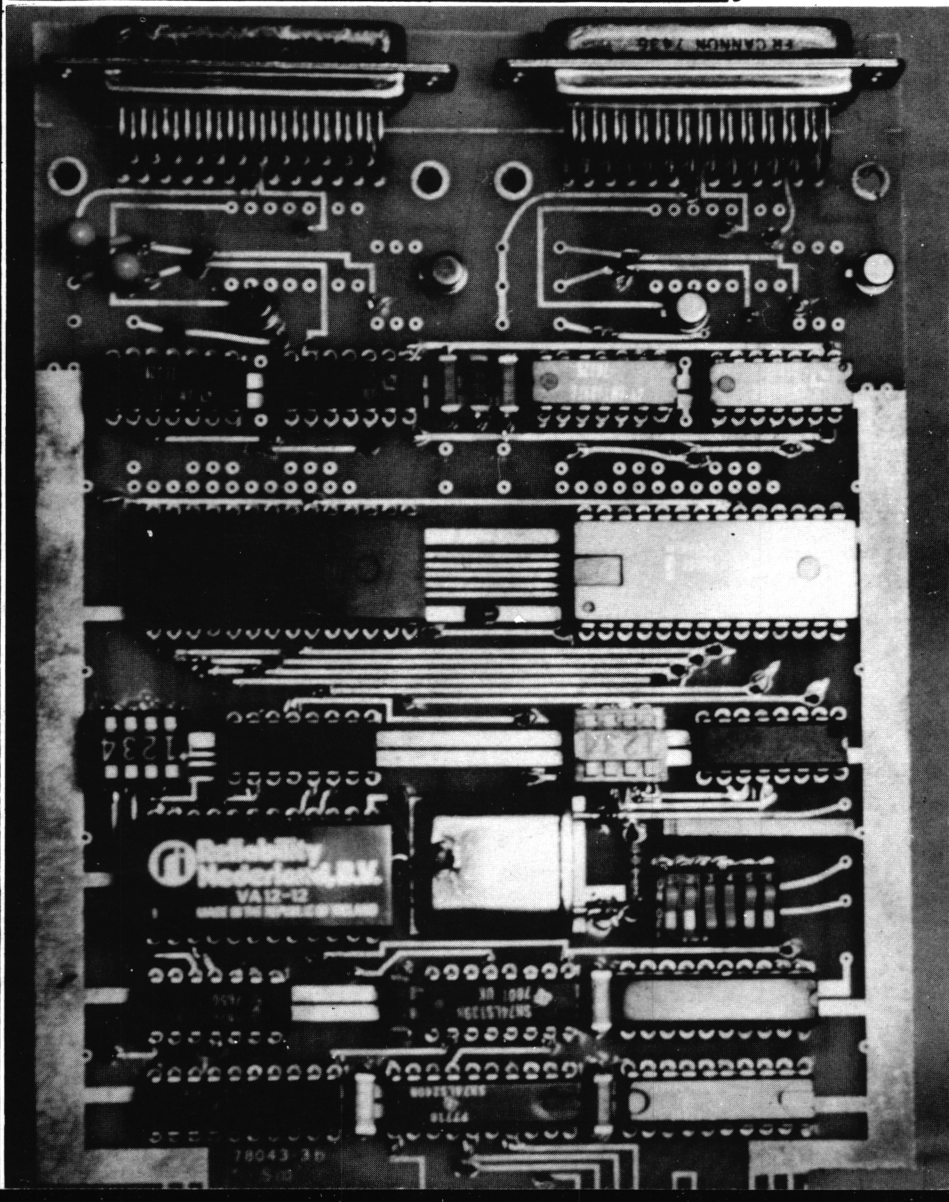
# 9 Håndbog for datamat-amatører

# RC 1978



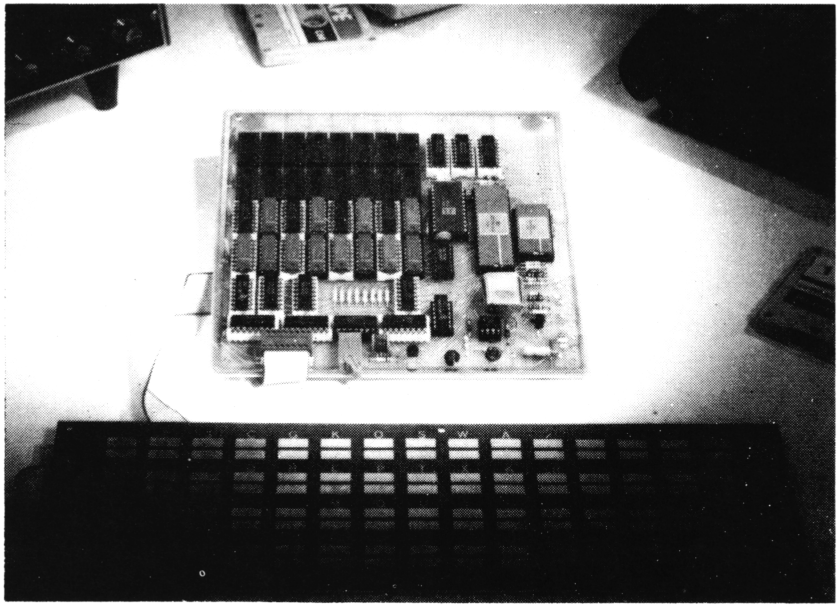
# INDHOLDSFORTEGNELSE




**Håndbog for datamat-amatører** udgives i løsbladsformat af Telepress ApS, Greve Strandvej 42, 2670 Greve Strand. Tlf. (02) 90 86 00. Giro nr. 1 15 53 69. Tryk: Fraling Offset, Viby Sj. HFD udsendes til abonnenter som trykksag d. 1. torsdag hver måned. 1. nummer udgivet er nr. 9/1977. Et årsabonnement koster kr. 100,– incl. ringbind og porto. Ansvarshavende udgiver: H. Lind.

ISSN 0105–581X

# TELMAC 1800



Har du et TV, en kassettebåndoptager og en strømforsyning?
For så klarer TELMAC 1800 resten!

- RCA CDP 1802 med 91 instruktioner.
- 2K RAM på kortet med direkte plads til yderligere 2K. Eksternt til 32K!
- Interface til TV og kassettebåndoptager.
- Tastatur med 64 mulige funktioner inkluderet.

RCA 1802 Håndbog Kr. 80,– incl. moms
TELMAC 1800 samlet + afprøvet 2195,– incl. moms

INTRODUKTIONSPRIS

TELMAC 1800 -KIT kr. 1.770,–
Kan leveres færdigsamlet

Programkassette kr. 175,–
priserne er incl. 18 % moms.

# piezodan aps.

**Bakkedraget 55 - DK 3480 Fredensborg - Tlf. (03) 28 37 44 - Teknisk afd. (01) 86 12 17**

## INDSTILLING

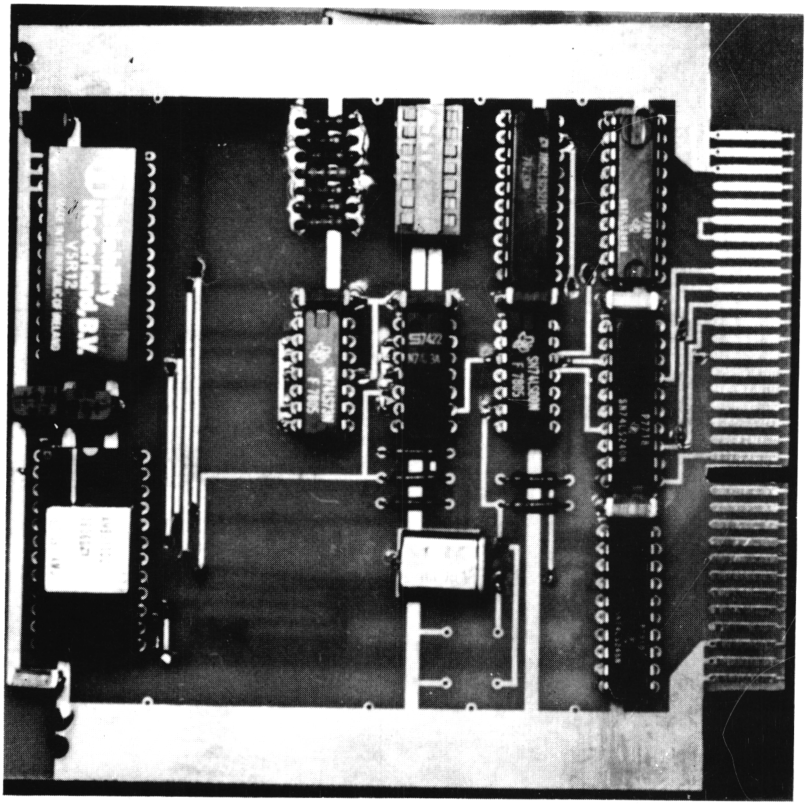
Som tidligere skrevet, kan kortet ved hjælp af loddelus ændres til forskellige serietransmissionsformer, men også adressen og de 2 kanalers Baud-rate kan stilles frit.

Som det fremgår af fotoet, er der regnet med DIL-switch til disse to ting; men for at gøre kortet billigere kan man selvfølgelig også her bruge lus.

Indstilling af adresser er lige ud ad landevejen. Blot skal man være opmærksom på at kortet, hvadenten begge kanaler er monteret eller ej, optager 4 adresser i rækkefølge. Disse er fordelt således, at de 2, hvor A0 er lig med 0 er dataindgange og de 2 andre kontrolindgange. Sagt med andre ord, sætter du kortets adresse til D8 (11011000), har kanal A's dataindgang adressen D8, kanal A's kontrolindgang adressen D9 (11011001),

| $S_3$ | $S_2$ | $S_1$ | $S_0$ | BAUD-RATE |
|---|---|---|---|---|
| L | L | L | L | X |
| L | L | L | H | X |
| L | L | H | L | 50 Baud |
| L | L | H | H | 75 Baud |
| L | H | L | L | 134.5 Baud |
| L | H | L | H | 200 Baud |
| L | H | H | L | 600 Baud |
| L | H | H | H | 2400 Baud |
| H | L | L | L | 9600 Baud |
| H | L | L | H | 4800 Baud |
| H | L | H | L | 1800 Baud |
| H | L | H | H | 1200 Baud |
| H | H | L | L | 2400 Baud |
| H | H | L | H | 300 Baud |
| H | H | H | L | 150 Baud |
| H | H | H | H | 110 Baud |

Tabel 1 - Baud-rate tabel

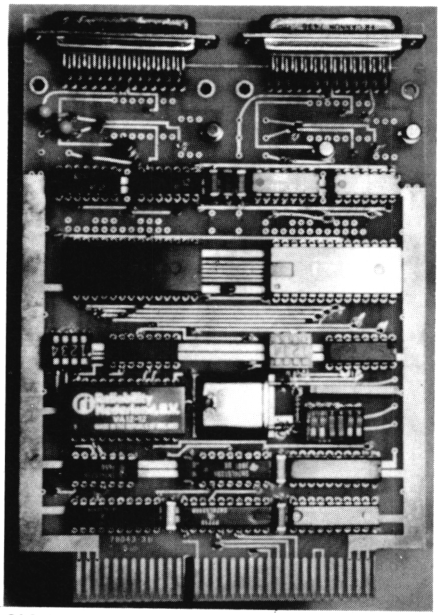


APU-kortet med den aritmetiske kreds.

kanal B dataindgang adressen DA (110-11010) og kanal B's kontrolindgang adressen DB (11011011). De 2 kanalers Baud-rate kan stilles uafhængigt af hinanden til 14 forskellige hastigheder. De sidste 2 muligheder, som findes i de 4-polede DIL-switch, er beregnet til specielle hardware-koblinger, og kan ikke udnyttes på dette kort.
De 14 muligheder fremgår af tabel 1. Det skal her understreges, at denne tabel forudsætter, at man anvender det i styklisten angivne krystal. Hvis man ønsker andre hastigheder, kan krystalværdien ændres, men dette påvirker begge kanaler.

## INITIALISERING

Den anvendte USART er af den programmerbare type. Dette betyder, at hver gang datamaten starter, eller at der gives restart, skal den modtage ordre fra CPU'en, som nærmere beskriver under hvilke betingelser, serietransmissionen skal foregå.
Det drejer sig blandt andet om karakterlængde, antal stopbit, synkron- eller asynkron transmission, lige- eller ulige paritet o.s.v.
Den fulde historie om disse kontrolordrer kan findes i de forskellige fabrikaters datablade, men her vil vi et øjeblik se nærmere på den mest almindelige initialisering, nemlig den, der benyttes i forbindelse med en normal asynkron terminal.
Først skal seriekanalens "mode" indstilles. Her bruges normalt en Baud-rate faktor på 16, 8 bit karakterlængde, ingen paritet og 2 stopbit.
Udfra disse oplysninger fortæller databladene så, at det første kontrolord skal være CE.
8251 skal derefter modtage det såkaldet "kommando-ord". Dette ord starter kredsens sender og modtager samt kontrollerer den tilsluttede terminal via DTR og RTS udgangene.
Her siger databladene, at kontrolordet "27" skal anvendes.
Tænker vi os nu, at initialiseringen er det første, der sker når maskinen startes,



**SIO-kortet med de to USART'er i midten.**

kommer programmet til at se således ud, idet alle tal er opgivet i hexadecimalt:

| Adr. | Code | |
|---|---|---|
| 0000 | 3E CE | LD A, Mode |
| 0002 | D3 C1 | OUT C1 |
| 0004 | 00 | NOP |
| 0005 | 00 | NOP |
| 0006 | 00 | NOP |
| 0007 | 00 | NOP |
| 0008 | 3E 27 | LD A, kontrol |
| 000A | D3 C1 | OUT C1 |

Kanalen, hvis dataadresse er C0, er nu klar til at arbejde; men dette program må gentages, hvis flere kanaler skal igang.
De 4 NOP (No Operation), som er indskudte mellem de 2 ordreord, er nødvendige, fordi der skal være en vis tid mellem disse.

## IN– OG OUTPUT

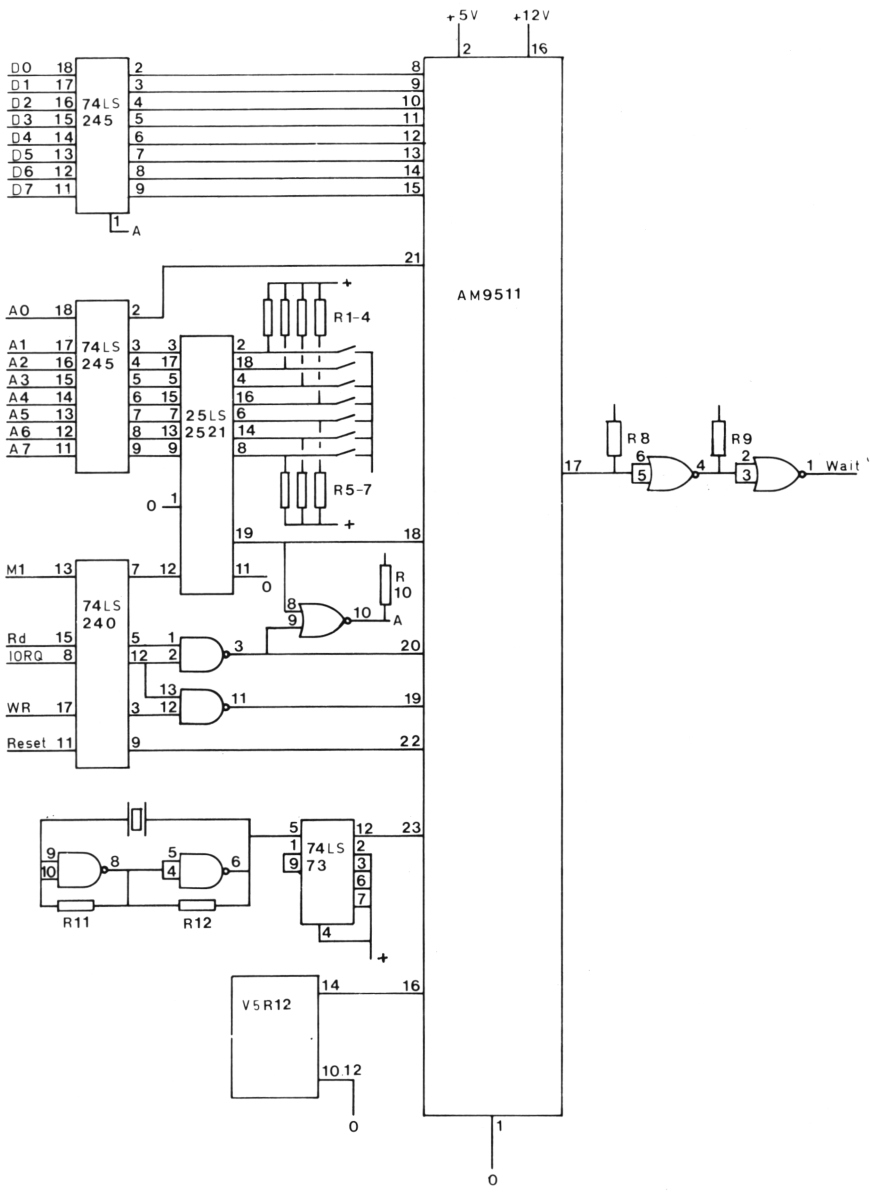
Den initialiserede kanal er nu klar til at arbejde, og overvåger konstant forbindelsen til terminalen for at se, om den sender noget, ligesom de data, den modtager fra CPU'en, straks videresendes. Der er imidlertid stadig et problem. CPU'en kan jo ikke vide, hvornår U–SART'en har en byte til den, ligesom CPU'en ikke kan vide, hvornår USART'en er færdig med at udsende den foregående byte.
For at få oplysning herom, kan CPU'en bede USART'en sende det såkaldte "statusord", som fortæller, om dens øjeblikkelige tilstand.
Også med hensyn til dette ord må vi henvis til databladene, idet vi dog her vil angive en standardrutine, som klarer forbindelsen mellem datamat og terminal. Igen er alle tal opgivet i Hex.

| ADR. | CODE | |
|---|---|---|
| 0010 | CD 17 00 | Call TTYRD |
| 0013 | E7 | RS7 TTYWR |
| 0014 | CB BF | RES PARITY, A |
| 0016 | C9 | RET |
| | | TTYRD: |
| 0017 | DB C1 | IN A, STATUS |
| 0019 | CB 4F | BIT TEST, A |
| 001B | 28 FA | JRZ, TTYRD |
| 001D | DB C0 | IN A |
| 001F | C9 | RET |
| | | TTYWR: |
| 0020 | F5 | PUSH AF |
| | | TTYST: |
| 0021 | DB C1 | IN A, STATUS |
| 0023 | CB 47 | BIT TEST, A |
| 0025 | 28 FA | JRZ, TTYST |
| 0027 | F1 | POP AF |
| 0028 | D3 C0 | OUT A |
| 002A | C9 | RET |

Dette program, som skal kaldes som rutiner fra hovedprogrammet, er samtidig en god illustration af, hvorledes de specielle Z-80 ordrer kan lette programmeringen.

## APU-KORT

Sidste gang skrev vi lidt om vore intentioner med hele dette projekt. For ligesom at understrege dette, vil vi denne gang se nærmere på en af de specielle funktioner, vi har indbygget i datamaten. Der er her samtidig tale om en


+5V
+12V
AM9511
74LS 245
74LS 245
25LS 2521
74LS 240
74LS 73
R1-4
R5-7
R8
R9
R10
R11
R12
V5R12
Wait
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
A5
A6
A7
M1
Rd
IORQ
WR
Reset

funktion, som kun meget få 8-bit datamater er udstyret med, fordi den hidtil har været meget kompliceret at opbygge, nemlig hard-ware matematik - også kaldet "number cruncher" eller "arithmetic processor unit".

## LIDT HISTORIE

For et par år siden var det en almindelig indstilling, at alt kunne - og burde - laves i software, fordi det var let at ændre. Til gengæld måtte man så accepterer, at datamaten blev langsom til at udføre visse funktioner. Et eksempel på dette er talbehandling. 8-bit maskiner er i virkeligheden ikke gode til at udføre aritmetiske operationer, fordi 8 bit kun giver 256 forskellige muligheder. Når der skal arbejdes med tal større end dette, er det derfor nødvendigt at bruge flere byte til at repræsentere hvert tal. De fleste BASICer anvender således 4.
Dette komplicerede de aritmetiske programmer og gør datamaten langsom til arbejde med tal.
Som et eksempel på, hvor langsomt det går, kan nævnes, at vi har et program til beregning af elektroniske kredsløb, hvori indgår transistorer og dioder. Programmet opstiller matricer for knudepunktsspændingerne og udregner derefter disse gennem et meget stort antal multiplikationer og additioner.
Det tager tid. Så lang tid, at det godt kan tage 1/2 time at beregne et netværk med ca. 10 komponenter.
Sådanne problemer har efterhånden ført til en erkendelse af at soft- og hardware må tilpasses hinanden og samarbejde, så en given opgave løses bedst muligt.
Denne udvikling har betydet, at de forskellige fabrikanter er begyndt at udsende kredse, som er beregnet til at hjælpe CPU'en med at udføre ganske specielle funktioner. I virkeligheden er der tale om små selvstændige datamater, som er indrettet optimalt til dette. Et af de foreløbige højdepunkter i denne udvikling skete for ca. 3 måneder siden, hvor Advanced Micro Devices blev i stand til, som de første, at levere en "Arithmetic Processing Unit" (APU) i én kreds. Kredsen er særdeles effektiv, idet den kan arbejde med alle 4 regningsarter samt logaritmiske- og trigonometriske funktioner - og det med en hastighed, som man påstår, er 100 til 200 gange hurtigere end hvis funktionerne udføres med software. Om denne påstand er rigtig, har vi desværre ikke været i stand til at måle, men væsentlig hurtigere er den i alle tilfælde.

### STYKLISTE FOR APU-KORT

| | |
|---|---|
| 1 stk. IC1 | AM9511 |
| 1 stk. IC2 | 74LS73 |
| 1 stk. IC3 | 74LS33 |
| 1 stk. IC4 | 74LS00 |
| 1 stk. IC5 | 25LS2521 |
| 2 stk. IC6, IC8 | 74LS245 |
| 1 stk. IC7 | 74LS240 |
| 1 stk. DC-DC converter | Reliability V5R12 |
| 9 stk. R1-R9 | 2K2 Ohm |
| 2 stk. R10-11 | 470 Ohm |
| 1 stk. X-tal | 8MHz |
| 7 stk. CX | 0,1 µF |
| 2 stk. C1-2 | 22 µF/16V Fast tantal |
| 1 stk. Pintkort | 78088 |
| 1 stk. DIL switch | 7 polet |

## PRINCIP

Når en sådan speciel funktion skal bringes til at arbejde sammen med CPU'en, er der normalt 3 principper, der kan vælges imellem, nemlig interrupt, DMA eller normal I/0 teknik. Hvilken af disse muligheder, der vælges, afhænger af i hvilke situationer, kortet forudses anvendt.
Vi har for dette kort regnet med, at det skal anvendes i forbindelse med et højniveausprog. Disse sprog foretager en opløsning af de matematiske udtryk, og når denne er færdig, beregnes udtrykket. Under denne beregning har CPU'en ikke rigtigt noget fornuftigt, den kan foretage sig, og vi har derfor valgt at lade den vente i en programsløjfe, medens APU'en knokler.

Dette betyder, at vi kan bruge den simpleste sammenkoblingsform, nemlig normal I/O-teknik. Sagt med andre ord, er de 2 enheder sammenkoblet gennem en normal I/O-port.

## DIAGRAMMET

De, der har fulgt vor artikelserie vil næppe blive særligt overraskede i den anledning. Gennem de sædvanlige buffere føres datalinierne og adresselinie A0 frem til APU'en. Adresselinierne A1 til A7 føres, sammen med kontrollinien M1 til en komparator, medens RD og WR kombineres med I0RQ og føres til APU'en. Komparatorens udgang kontroller, sammen med RD signalet, retningen af databufferne.
I visse situationer skal APU'en bruge lidt tid. Dette giver den besked om på en særlig udgang kaldet "Pause". Signalet herfra føres til CPU'en via WAIT-linien.
Den aritmetiske kreds skal også have et kloksignal. Dette skal være på enten 2 eller 4 MHz, afhængig af hvilken udgave, du har købt. Dette signal frembringes af en 8MHz klokgenerator, hvis signal 2 eller 4-deles. Der vælges mellem disse 2 muligheder med en lus.
Til sidst er der et lille spændingsproblem. APU'en skal have både + 5 og + 12 Volt. Det sidste klares med en på kortet anbragt DC-DC-konverter, så kortet kun skal tilføres + 5V.

## SOFTWARE

Som tidligere nævnt "taler" CPU og APU sammen via en I/O port. Dette betyder, at CPU'en først spørger via en input-instruktion, om APU'en er klar, og når dette er tilfældet, sendes de 4 eller 8 byte via output-instruktioner - som skal beregnes og til sidst sendes kommandobyten, som fortæller, hvilken type instruktion, der skal udføres. CPU'en går derefter ind i en sløjfe, hvor den hele tiden spørger på status. Når denne viser, at beregningerne er færdige, tilbagelæses resultatet via input-instruktioner.

# PROGRAMMER

## SÆTNINGER OG PROGRAMLINIER

Vi har i det foregående set en **sætning**, som denne:

```
LET H = 5
```

og vi har set, hvad den består af (side P-56). Denne sætning kan vi få **udført** på to forskellige måder. Vi kan indtaste den, som den står her, og derpå trykke på terminalens 'RETURN'-taste. I så fald bliver sætningen udført med det samme af BASIC-systemet, og tallet 5 kan bagefter genfindes under navnet H i systemets **arbejdslager**. Man kan imidlertid også vælge at indtaste den således:

```
10 LET H = 5
```

Når man derpå trykker på 'RETURN'-tasten, bliver sætningen gemt i arbejdslageret, men **ikke udført før efter nærmere ordre.** Tallet foran sætningen kaldes et **linienummer** og linienummeret sammen med den efterfølgende sætning kaldes en **programlinie** (eller blot en linie).

Det er vigtigt at skelne mellem sætning og linie, idet en linie i visse tilfælde kan indeholde **flere sætninger.**

Et BASIC-program består af én eller flere **programlinier**, og vi kan begynde med at se på følgende eksempel:

```
0010 LET H=5
0020 LET G=20
0030 LET A=G*H
```

Det lille program består åbenbart af 3 linier med hver sin tildeling. Nummereringen 10, 20 & 30 er valgt med overlæg, og vi skal se nedenfor, hvilke grunde man kan have hertil. Hver linie er indtastet ved terminalen og afsluttet med et tryk på 'RETURN'-tasten. Når man ønsker, at BASIC-systemet skal **udføre** et program, skal man indtaste ordet "RUN" og trykke på 'RETURN'-tasten.

De enkelte linier bliver derpå udført i **nummerorden**, og når udførelsen er slut, skriver systemet et tegn og evt. en tekst, der viser, at det er færdigt med opgaven og klar til at udføre en ny. Denne såkaldte **klarmelding** ser forskellig ud for de forskellige BASIC-versioner. Meldingen herunder er fra RC-BASIC:

```
END
AT 0030
* RUN
```

## REDIGERING AF PROGRAMMER

Lad os vende tilbage til vort lille program fra forrige afsnit, og lad os tænke os, at vi ønsker at **rette** sætningen i linie 20 til:

```
LET G = 20
```

Man indtaster da blot følgende linie:

```
0020 LET G=20
```

og trykker på 'RETURN'-tasten. Derved slettes den tidligere linie 20 og **erstattes** af den nye. Man siger også, at den gamle linie 20 er blevet **overskrevet** af den nye udgave af linien. Det er altså uhyre simpelt at rette en linie. Man skriver blot en ny med **samme linienummer** og trykker

på 'RETURN'-tasten, så bliver den gamle linie udskiftet med den nye.
Hvis man ønsker at **fjerne en linie** fra et program, skal man blot skrive liniens nummer og derpå trykke på 'RETURN'-tasten. Man kan evt. forestille sig, at man derved har indtastet en **tom sætning** til at erstatte den, der stod på linien i forvejen.
I mange tilfælde ønsker man også at **indsætte** nye linier blandt de allerede givne i et program. Lad os som eksempel indføje følgende sætning i programmet ovenfor:

```
LET T = 0
```

mellem sætninger i linie 10 og 20. Dette kan gøres ved at man f.eks. skriver:

```
15 LET T = 0
```

og trykker på 'RETURN'-tasten. Systemet **indsætter** nu den nye linie blandt de allerede givne, således at **nummerordenen bevares**. Det kan vi få bekræftet ved at indtaste ordet "LIST" og trykke på 'RETURN'-tasten. Derefter skriver systemet en **programliste**:

```
* LIST
  0010 LET H=5
  0015 LET T=0
  0020 LET G=20
  0030 LET A=G*H
```

Ordene "RUN" og "LIST" kaldes **systemkommandoer** eller blot **kommandoer**, og vi skal efterhånden lære flere sådanne kommandoer at kende. I modsætning til programsætninger udføres kommandoer altid straks og kan i almindelighed **ikke** indsættes i programmer med linienumre.
Nummereringen 10, 20 & 30 er naturligvis valgt, så man evt. senere kan få plads til at indsætte nye linier i programmet. I nogle BASIC-versioner findes en kommando "RENUMBER", der bevirker, at alle linienumre i programmet normaliseres til 10, 20, 30, . . ., uanset hvad der


har stået i forvejen. Hvis programmet ovenfor bliver modificeret ved brug af en RENUMBER-kommando, ser det sådan ud bagefter:

```
* LIST
  0010 LET H=5
  0020 LET T=0
  0030 LET G=20
  0040 LET A=G*H
```

På den måde kan man altid skaffe sig ekstra „luft" mellem linierne i et allerede skrevet program. Dette kan være nyttigt, hvis man f.eks. i et bestemt afsnit er nødt til at indsætte mange ekstra linier.

Lad os kort sammenfatte:

Et **program** består af én eller flere **programlinier.**
En **programlinie** begynder med et **linienummer** efterfulgt af én eller flere **programsætninger.**
En programlinie kan **ændres** ved at man indtaster en ny linie med **samme nummer,** som den, der skal ændres. Den nye linie vil da erstatte den givne.
En programlinie kan **slettes** ved at man indtaster dens nummer efterfulgt af et tryk på 'RETURN'-tasten.
Nye linier kan **indsættes** og vil af systemet blive anbragt således, at **nummerordenen** opretholdes.
Når programmet **udføres,** sker dette linie for linie i **nummerorden.**
Når kommandoen "RUN" indtastes, **udføres** programmet.
Når kommandoen "LIST" indtastes, får man **udskrevet en liste** over programmet.

**Bemærkning.** I mange BASIC-versioner kan man udvide kommandoerne LIST og RUN. Således kan f.eks. følgende:

```
LIST 30
```

betyde, at **kun linien med nr. 30** skal listes, mens f.eks.

```
RUN 30
```

kan betyde, at udførelsen af programmet skal **begynde med linie 30.**
Den slags modificerede kommandoer kan være meget nyttige, og læseren bør konsultere manualen over den BASIC-version, der er til rådighed, for sådanne faciliteter.

## 6. UDDATA–SÆTNINGER.

Det er naturligvis ikke til megen nytte, at man har fået tildelt værdier til én eller flere variable, hvis man ikke kan få disse værdier skrevet. I så fald ville datamaten blot bevare dem som en hemme-

lighed i sit arbejdslager, og det kan højst have interesse for filosoffer eller psykologer med helt specielle fagområder. Naturligvis har man i alle programmeringssprog nogle sætninger, der bevirker, at resultater skrives. I BASIC har man sætninger som den følgende til rådighed:

```
PRINT A
```

Når denne sætning udføres, bliver **værdien** af A skrevet på terminalen. En sætning, der indledes med ordet PRINT, kaldes også en **unddata-sætning**, og resultatet af udskriften kaldes **uddata**. Man kan have flere uddata efter ordet PRINT. Således vil en sætning som denne:

```
PRINT H,G,A,
```

bevirke, at værdien af de variable H, G og A udskrives. Udseendet af denne udskrift kan variere noget fra system til system, men som regel vil det være sådan, at der sættes et bestemt antal skrivepositioner mellem hver påbegyndt udskrift. Man kan fx. have følgende:

```
END
AT 0040
* RUN
 23.5          17.8          418.3
```

fra sætningen ovenfor, hvis H, G og A er tildelt de tre anførte værdier i den angivne rækkefølge.
Man kan også anføre tal **direkte** i en uddata-sætning, og man kan skrive **udtryk**, hvis værdi i så fald bliver **udregnet** og **udskrevet**, når sætningen udføres. Lad os fx. tænke os, at H og G har de værdier, som er vist ovenfor, og at vi derpå får udført følgende sætning:

```
PRINT H,G,H x G
```

I det tilfælde får vi den samme udskrift som er vist herover, idet udtrykket H x G bliver udregnet til værdien 23.5 x 17.8, altså 418.3.

Foruden tal, variable og formler kan man også indsætte **tekster** i PRINT-sætninger. Lad os som eksempel se på følgende lille program:

```
* LIST
  0010 LET H=23.5
  0020 LET G=17.8
  0030 PRINT "NAAR HOEJDEN ER:",H
  0040 PRINT "OG GRUNDLINJEN ER:",G
  0050 PRINT "BLIVER AREALET:",H*G
```

Når det program bliver udført, får man følgende udskrift:

```
END
AT 0050
* RUN
NAAR HOEJDEN ER:          23.5
OG GRUNDLINJEN ER:        17.8
BLIVER AREALET:           418.3
```

idet de tre tekster bliver skrevet med ud på **nøjagtig den form**, som de har fået ved indtastningen. Den lidt besynderlige stavemåde skyldes, at terminalen ikke har danske bogstaver til sin rådighed. På de fleste nyere terminaler findes danske bogstaver, men mange amatører har næppe råd til at købe disse og må nøjes med brugte. I så fald må man klare sig som vist herover.

Brugen af komma som skilletegn mellem uddata i PRINT-sætninger kan give anledning til, temmeligt roded udskrifter, og det er ikke sikkert, det falder i brugerens smag. Derfor kan man også i stedet bruge tegnet **semikolon** (;) som skilletegn mellem de enkelte data:

PRINT H;G;H x G

Denne sætning giver med værdierne ovenfor følgende udskrift:

```
END
AT 0050
* RUN
 23.5  17.8  418.3
```

Hvis vi i det lille program ovenfor bruger semikolon i stedet for komma efter teksterne, får vi følgende udskrift:

```
END
AT 0050
* RUN
NAAR HOEJDEN ER: 23.5
OG GRUNDLINJEN ER: 17.8
BLIVER AREALET: 418.3
```

Med semikolon mellem de enkelte data har man som regel bedre styr på udskriften, end når man anvender komma, og man bør derfor foretrække semikolon undtagen måske ved udskrifter, der kun består af rene talværdier uden forklarende tekster.

Ved at udnytte at **mellemrum (blanktegn)** naturligvis optræder på lige fod med alle andre tegn, kan man fremtvinge særdeles velordnede udskrifter. Lad os fx. se på følgende program, der er fremkommet ved at rette på det,

```
* LIST
  0010 LET H=23.5
  0020 LET G=17.8
  0030 PRINT "NAAR HOEJDEN ER:     ";H
  0040 PRINT "OG GRUNDLINJEN ER:   ";G
  0050 PRINT "BLIVER AREALET:      ";H*G
```

Programmet giver denne udskrift:

```
END
AT 0050
* RUN
NAAR HOEJDEN ER:    23.5
OG GRUNDLINJEN ER: 17.8
BLIVER AREALET:     418.3
```

**Formatet** af udskriften, er helt i hænderne på programmøren og bestemmes ikke af mere eller mindre tilfældige luner hos den, der har lavet systemet.
Efter en uddata-sætning følger normalt et **linieskift** (ny linje). Dette kan man forhindre ved at sætte et **komma** eller et **semikolon efter det sidste datum** på linjen. Ændrer vi således linje 30 og 40 i programmet herover til:

```
* LIST 30,40
  0030 PRINT "NAAR HOEJDEN ER:";H;
  0040 PRINT "OG GRUNDLINJEN ER:";G
```

Får man denne udskrift, når det udføres:

```
END
AT 0050
* RUN
NAAR HOEJDEN ER: 23.5 OG GRUNDLINJEN ER: 17.8
BLIVER AREALET:      418.3
```

Et komma efter den variable H i linje 30 har næsten samme virkning, idet der dog kan være større afstand mellem udskriften af talværdien af H og den tekst, der er anført i linje 40, på helt samme måde som et komma i datalisten i almindelighed giver større afstand mellem de implicerede udskrifter end semikolon.
Der kan være visse forskelle mellem de forskellige systemer med hensyn til virkningen af kommaer i uddata-sætninger. Man bør derfor se efter i den manual, der følger med systemet, for at finde ud af, hvilke regler der gælder for udskrifter. Har man ingen manual - ikke så utænkeligt som man skulle tro - må man simpelthen prøve sig frem for at finde ud af, hvilke regler der gælder for det aktuelle system.

Man kan have **flere tekster** i samme uddata-sætning, og ind imellem disse tekster kan man anbringe konstanter, variable eller udtryk. Lad os slutte afsnittet med følgende eksempel:

```
* LIST
  0010 LET T=20
  0020 LET H=3
  0030 PRINT "PAA";T;"MIN. KAN OLE OLSEN KOERE";T*H;"KM."
  0040 PRINT "HAN ER ALTSAA HURTIG."
```

Når det program udføres, får man denne udskrift:

```
END
AT 0040
* RUN
PAA 20 MIN. KAN OLE OLSEN KOERE 60 KM.
HAN ER ALTSAA HURTIG.
```

# PROGRAM: Omregning mellem talsystemer

Dette program til TI 58/TI 59 er fremstillet af Kim Sparre i Langå, og kan bruges af bl.a. folk, som ikke er i besiddelse af en assembler til deres mikrodatamat.

Dette program for TI 58/TI 59 omregner mellem 10-talsystemet og et talsystem med grundtallet b ($2 \leqslant b \leqslant 100$), idet hvert ciffer i b-talsystemet repræsenteres af et eller to cifre henholdsvis for grundtal mindre end eller lig 10 og for grundtal større end 10.

- b grundtal(base)
- m multiplikator (m=10 for $b \leqslant 10$, m=100 for $b > 10$).
- $c_i$ ciffer i
- i tæller som indikerer cifrets plads
- $A_{10}$ tal i 10-talsystemet
- $A_b$ tal i b-talsystemet

$A_{10} \rightarrow A_b$:

$c_i = \text{mod}(A_{10}, b)$

$A_{10} = \text{Int}(A_{10} \div b)$

$A_b = A_b + c_i b^i$

$A_b \rightarrow A_{10}$:

$c_i = \text{mod}(A_b, m)$

$A_b = \text{Int}(A_b \div m)$

$A_{10} = A_{10} + c_i 10^i$

Først elimineres en eventuel decimaldel ved gentagen multiplikation med b (m), idet i samtidig tælles lige så mange gange ned. Når der ikke længere er nogen decimaldel tilbage, omregnes A efter formlen ovenfor, idet i tælles op hver gang, der divideres med b (m), og der stoppes, når $A_{10} = 0$ ($A_b = 0$).

A: grundtal
B: $A_{10} \rightarrow A_b$
C: $A_b \rightarrow A_{10}$

```
BASE CONVERSIONS

   13/04/78

000   76 LBL
001   11  A
002   42 STO
003   01  01
004   32 X:T
005   01  1
006   00  0
007   77  GE
008   00  00
009   11  11
010   33 X²
011   42 STO
012   02  02
013   32 X:T
```

```
014  99 PRT      056  43 RCL
015  98 ADV      057  01  01
016  91 R/S      058  95  =
017  76 LBL      059  65  ×
018  12  B       060  43 RCL
019  99 PRT      061  02  02
020  42 STO      062  45 Y^
021  03  03      063  43 RCL
022  32 X:T      064  00  00
023  00  0       065  95  =
024  42 STO      066  44 SUM
025  00  00      067  04  04
026  42 STO      068  69 DP
027  04  04      069  20  20
028  32 X:T      070  43 RCL
029  22 INV      071  03  03
030  59 INT      072  22 INV
031  67  EQ      073  67  EQ
032  00  00      074  00  00
033  70  70      075  45 .45
034  43 RCL      076  43 RCL
035  01  01      077  04  04
036  49 PRD      078  99 PRT
037  03  03      079  98 ADV
038  69 DP       080  92 RTN
039  30  30      081  76 LBL
040  43 RCL      082  13  C
041  03  03      083  48 EXC
042  61 GTO      084  01  01
043  00  00      085  48 EXC
044  29  29      086  02  02
045  75  -       087  48 EXC
046  53  (       088  01  01
047  24 CE       089  71 SBR
048  55  ÷       090  12  B
049  43 RCL      091  48 EXC
050  01  01      092  01  01
051  54  )       093  48 EXC
052  59 INT      094  02  02
053  42 STO      095  48 EXC
054  03  03      096  01  01
055  65  ×       097  91 R/S
```

# PROGRAM: Mastermind

1 Indlæs hele programmet
2 Vælg et tilfældigt tal $0 < x < 10$ med mindst 1 decimal, og tryk: STO 7.
3 Tryk E (hemmeligt tal findes)
4 Tag et gæt, f.eks. 637, og tryk A.

POINT:
0 for ingen rigtige tal
1 for hvert rigtigt tal, forkert placeret
4 for hvert rigtigt tal, rigtigt placeret

Ved nyt spil, blot repetér step 3 og 4

Det hemmelige tal består af tre forskellige cifre, dog ej nul. (Nul fjernes ved blok 182-183).

Eks.: $x = \sqrt{\pi}$

| indtast | udlæsning |
|---|---|
| 347 | 1 |
| 138 | 5 |
| 418 | 1 |
| 193 | 8 |
| 153 | 12 |
| | |
| 543 | 0 |
| 982 | 4 |
| 872 | 8 |
| 172 | 8 |
| 672 | 12 |

## ET PAR ORD OM MASTERMIND

000-013 Integer
014-021 Point (4)
022-067 Det indtastede tal splittes op i de enkelte elementer, og lagres hver for sig.
068-075 0 lagres i reg. 08 og reg. 18, samtidig hejses flag 0.
076-116 Tallenes enkelte elementer sammenlignes parvis.
(xyz = hemmeligt tal. RST = gættet tal)
z - T hvis 0 springes til 014 (4 point)
y - S hvis 0 springes til 014 (4 point)
x - R hvis 0 springes til 014 (4 point)
117-154 Vi stryger flaget, og samtidig rokeres RST til TRS.
Den påfølgende sammenligning giver da kun 1 point for hvert ens talpar:
z - S hvis 0 1 point
y - R hvis 0 1 point
x - T hvis 0 1 point
Den sidste sammenligning ser således ud:
z - R hvis 0 1 point
y - T hvis 0 1 point
x - S hvis 0 1 point
STR vender dog tilbage til sit oprindelige udseende (RST) før programslut.
155-189 3 hemmelige tal-elementer findes og lagres hver for sig (xyz). Hvis blot et af dem er 0 begyndes forfra.
190-221 De tre tal-elementer skal være indbyrdes forskellige, det undersøges ved følgende sammenligning:
x - y hvis 0 forfra
x - z hvis 0 forfra
y - z hvis 0 forfra

MASTERMIND 2

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | LBL | 045 | E′ | 090 | 0 | 135 | 4 | 180 | = |
| | E′ | | IND | | 0 | | EXC | | E |
| | INV | | STO | | - | | 0 | | if zro |
| | EE | | 1 | | IND | | 5 | | 1′ |
| | INV | | 9 | | RCL | | STO | | IND |
| 005 | DMS | 050 | INV | 095 | 1 | 140 | 0 | 185 | STO |
| | INV | | SUM | | 9 | | 4 | | 0 |
| | DMS | | 0 | | = | | RCL | | 0 |
| | fix | | 9 | | INV | | 1 | | dsz |
| | 0 | | 1 | | if zro | | 8 | | 1′ |
| 010 | DMS | 055 | 0 | 100 | 6′ | 145 | - | 190 | RCL |
| | INV | | PROD | | if flg | | 3 | | 0 |
| | fix | | 0 | | 0 | | = | | 1 |
| | rtn | | 9 | | 5′ | | INV | | - |
| | LBL | | RCL | | 1 | | if zro | | RCL |
| 015 | 5′ | 060 | 1 | 105 | SUM | 150 | 3′ | 195 | 0 |
| | 4 | | 9 | | 0 | | RCL | | 2 |
| | SUM | | - | | 8 | | 0 | | = |
| | 0 | | 6 | | LBL | | 8 | | if zro |
| | 8 | | = | | 6′ | | rset | | E |
| 020 | GTO | 065 | INV | 110 | 1 | 155 | LBL | 200 | RCL |
| | 6′ | | if zro | | INV | | E | | 0 |
| | LBL | | 2′ | | SUM | | 3 | | 1 |
| | A | | STO | | 1 | | STO | | - |
| | ÷ | | 0 | | 9 | | 0 | | RCL |
| 025 | 1 | 070 | 8 | 115 | dsz | 160 | 0 | 205 | 0 |
| | 0 | | STO | | 4′ | | LBL | | 3 |
| | 0 | | 1 | | 1 | | 1′ | | = |
| | = | | 8 | | SUM | | RCL | | if zro |
| | STO | | st flg | | 1 | | 0 | | E |
| 030 | 0 | 075 | 0 | 120 | 8 | 165 | 7 | 210 | RCL |
| | 9 | | LBL | | INV | | + | | 0 |
| | 3 | | 3′ | | st flg | | 2 | | 2 |
| | STO | | 3 | | 0 | | = | | - |
| | 1 | | STO | | RCL | | $y^x$ | | RCL |
| 035 | 9 | 080 | 0 | 125 | 0 | 170 | 9 | 215 | 0 |
| | LBL | | 0 | | 5 | | - | | 3 |
| | 2′ | | 6 | | EXC | | E′ | | = |
| | 1 | | STO | | 0 | | = | | if zro |
| | SUM | | 1 | | 6 | | STO | | E |
| 040 | 1 | 085 | 9 | 130 | STO | 175 | 0 | 220 | 0 |
| | 9 | | LBL | | 0 | | 7 | | rset |
| | RCL | | 4′ | | 5 | | × | | |
| | 0 | | IND | | RCL | | 1 | | |
| | 9 | | RCL | | 0 | | 0 | | |

## MICROCOMPUTER SYSTEM DESIGN KIT SDK-85

Den hastige udvikling inden for den integrerede teknik har minimeret microcomputeren så meget, at vi i dag ikke taler øm den som et system,- men som en KOMPONENT.

Den kraftige integrering betyder samtidigt væsentlige prisreduktioner og stærkt simplificeret produktdesign samt i sidste instans et billigt og bedre produktDette indebærer, at microcompu bliver den helt naturlige grundkomponent inden for hele styrings- og logikområdet, - ja, den breder sig over hele automationsspektret og den kommercielle elektronik. I mange tilfælde som den eneste komponent i det pågældende system. Alt dette betyder, at flere og flere må erhverve sig viden om microcomputeren.

INTEL's MCS-85 SYSTEM DESIGN KIT, SDK-85 er det rigtige indlæringsværktøj.

Dette kit er en komplet Single Board Computer indeholdende CPU, hukommelse og input/ output linier, Monitor og betjeningspanel med numerings display og tryktastatur. Altsammen på eet printkort.

SDK-85's Monitor indeholder kommandoer som Programeksekvering, Single Step, Examine/Modify registre, Substitute/Display/Move memory og Insert Instruktions.

CPU'en er INTEL's avancerede 8085 microprocessor.

Næsten halvdelen af printkortet er reserveret til udbygning med egne kredsløb, der kan monteres med wire-wrap teknik ( eller loddekolbe).

Nærmere oplysninger fås ved henvendelse til LYNGSØ KOMPONENT Søborg TLF. (01) 67 00 77.

---



1/1 side annonce Pops Elektronik

DET KUNNE JEG TÆNKE MIG AT LÆSE MERE OM:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

**KLIP LANGS DE FULDT OPTRUKNE STREGER SENDES SOM BREVKORT, HUSK PORTO**

**Giv dette brevkort til en ven, som gerne selv vil have sit eget eksemplar af Håndbog for Datamat-amatører, eller brug det til meddelelser angående flytning.**

HFD udkommer med 11 numre om året, og et abonnement kan tegnes når som helst og med start fra valgfri måned. Det anbefales dog, at der tegnes abonnement fra bladets start, da den specielle opbygning bedst udnyttes, hvis alle numre haves. 1. nummer udsendt er nr. 9/1977. Abonnement koster idag kr. 100,– for en årgang og inkluderer 11 numre af HFD, 1 praktisk og solidt ringbind til en hel årgang, porto og moms.

**Undertegnede bestiller herved et abonnement på HFD for 1 år i henhold til ovenstående for kr. 100,–.**
**Jeg ønsker at abonnementet starter med nr.** . . . . . .

☐ Beløbet, kr. 100,–, vedlægges i check.

☐ I bedes fremsende girokort.

**Undertegnede ønsker at meddele adresseforandring på mit abonnement på HFD.**

Navn: . . . . . .

Gl. gadeadresse: . . . . . .

Gl. postnr. og by: . . . . . .

Ny gadeadresse: . . . . . .

Nyt postnr. og by: . . . . . .

Ovenstående adresseændring træder i kraft d. . . . . .

Andet: . . . . . .

. . . . . .

BREV

Husk afsender

Porto
120
øre

Til:

Telepress ApS

Greve Strandvej 42
2670 Greve Strand

**KLIP LANGS STREGERNE HELT TIL BLADETS KANT**

BREV

Husk afsender

Porto
120
øre

Til:

Telepress ApS

Greve Strandvej 42
2670 Greve Strand