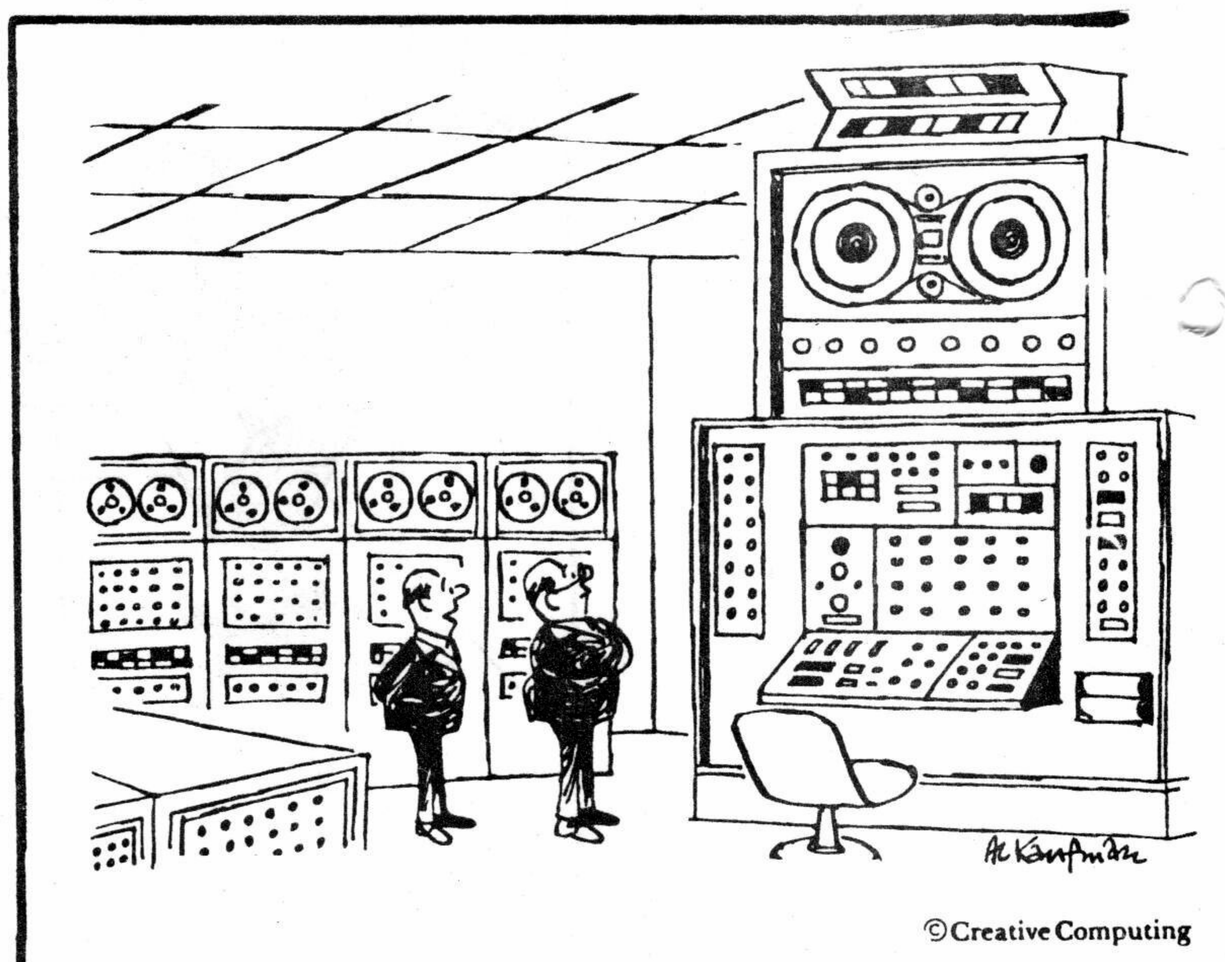
Det allra vanligaste diskoperativsystemet är Flex för 6800 eller 6809. I registret står Flex 2/9. Några andra förekommer också, men TSC verkar ha tagit hem storslam.

Sista kolumnen innehåller printrar. Allt från TTY till den snabbskrivande TI 810 från Texsas Instrument finns med.

Den som funderar över att köpa en viss utrustning kan kanske få tips i den här listan.

Kolla gärna igenom det som rör dig i registret. Felaktig adress anmäls med fördel på ett portofritt litet kort som finns på varje postkontor. Vi föredrar detta kort framför att få en adressändring i slutet på ett brev i ett helt annat ärende. Vill du komplettera med uppgifter, exempelvis telefonnummer eller utrustning, skriv en rad.

TB1



©Creative Computing

"Den kan faktiskt inte tänka, men då den gör ett misstag, kan den skylla på en annan dator."

RADIOAMATÖRER INOM PD68

AX, BÖRJE	SM7FON
FORSBERG, JÖRGEN	
HALL, TORE	SM5BTR
JACOBSSON, CURT	SM5FEX
KARLSSON, THOMMY	SM6HUB
LINGEN, CHRISTIAN	
LUNDBERG, P-A	SMØJCF
NIKKELSEN, JÖRN	
NOORD, ROLF	SMTGED
SÖDERBERG, JAN	SM5ESP
ÅSENTORP, GÖRAN	SM3DSK
ÖYSTILÄ, ÅKE	SM5CM1

Sammanfattning av enkät om klubbdator.

Mindre än hälften av klubbens medlemmar svarade på enkäten. Övervägande delen (ca 60 st.) av svaren var positiva till klubbdator. Dock med en hel del synpunkter, som här redovisas med de flest återkommande åsikterna först.

Enbart programbank.

Möjlighet till utskrift på printer.

Mer än ett modem.
Billigt modemformat.
(Då televerkets normer ej tillåter KCS-modem, kan det inte användas.)

Angående processortyp fördelade sig svaren ungefär lika mellan 6800 och 6809.
Dock med någon övervikt för 6800. Även 68000 föreslogs.

I frågan, om massminne, vill de flesta ha så stort minne som möjligt, med både 8" och 5 1/4".

När det gäller vilken mjukvara som skulle finnas i maskinen var åsikterna många, se nedan.
Uniflex (multi-task operativsystem).
Kompilatorer för BASIC,FORTRAN,PASCAL.
Eget operativsystem för klubben.
(Vem/vilka skriver ett sådant ?)

Allmänt:
Lokal för datorn eller ska den hysas in hos någon ?
Vem blir ansvarig för den ?
Datalagen !
Dyra telefonräkningar för de som bor långt ifrån datorn.
Kostnad för medlemmarna att köra mot datorn ?
Att användas för att skicka meddelande till medlemmarna ?
Passwords ?
Tidsbegränsning ?

Anders