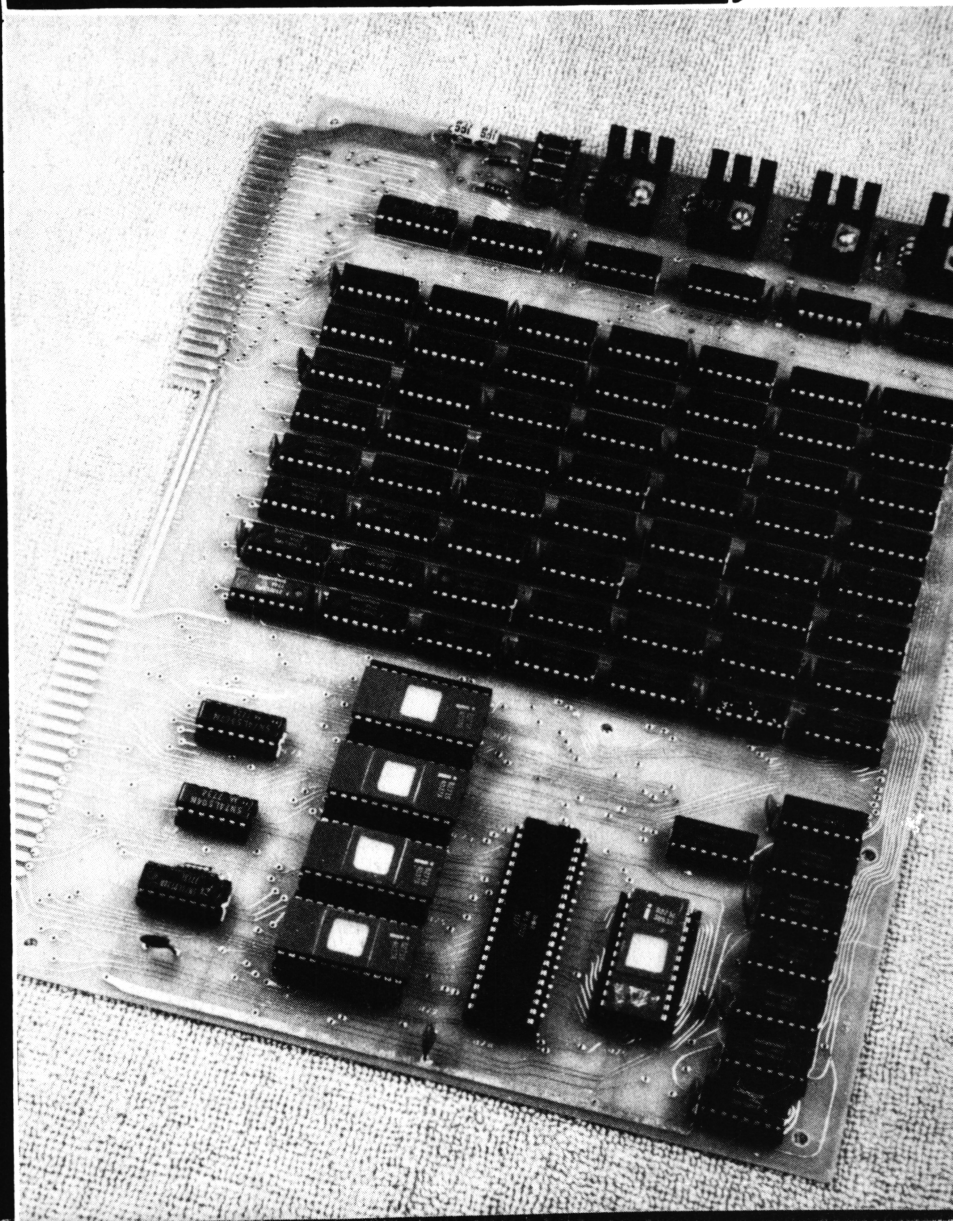
6/7 Håndbog for datamat-amatører

HRO 1978



INDHOLDSFORTEGNELSE

FORSIDEN:
Dette kort, som minder om en KIM-1, er i virkeligheden et ekstra lager til KIM-1. Kortet, der benævnes MEMORY Plus, har 16K dekodning og 8K RAM monteret samt sokler til 8K EPROM. Endvidere er EPROM-programeringskredsløbet inkluderet. Dette udvidelseskort, som kommer fra Micronor (tlf. (06)83 60 08) koster kr. 2.190 excl. moms.

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.

Alment om programmerbare maskiner	A
Biblioteket - programmer	B
CPU-arkitektur	C
Datamat-litteratur	D
Interfacing	I
Klubinformation	K
Lommeregnere	L
Microdatamater	M
Programmeringsteknik	P
Selvbyggerprojekter	S
Tilbud fra læserne	T
Undervisningsudstyr	U
Ydre enheder	Y

Er der ord, som du ikke forstår? Et program til din KIM-1, som driller? Savner du et billigt tastatur? Vil du gerne køre Fortran på en IMSAI 8080? Har du problemer med file-håndtering på disc? Brug sommerferien til at fortælle os om det!

Vi ved, at vore læsere befinder sig på vidt forskellige stadier og med lige så forskellig baggrund og interesse.
Vi tilstræber derfor at medtage så mange generelle oplysninger som muligt her i HFD.
Men vi vil gerne vide en smule mere om, hvor „skoen trykker" hos den enkelte læser, så vi bedst muligt kan tilgodese de flestes behov.
Det ene af brevkortene er denne gang udformet, så der er plads til adskillige ord om aktuelle problemer, punkter, der fortjener yderligere belysning, interfacing, som driller o.s.v.
Vi kan ikke på forhånd garantere, at vi kan få plads til at løse ethvert problem og besvare alle spørgsmål i løbet af efteråret, men vi vil sætte ind dér, hvor der er størst behov.
Benyt derfor sommerferien til at nedfælde på brevkortet på bagsiden, hvad du kunne tænke dig at se belyst her i HFD – både generelt og specifikt – så skal vi gøre, hvad vi kan, for at du får et endnu bedre blad.

Vi minder forresten om, at dette nummer dækker både juni og juli, og at det næste nummer af HFD udsendes den 1. torsdag i august.

RIGTIG GOD SOMMERFERIE

PROGRAM: Reaktionstid

BASIC og BASIC er ikke 2, men hundrede forskellige sprog, som det fremgår af artiklen side P55. Det foranstående BASIC-program er beregnet til en Motorola TDS, som har en relativ simpel BASIC-fortolker. Nedenstående program er skrevet til en PET 2001, der benytter en særdeles avanceret BASIC, som i kraft af maskinens indbyggede timer giver mulighed for specielle løsninger. Programmet er samtidig illustration af brugen af instruktionen GET og RND, som gør det muligt at få datamaten til at udføre „tilfældige" procedurer.
Hvis dette program skal omskrives til andre maskiner, er det nødvendigt, at disse har mindst én timer indbygget, samt at både maskine og fortolker er rimelig hurtig.

Da vi havde rådighed over en PET 2001 i et par dage, kunne vi ikke dy os for at lave et lille program, som udnytter dens indbyggede timer og GET funktion.
Sidstnævnte er en inputfunktion, som minder stærkt om maskinkodefunktionen, hvor man med ordren GET blot får den sidst indtastede karakter fra tastaturet lagret i den ønskede variabel.
Ved den rette programmering kan GET benyttes som interupt fra tastaturet i det man med jævne mellemrum kalder en rutine, som i tilfælde af, at en indtastning har fundet sted, sørger for en passende aktion. Hvis ingen indtastning er sket, returneres til programmet.
Brugen af GET giver også mulighed for lettere behandling af visse former for input, idet de fleste BASIC-fortolkere vil gå ud af programmet, hvis alfanumeriske tegn indtastes i forbindelse med en almindelig INPUT instruktion.
Ligeledes vil GET ikke automatisk give udtastning af et ?, hvilket også kan være praktisk. Yderligere brug af GET er omtalt på side P19.

REAKTIONSTID

Da PET 2001 har indbygget timer (ur), kan man lade tid indgå som variabel og dermed også som funktion.
Der er indbygget ialt 2 ure, som kan kaldes hver for sig. Variablen TIME indeholder en 60 Hz generator, og hvis man kalder TIME/60, får man altså sekunder tilbage.
TIME$ er en string-variabel (alfanumerisk) med formatet "TTMMSS" = Timer, Minutter og Sekunder.
Nulstilling af TIME nulstiller kun denne, mens nulstilling af TIME$ nulstiller begge timere.
I efterfølgende program benyttes udelukkende TIME til flere forskellige funktioner, mens TIME$ anvendes til nulstilling.
Med sin nuværende opbygning er løkken, som måler reaktionstid, istand til at

registrere ned til 0,0167 sekund, så til dette formål er programmet rigeligt hurtigt. I praktisk kørsel har vi ikke været under 0,65 sekund, og de fleste gange har vi ligget på lige knap 1 sekund.
Hvis løkken, som registrerer tid, havde indeholdt flere instruktioner, så der var et større tidsforbrug i selve programmet, ville det have været en god idé at lægge denne del som en rutine forrest i programmet, så opsøgningen af adresserne tog mindst mulig tid.
Programmet måler reaktionstid ialt 10 gange og beregner derefter et gennemsnit for disse 10 forsøg, og det ville måske have været naturligt at etablere en stor løkke til at holde styr på disse 10 forsøg, istedet for selv at tælle op i K.
Erfaringen viser dog, at der under en stor løkke med mange operationer og nødvendige subrutinekald udenfor løkken let kan ske programmeringsmisforståelser, som resulterer i fejl, idet løkken ikke bliver gennemløbet korrekt. I tilfælde af programafsnit på 10 eller flere ordrer foretrækker vi derfor selv at etablere den nødvendige tælling.

PROGRAMMETS FUNKTIONER

Vi løber ganske let gennem programmet og ser på de lidt specielle funktioner, som kan være interessante.
10 - 40 indeholder startrutinen med kort forklaring af programmets virkemåde med instruktion til brugeren.
42 - 47 nulstiller et par variable. Ved at placere disse i denne rækkefølge gives der mulighed for at hoppe retur til forskellige nulstillinger.
50 - 160 indeholder en rutine for start

```
 10 PRINT: PRINT: PRINT "   »»» TEST AF REAKTIONSTID »»»"
 20 PRINT: PRINT: PRINT "SVAR MED J (JA), N NEJ) ELLER ET CIFFER"
 30 PRINT: PRINT "NÅR DER UDSKRIVES ET TAL, SÅ TRYK"
 40 PRINT: PRINT "DET SAMME TAL IND!"
 42 TIME$ = "000000"
 45 P = 0
 46 K = 0
 47 C$ = " "
 50 PRINT: PRINT "ER DU KLAR?" : PRINT
 70 T = TIME
 80 GET C$
 90 IF TIME > T + 500 THEN 150
 95 IF C$ = " " THEN 80
100 IF C$ = "J" THEN 200
110 IF C$ = "N" THEN 130
115 PRINT "VI HAR SPILLET I"; TIME/3600; "MINUTTER"
116 PRINT: PRINT "HUSK AT TRYKKE J ELLER N!"
120 GOTO 50
130 PRINT "NEJ – OK, VI VENTER LIDT!"
140 T = TIME
142 IF TIME > T + 1000 THEN 47
145 GOTO 142
150 PRINT "NU MÅ DU SNART BESTEMME DIG"
160 GOTO 50
200 IF K = 0 THEN PRINT "JA – SÅ BEGYNDER VI"
202 IF K > 0 THEN PRINT "JA – HER KOMMER NR."; K + 1
205 PRINT: PRINT
206 IF TIME < 54000 OR TIME > 60000 THEN 210
207 IF K <> 4 THEN 210
208 PRINT " HA – DER BLEV DU VIST NERVØS!": PRINT
209 PRINT " NU KOMMER NR. 5 FØRST:" : PRINT: PRINT
```

af selve kørslen. I 50 spørges, om spilleren er klar, og T sættes lig den daværende TIME.
I løkken 80 til 95 undersøges, om der dels er blevet indtastet en karakter, og om der er gået en rimelig mængde tid, siden der blev spurgt, om spilleren var klar.
Denne tidskontrol sker med ordren i 90, som tillader godt 8 sekunders betænkningstid. (500/60 = 8,33).
Hvis tiden udløber, hoppes til 150 og derefter op til 50 igen.
Hvis der indtastes en karakter, vil 100 og 110 sortere J (JA) og N (NEJ) fra.
Alle andre karakterer vil give en udskrift af tidsforløbet med erindring om, at der normalt skal benyttes J og N.
Tidsforløbet beregnes i selv udskriftsinstruktionen - 115 - med TIME/3600 = antal minutter siden nulstilling af TIME i programmets start.
Hvis der er indtastet N, gås der til en rutine i 130, hvor spilleren tillades ca. 16 sekunders betænkningstid.
I de fleste tilfælde vil spilleren have svaret J (JA), og der fortsættes i 200.
200 - 209 indeholder en udskriftrutine, som normalt siger "klar" kort inden der udskrives det tal, som spilleren skal efterligne, men hvis der har været spillet omkring et kvarter, kan det risikeres, at der kommer en drilsk kommentar (206 - 209).
Normalt udskrives fra 202, men første gang i hvert spil udskrives fra 200. K vil da være lige med 0.
Det ville ikke være smart, hvis man lærte, hvor lang tid, der gik, før et tal kom på syne på skærmen, så her benytter vi

```
210 T = TIME
220 A = INT(RND(1) * 500)
225 IF TIME < T + A THEN 225
240 A = INT(RND(1) * 10
242 IF A = 0 THEN 240
245 C = 0
250 PRINT TAB(INT(RND(1) * 40)); A
260 T = TIME
270 GET C
280 IF C = A THEN 300
285 IF C = 0 THEN 270
290 PRINT "FORKERT – SKYND DIG AT RAMME"; A: PRINT
295 GOTO 270
300 T = (TIME – T)/3 * 5
305 P = P + T
310 PRINT: PRINT T/100; "SEKUND": PRINT
320 K = K + 1
330 IF K < 10 THEN 50
340 PRINT: PRINT "DIT GENNEMSNIT FOR 10 FORSØG"
350 PRINT: PRINT "BLEV"; P/1000; "SEKUND": PRINT
351 IF P/1000 > 3 THEN PRINT "DET ER VIST IKKE DIN DAG!"
352 IF INT(P/1000) = 2 THEN PRINT "VAR DU UHELDIG? PRØV IGEN"
353 IF INT(P/1000) = 1 THEN PRINT "IKKE DÅRLIGT AF EN BEGYNDER"
354 PRINT
355 IF TIME > 54000 THEN PRINT "ER I KLAR OVER, AT DER ALLEREDE
    ER GÅET ET HELT KVARTER?"
356 IF TIME > 54000 THEN TIME$ = "000000"
357 IF P/1000 < 1 THEN PRINT "BRAVO – GODT KLARET!"
358 IF P/1000 < .9 THEN PRINT "EXCELLENT!!!"
359 IF P/1000 < .8 THEN PRINT "KAN IKKE GØRES BEDRE!!!"
360 GOTO 45
```

en kombination af tid og en tilfældighedsgenerator til at give et tilfældigt tidsforløb. Vi danner en tilfældig størrelse i A, og i 225 lader vi den ønskede tid forløbe.
Funktionen RND(1) vil give et tilfældigt tal mellem 0 og 1, og dette benyttes også i 240, hvor vi danner et tilfældigt tal mellem 0 og 10 - højst 9,999999.
Ved hjælp af INT og kontrollen i 242 sikrer vi os, at tallet ligger mellem 1 - 9 incl.
I 250 udskriver vi i en tilfældig position på linien det netop genererede tal - i tabposition 0 til 39.
Strukturen omkring instruktionen 210 - 250 er interessant, idet vi genererer et tilfældigt tal, som udskrives på et tilfældigt punkt på skærmen efter et lige så tilfældigt tidsforløb. Altsammen indenfor „aftalte" rammer. Normalt plejer der intet tilfældigt at være i forbindelse med datamater, men det er der altså i høj grad her.
Løkken omkring tidtagningen går fra 260 til 300. Der undersøges, om der indtastes et tal, og hvis dette er identisk med det, som datamaten sendte ud (A), beregnes tidsforbruget i 300. At gange forbruget med 3/5 giver os 100-dele sekunder, hvilket er lettere at vurdere i en udskrift end 60-dele sekunder.
I 305 summeres tidsforbruget for de 10 forsøg i P, og resultatet af det pågældende forsøg udskrives i 310.
K tælles op, og hvis vi ikke har haft 10 forsøg, går vi retur til 50.
Efter 10 forsøg beregnes gennemsnittet i 350 (P/1000 = 100 for sekunder og 10 for antal forsøg).
351 - 359 bringer kommentarer til det beregnede resultat.
I 355 og 356 ligger en lille spøgefuldhed, som højst vil dukke op 1 gang hvert kvarter.
I 360 returneres til nulstilling af tællere og klart til nyt forsøg.

Indledningen på programkørslen affotograferet fra skærmen på en PET 2001.
Efter spørgsmålet: „Er du klar?" trykkes N, og datamaten skriver selv hele næste linie. Efter andet spørgsmål bliver datamaten utålmodig og rykker for et svar: „Nu må du snart bestemme dig."
Endelig er operatøren klar, og det første tal dukker op: 9.

```
# # # TEST AF REAKTIONSTID # # #

SVAR MED J (JA), N (NEJ) ELLER ET CIFFER

NAAR DER UDSKRIVES ET TAL, SAA TRYK
DET SAMME TAL IND!

ER DU KLAR?

NEJ - OK, VI VENTER LIDT!

ER DU KLAR?

NU MAA DU SNART BESTEMME DIG

ER DU KLAR?

JA - SAA BEGYNDER VI

          9
```

af ialt 10 cifre, hvor de sidste 4 er adskilt fra de første med en bindestreg.
Når et sådant tal skal indlæses i maskinkode, må hvert eneste af disse 11 tegn indlæses for sig, og det er nødvendigt med en form for kontrol af det indtastede, så chancen for fejl minimiseres.
De forskellige datamater behandler sådanne inputrutiner på vidt forskellig måde, og dette er én af årsagerne til, at vi til dato har benyttet et pseudo-sprog, som skulle være umiddelbart forståeligt for alle.
Der har imidlertid været fremført klage over dette, og selvom vi finder dette en smule uberettiget, lytter vi naturligvis til vore læsere.
På den anden side vil vi ved at vælge én bestemt maskinkode lade praktisk talt alle andre i stikken, og vi har derfor til følgende indlæsningsrutine valgt en ret utraditionel løsning: Maskinkode i BASIC!
Dette kan lyde som en grov selvmodsigelse, men ved at opbygge programmet med få og separate algoritmer (instruktioner) får det samme karakter som maskinkode, og ved ligeledes at benytte instruktionen GET, der ganske svarer til den netop omtalte inputrutine GET-KEY fra KIM-1 og lign. datamater, vil programmet ikke alene få en maskinkodelignende struktur, men det vil være langt lettere at både løbe igennem og gennemskue.

INDLÆSNING AF ET CPR-NR.

Det efterfølgende program vil direkte kunne køre på en PET 2001 og andre datamater med BASIC indeholdende instruktionen GET samt med fuld kontrol over kursor (Den lille prik på TV-skærmen, som viser, hvor næste karakter bliver placeret).
Under indlæsningen sikrer vi os, at bindestregen bliver indtastet på det rigtige tidspunkt, men vi gemmer ikke denne karakter, da den vil være fælles for alle CPR-nr.
I BASIC vil man normalt gemme hele tallet i én celle, eller - hvis der ikke kan behandles 10 betydende cifre - i 2. I maskinkode er vi nødt til at „pakke" tallene sammen i BCD-form (Binary Coded Decimals), hvor der i hver celle kan ligge 2 decimalcifre.
Programmet gør det samme, og hvert CPR-nr. gemmes derfor i 5 ord.
005 - her afsætter vi plads til ét CPR-nr.
I maskinkode vil man blot nedskrive adressen på et stykke papir og huske den under indlæsningen - uanset kodeformen, må man huske at afsætte den nødvendige plads til sine variable.
10 - vi springer ned til linie 60, da vi af hastighedshensyn har selve indlæsningsproceduren liggende forrest. Dette har ingen betydning i maskinkode, men i BASIC vil fortolkeren hver gang søge en ny adresse forfra i lageret, så hyppigt anvendte rutiner og variable bør placeres forrest.
60 - Vi laver en vognretur = ny linie, så vi er sikre på, at vi ikke skriver ud på skærmen i allerede placeret information.
70 - 80 - A% og B% er to heltals-variable, som vil kunne indeholde værdier op til ca. 65.000. I maskinkode vil man i en 8-bit maskine kunne tælle op til 256 i ét ord, og det vil til dette formål også være tilstrækkeligt. Disse variable gives deres start-værdier. A% benyttes til både optælling og adressering, og B% bliver benyttet til især kontrol.
85 - Vi giver operatøren besked om, at vi er klar til at acceptere et CPR-nr.
86 - Vi giver et ekstra linie-skift for at få pæn afstand mellem sidste budskab og de næste udskrifter.
90 - Vi etablerer en løkke, som tæller med 1 ad gangen fra A% til B%. I maskinkode vil en enkelt celle være nok til J, og man skulle tro, at en variabel som J% ville være mere passende end J, men pudsigt nok accepterer de færreste BASIC-fortolkere heltals-variable i denne del af step-funktionen - derfor J.
Når vi begynder løkken, vil værdien af A% og B% betyde, at vi stepper fra 1 til 3 med 1 pr. step.
100 - Her danner vi en ny løkke, som går fra 1 til 2. Når 2 løkker ligger inden i hinanden, vil den inderste blive

udført sit nødvendige antal gange for hver gang den yderste påbegyndes igen. Vi kommer tilbage til K-løkken.

110 - Nu springer vi ud af løkken til vor indlæsningsrutine forrest i programmet.

20 - Vi begynder med at tælle op i variablen P. Den burde af pladshensyn have været defineret som P%, men det blev overset under programmeringen - og den virker lige så godt på denne måde.

I maskinkode burde man have nulstillet P i begyndelsen af programmet - I BASIC nulstilles alle variable automatisk ved programmets start. Ved udførelsen af linie 20 her første gang vil P altså få værdien 1.

21 - Vi igangsætter vor GET - instruktion, der i dette tilfælde er i forbindelse med **C$**. I BASIC betyder **$** efter en variabel, at den indeholder en alfanumerisk karakter kodet i ASCII. I maskinkode behøver man ikke skelne mellem cifre og tal, idet man altid får karakteren ind som en kode, der må nedbrydes til det rette. I ASCII har tallet 1 værdien 49.

Når vi bruger **C$** istedet for blot C, som vil behandle udelukkende cifre, skyldes det, at vi ikke ønsker afbrydelse fra BASIC-fortolkeren, hvis andet end et ciffer indlæses - og på et tidspunkt skal vi jo have en bindestreg ind!

30 - Dette en en mini-løkke, som vil vende tilbage til 21, indtil én karakter er indtastet og dermed gemt i **C$**. Der er i virkeligheden intet mellem de to sæt " (anførelsestegn) - vi har af klarhedshensyn placeret et mellemrum.

31 - Vi kommer kun hertil, hvis der er blevet indtastet en karakter, og her har vi en kursor-kontrol instruktion. Vi har benyttet to pile til at angive, at kursoren skal bevæges to linier op. I en rigtig programlistning vil et maskinafhængigt kontroltegn stå i listningen. De to pile svarer altså hver til instruktionen: KURSOR OP.

Denne instruktion vil - afhængig af den indbyggede monitor - ofte være overflødig i maskinkode, da man her normalt selv må flytte kursoren ned for hver ny linie.

```
005 DIM CPR(5)
010 GOTO 60
020 P = P + 1
021 GET C$
030 IF C$=" " THEN 21
031 PRINT "↑↑"
032 PRINT TAB(P); C$
040 C% = ASC(C$)
050 RETURN
060 PRINT
070 A% = 1
080 B% = 3
085 PRINT "INDTAST CPR–NR."
086 PRINT
090 FOR J = A% TO B%
100 FOR K = 1 TO 2
110 GOSUB 20
120 IF C% > 47 AND C% < 58
    THEN 150
130 PRINT "FEJL"
131 PRINT "↑↑"
132 P = P – 1
140 GOTO 110
150 C% = C% – 48
160 IF K = 1 THEN C% = C% * 10
170 CPR(J) = CPR(J) + C%
180 NEXT K
190 NEXT J
200 IF A% = 4 THEN 280
210 GOSUB 20
220 IF C% = 45 THEN 250
230 PRINT "FEJL: TEST –"
231 P = P – 1
232 PRINT"↑↑"
240 GOTO 210
250 A% = 4
260 B% = 5
270 GOTO 90
280 PRINT "CPR–NR. ER:"
290 PRINT CPR(1); CPR(2); CPR(3);
    "–"; CPR(4); CPR(5);
300 PRINT
310 PRINT "NYT NR. TAST –"
320 GOSUB 20
330 IF C% <> 45 THEN END
350 FOR J = 1 TO 5
360 CPR(J) = 0
365 NEXT J
366 P = 0
370 GOTO 60
```

P
20

Når vi i det hele taget flytter på kursoren, skyldes det, at vi ønsker at cifrene i CPR-nr'et kommer efter hinanden på samme linie, og i BASIC vil en PRINT-instruktion automatisk skifte linie.

32 - Nu udskriver vi den karakter, som vi netop har fået indlæst. Da hele CPR-nummeret skal stå på samme linie, må vi specificere positionen på linien, og dette bruger vi TAB(P) til. På nuværende tidspunkt er P = 1, og det første ciffer vil derfor blive udskrevet i 1. position på linien. I maskinkode vil man flytte kursor'en P–1 gange til højre.

40 - Vi kan ikke arbejde aritmetisk med bogstaver gemt i tekst-variable, og vi må i BASIC derfor ændre indholdet af C$ til alm. tal. Dette gør vi ved at flytte indholdet af C$ over til C% via en ASC-instruktion. Dette er overflødigt i maskinkode.

50 - Indlæsningsrutinen er slut, og vi går tilbage til vort udgangspunkt - denne gang instruktionen efter 110.

120 - Vi skal have sorteret evt. fejlkarakterer fra, og det gør vi med dette spørgsmål. Tallene 0 - 9 har værdien 48 - 57 i ASCII, og hvis værdien af C ligger der imellem, er det indtastede sandsynligvis korrekt, og vi springer fejlrutinen over ved at gå til 150.

150 - Her laver vi ASCII-koden om til alm. decimalværdi ved at fratrække 48. Da 1 i ASCII er = 49, vil vi ved at trække 48 fra netop få 1 - og ligeledes med de andre cifre. Dette må tilsvarende gøres i maskinkode.

160 - Nu skal vi bruge K fra den inderste løkke. Hvis K = 1, er vi i færd med første gennemløb, og det drejer sig altså om det første ciffer af 2, som kan placeres i ét ord. I maskinkode laver vi et skift til venstre på 4 positioner - i BASIC ganger vi med 10.

170 - Nu kan vi blot addere C til den celle, som skal indeholde CPR-nr'et. Vi arbejder i dette tilfælde kun med 1 nr, som vi gemmer i 5 fortløbende ord med fællesadressen CPR. Den yderste løkke med J som tæller vil give os den relative adresse indenfor rækken af CPR-ord.

Her ved første gennemløb vil vi altså få indlæst værdien af C (ganget med 10) i det første ord i CPR-rækken.

180 - Vi slutter her 1. gennemløb af K-løkken, og K tælles op til 2.

Næste gennemløb er ganske identisk med det netop udførte, indtil vi når til linie 160. Her vil værdien af K betyde, at vi ikke ganger den indlæste karakter med 10, og når den i linie 170 adderes til samme celle, som vi før havde fat i, vil det ske som en ener.

Hvis vi har indlæst to 1-taller, vil vi første gang lave 1-tallet om til 10, og anden gang vil vi addere 1 og få 11 gemt i den ønskede celle.

Nu har vi gennemløbet K-løkken de 2 gange, som vi skal, og vi går til 190.

190 - Vi har gennemløbet J-løkken én gang, og J tælles op med 1 til 2, og vi gennemløber atter rutinen 2 gange.

Da vi benytter J til placering af den indlæste information, vil vi ved andet gennemløb få gemt de næste 2 indlæste karakterer i CPR(2).

Da J denne gang ikke kan blive højere end værdien af B% = 3, vil de foranstående rutiner blive gennemløbet ialt 3 gange, og når vi er færdig i denne omgang med J-løkken, vil vi have fået indlæst 6 cifre som 3 stk. 2-cifrede tal, der er gemt i de første 3 celler af CPR-rækken.

Hvis der under denne del af indlæsningen var indtastet andet end et ciffer, ville vi i linie 120 være blevet dirigeret til linie 130.

130 - PRINT-ordren vil flytte kursoren én linie ned, og dér vil der blive udskrevet ordet FEJL.

131 - Endnu en gang må vi flytte vores kursor op - det er symboliseret med de to pile.

132 - Vi må tælle P ned, da det næste ciffer, som indtastes, jo er en erstatning for den karakter, som fejlagtigt kom ind istedet - og det er P, som bestemmer, hvor på linien, næste karakter udskrives.

140 - Vi går til indlæsning af ny karakter via linie 110, da vi ønsker at komme til 120, når subrutinen i 20 er udført.

På ét eller andet tidspunkt må det formodes, at det er lykkedes at få indtastet

6 cifre i rækkefølge, og vi kan da fortsætte, hvor vi slap.
200 - A% vil ved første gennemløb være lig med 1, så vi fortsætter til næste linie.
210 - Vi kalder indlæsningsrutinen, som vil hente en karakter til os i C%.
220 - Nu er vi nået til det punkt i CPR-nr'et, hvor vi skal indlæse bindestregen, og vi sikrer os derfor, at ASCII-værdien af C% = 45 = en bindestreg. Hvis dette er tilfældet, fortsætter vi i 250.
250 - Det var en bindestreg, og den benytter vi ikke til noget som helst, da vi ved, at den altid skal være der.
Derfor gør vi klar til indlæsning af de næste 4 cifre ved at give nye værdier til A% og . . .
260 - B%.
270 - Og så op til J- og K-løkken igen, hvor de næste 4 cifre indlæses til de 2 sidste ord i CPR-rækken.
230 - Hvis der ikke blev indtastet en bindestreg, ville spørgsmålet i 220 have bragt os hertil, hvor vi udskriver på linen under CPR-nr'et på skærmen, at det var en fejl med den manglende bindestreg.
231 - Også denne gang må vi tælle P ned, så den rigtige bindestreg kommer på den plads, hvor den forkerte karakter nu står.
232 - Ligeledes må vi give kursoren besked om at gå et par linier tilbage, så den næste karakter kommer på den rigtige linie.
240 - vi skal atter via en anden linie - her nr. 210 - til indlæsningsrutinen.
Når vi efter indlæsning af alle 10 cifre har fået disse godt af vejen, vil vi ved udløbet af K- og J-løkken 2. gang nå frem til linie 200. Og denne gang vil A% jo netop være = 4, hvorfor vi går ned til linie 280.
280 - Dette har egentlig ikke noget med indlæsningen at gøre, men er en kontrol af, at det indtastede nr. er blevet rigtigt indlæst og gemt. Vi skriver først en kort forklarende linie ud.
290 - Udskriften af CPR-nr'et foregår rent slavisk med de første 3 celler af CPR-rækken, en bindestreg, og endelig de 2 sidste celler. Hvis vi havde lavet en løkke til dette, ville hver print-instruktion have givet et linieskift, og vi skulle have einkluderet en kursor-retur.
300 - Linieskift.
310 - Skal vi én gang til?
320 - også her benytter vi vor indlæsningsrutine i 20.
330 - Vi bad om en bindestreg, hvis vi skulle fortsætte, og hvis der er kommet andet ind, slutter programmet med END, og vi returnerer til BASIC-fortolkeren.
350 - Hvis vi fik en bindestreg ind, laver vi en lille ny J-løkke, som i de næste par linier nulstiller CPR-rækken.
Hvis vi havde ønsket at læse en hel række CPR-nr. ind og gemme dem alle, ville vi her have øget adressen til et nyt sted i CPR-rækken.
Vi kunne eksempelvis i første instruktion i programmet have defineret DIM CPR som (100,5), og en ekstra variabel havde derefter styret den første del af adressen i CPR-rækken - og denne variabel var så blevet talt op på dette sted.
366 - Da programmet startede, var P = 0. Det er den ikke nu, så før næste tal læses ind, må vi nulstille P.
370 - Op til nyt nr.

KOMMENTARER

Hvis programstumpen havde været inkluderet i et „professionelt" program, hvor det var vigtigt med færrest mulige fejl, ville yderligere en række kontrolfunktioner have været indbygget.
CPR-nr'et er som bekendt opbygget over fødselsdage, og det kan derfor med sikkerhed forudsiges, at hvis de 2 første cifre tilsammen er større end 31, er der en fejl. Ligeledes kan de næste 2 ikke være højere end 12.
Tilsvarende vil man have mulighed for at kontrollere, at de sidste cifre ikke er større end det kendte maksimum, som vistnok er 2999. Evt. kunne man kontrollere køn versus lige/ulige.
Det er vigtigt ved al programmering, at man forudser samtlige teoretiske fejlmuligheder fra operatørens side og laver passende kontrol- og korrektionsforanstaltninger - ellers vil man let komme i vanskeligheder.

Telmac 1800

I forrige nr. af HFD kunne vi fortælle om en ny prisbillig hobby-datamat: TELMAC 1800. Her følger en nærmere beskrivelse. Der blev ikke plads til et foto, men der er et billede af TELMAC'en på side K20.

TELMAC 1800 er en micro-datamat på et enkelt print. Den minder derfor en del om KIM-1, som vi tidligere har beskrevet, men den adskiller sig alligevel på væsentlige punkter fra denne.
For det første leveres TELMAC'en som byggesæt, og det er sandsynligvis én af forklaringerne på, at den er en smule billigere.
Dernæst er der ikke det konventionelle hexadecimal tastatur, men et stort tastatur med berøringsfelter, som giver mulighed for indtastning af langt flere forskellige karakterer.
Endelig er der intet display på selve printet, men en TV-generator, som via en TV-modulator gør det muligt at se resultaterne direkte på sit eget TV.

SÆRPRÆGET DATAMAT

Datamatens CPU er en RCA CDP 1802 i 1800-serien, som vi skal bringe arkitektonisk beskrivelse af snarest.
Det specielle ved denne CPU - at den ikke har nogen nedre begrænsning i clockfrekvens - kunne man godt have benyttet til variabel clock-generator, så man under indkørsel af programmer havde mulighed for at følge disse i „slow-motion."
Iøvrigt er CDP 1802 på mange måder en spændende kreds, som desværre er blevet overset af de fleste - men altså ikke af TELMAC-folkene.
Det er ligeledes en ny kreds fra RCA, en CDP 1861CD, som fungerer som video display controller.
Når byggesættet leveres, inkluderes 2K bruger RAM, og der er på printet sokler og adressedekodning til yderligere 2K.
Der er yderligere på printet en ROM, som indeholder en simpel monitor til styring af tastatur og video-display.
Iøvrigt bærer det lille og relativt enkelt opbyggede print præg af, at de nyere kredse indeholder langt flere funktioner, da der er forbløffende få kredse og separate komponenter involveret.
TELMAC 1800 skal blot tilsluttes + 5 V, så strømforsyningen - som ikke er inkluderet - skulle ikke volde vanskeligheder.

SAMLING AF BYGGESÆTTET

Dette er sandsynligvis datamatens svageste punkt. I den medfølgende dokumentation fandt vi intet sted forklaring på modstandenes farvekoder, og i instruktionen oplyses der blot om værdi.
Yderligere forbeholder man sig ret til at supplere andre værdier, når disse ikke er kritiske.
Dette sidste var faktisk tilfældet for 2 modstandes vedkommende, og man føler sig ikke tryg ved selv at skulle vurdere, om den ene eller anden erstatning er tilstrækkelig ukritisk.
Byggevejledningen var endvidere vanskeliggjort ved, at der på ét ark er listet komponentværdier, på et andet ark får man oplyst monteringsrækkefølgen, og på et tredie ark ser man monteringsplanen. Og hvis man ikke har farvekoderne i hovedet, skal man altså yderligere konsultere et fjerde ark.
Monteringsanvisningen var dog i sig selv ganske godt beskrivende, og hvis man tager den fornødne tid, skulle man ikke

løbe ind i problemer.
Printet selv er velforarbejdet, men uden komponenttryk, så der er ind imellem en smule tællearbejde.
Da printet er temmelig småt, er alle komponenter tæt placeret, og det nødvendiggør brugen af en meget fin loddekolbe, da man ellers meget let kan komme til at lave utilsigtede forbindelser.

TELMAC 1800 I BRUG

Ifølge svenskerne var der blandt de første 100 stk. TELMAC kun 6, som ikke virkede efter monteringen, og alle 6 skyldtes dårlige lodninger, så rent statistisk skulle man have en god chance for, at ens TELMAC virker, første gang, man slutter den til.
Det gjorde vores prøveeksemplar også, men vi havde desværre ikke tid til de mange og lange eksperimenter. Rent praktisk skete der nemlig det, at da vi fik ét af de første eksemplarer udleveret, var programkassetten endnu ikke produceret i større antal, og vi fik i hast en interimistisk kopi fremstillet - og den kunne ikke læses ind - signalet var forvrænget på båndet. Dette vil naturligvis ikke ske med de programkassetter, som leveres til kunderne.
Videodisplayet er specielt på den måde, at man nederst på skærmen får udlæst adresse og indhold af den ønskede celle i hexadecimal format, mens resten af skærmen er en repræsentation af en del af laget - bit for bit i hvide eller sorte felter.
Ved at ændre indholdet af lageret er det derfor muligt at frembringe grafiske effekter uden større besvær, og da TELMAC 1800 programmeres i maskinkode eller CHIP 8 - et meget forenklet miniassemblersprog med et par specielle instruktioner - er maskinen meget hurtig og kan let bruges til bevægelige grafiske figurer. I så fald vil man dog nok ønske at udbygge med de ekstra 2K, som er direkte tilgængeligt på printet.

YDRE TILSLUTNINGER

Man kan tilslutte en højttaler og få maskinen til at knase hyggeligt eller spille ens yndligsmelodier. Ligeledes er der indbygget kassette-interface, så man kan gemme og indlæse programmer og data.
Der er mulighed for udbygning op til et fuldt 64K lager, ider ROM'en dog optager en smule af pladsen - men der er næppe mange, som drømmer om at lave maskinprogrammer på op til 64K.
Tastaturet drillede en smule, idet det er ret brum-følsomt, og det viste sig, at den bedste kontakt-behandling foregår med elektrisk ledende skum af den type, som de medfølgende MOS-kredse er monteret i.
Udskift viskelæderet på en blyant med sådan et stykke skum, og tastaturet er let at betjene. Kritik af tastaturet iøvrigt må tage hensyn til den samlede pris.

KONKLUSION

Vi fandt TELMAC 1800 lidt vanskelig at bygge, men sjov og underholdende i sine muligheder.
Den er lille og kompakt og virker prisrimelig, når tilslutningsmuligheder og det medfølgende tastatur tages med i betragtning.
Det er nødvendigt med en TV-modulator, hvis man ønsker tilslutning direkte til TV's antenneindgang, og en sådan modulator er dels beskrevet her i bladet, og dels kan den anskaffes for ca. kr. 75,—.
Den medfølgende programkassette indeholder en lang række underholdende programmer, og instruktionssættet er let at lære og benytte.
TELMAC 1800 må derfor hilses velkommen på markedet som et alternativ for de hobby-amatører, som gerne vil have funktioner via tastatur og TV-skærm, og som gerne vil holde sig et stykke under kr. 2.000,—. PH

TELMAC 1800 distribueres af Fa. Piezodan og kan fås incl. programkassette og TV-modulator for under kr. 2.000,—. Yderligere info på tlf. (03) 28 37 44.

Logiske operationer

Heldigvis behøver vi sjældent at lave komplicerede beregninger i binær form. Hvis der arbejdes med maskinkode, vil det som regel dreje sig om processor-orienterede programmer, hvor det gælder om at styre ydre enheder på det rigtige tidspunkt og efter de rigtige parametre.
Det kan f.eks. dreje sig om en vaskemaskine, hvor skylning, centrifugering etc. påbegyndes udfra et sæt data bestående af tid, allerede udførte funktioner, temperatur, programvælger m.m.
Når denne form for opgaveløsning skal programmeres, benyttes hyppigst maskinkode, som så i masseproduktion indlægges i ROM-kredse, så forbrugeren aldrig bemærker, at der er en datamat indbygget i vaskemaskinen.
Ved al processtyring er det hyppigere logiske beslutninger end matematiske funktioner, som er afgørende for programmets forløb.
Ganske som vi i matematikken benytter et sæt tegn til at tilkendegive oprationen mellem cifrene, f.eks. + ved addition, anvender vi ved logiske operationer en række symboler.

"OR" FUNKTIONEN

"OR" betyder "eller" og indgår i en logisk operation som f.eks:

IF A OR B THEN C

Oversat til dansk betyder dette:

Hvis A ELLER B så C

Internt vil datamaten, når et udtryk som dette mødes, beregne værdien af hhv. A og B. Værdien 0 opfattes som FALSE (usand), mens værdien 1 opfattes som TRUE (sand).
Hvis vi i en vaskemaskine ønsker at tænde en blinkende lampe, hvis vandet løber over, eller temperaturen bliver for høj, kan vi udtrykke dette således:

Hvis Vandstand ORTemparatur,
så Alarm.

Det logiske tegn for en OR-funktion, er et alm. plus-tegn (+), som ikke må forveksles med det aritmetiske plus.
Hvis vi i vor datamat har følere for vandstand og temperatur tilkoblet, vil vi med et OR-udtryk kunne få advarselslampen til at lyse.
OR-udtrykket vil nemlig resultere i et sandt svar, hvis enten den ene, eller den anden, eller dem begge er sande. Kun hvis ingen af udtrykkene er sande, vil resultatet også være usandt - og lampen forbliver slukket.

"AND" FUNKTIONEN

Almindeligvis benyttes $\wedge$ til at angive en AND-funktion. "AND" betyder "og", og funktionen kunne passende kaldes for "både og".
Det kræves nemlig, at begge de to funktioner er opfyldt (sande), for at løsningen er sand.
Hvis vi i vort ovenstående eksempel havde benyttet AND istedet for OR, ville vor advarselslampe kun have tændt, hvis både vandstand og temperatur var galt afmarcheret.
En AND funktion er derfor ikke særlig hensigtsmæssig i forbindelse med alarm, og en mere typisk anvendelse vil være igangsætning af Centrifugen, hvis både Programvælger og Tid er sande:

Hvis Programvælger AND Tid, så
Centrifugering.

"XOR" FUNKTIONEN

XOR er en forkortelse for EXCLUSIVE OR og kan bedst fordanskes med "enten

eller". Og det ganske bogstaveligt, så udtrykket kun er sandt, hvis én af faktorerne er sande - altså ikke, hvis de begge er sande som i AND (både og).
Det er lidt forskelligt, hvorledes XOR betegnes, men det korrekte er ⊻
XOR anvendes ikke nær så hyppigt, som de to forangående, men hvis vi forestiller os, at sikringen i vaskemaskinen af sikkerhedshensyn er beregnet, så den går, hvis både motoren centrifugerer, og varmelegemet er igang samtidigt, og at man har tilrettelagt vaskeprogrammet, så dette ikke vil kunne ske under normale forhold, kan man igen forestille sig dette udtrykt logisk således:

Hvis Centrifugering XOR varme,
så Strøm.

Et sådant udtryk vil have til følge, at så længe kun én af de omtalte funktioner er igang, vil der være strøm til rådighed. Hvis de afbrydes, eller hvis de begge starter samtidig, vil strømmen stoppe.
I praktisk brug vil dette udtryk naturligvis ikke kunne stå alene, men ideen med XOR skulle gerne være belyst hermed.

"NOT" FUNKTIONEN

"NOT" betyder "ikke", og udtrykket ændrer værdien af den efterfølgende variabel ganske som man kan ændre fortegn r almindelige variable.
H i er sand, som vil NOT A være usand.

NOT er den eneste logiske opration, som kan arbejde sammen med en enkelt variabel - de 3 forangående skal alle stå mellem 2 logiske variable.
NOT er særdeles anvendelig, idet man med de logiske operationer ofte har behov for det modsatte resultat.
Vi kan forestille os, at vandstandsmåleren har værdien 0, så længe vandet ikke når op til max. højde, så vil en føler give måleren værdien 1.
Når vaskemaskinen skal tage vand ind, sker det i henhold til programvælgerens stilling, og der kan indtages vand, indtil vandstanden er den ønskede.
Dette vil kunne styres med en sammensat logisk funktion, hvor vi sikrer os, at programvælger og vandstandsmåler giver tilladelse til opfyldning:

Hvis Programvælger AND NOT
Vandstand, så Fyld.

"NOR" FUNKTIONEN

De hidtidige funktioner har kunnet benævnes som "eller", "både og", "enten eller" og "ikke". I denne sammenhæng savner man ligesom "hverken eller", og det logiske udtryk herfor er NOR.
Dette er mest af akademisk interesse, og der er ikke mange datamater, som har NOR med i deres logiske operationer.
Hvis man skulle komme ud for at ville udtrykke NOR, kan dette gøres ved at benytte NOT foran AND funktionen - det vil give samme resultat.

SANDHEDSTABEL FOR LOGISKE OPERATORER

IF A	B	THEN C = AND	THEN C = OR	THEN C = XOR
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	FALSE

True = sand = 1. False = usand = 0.

Høj-niveau sprog: BASIC

Da flere og flere datamater forsynes med fortolkere af høj-niveau sprog, må det være på tide, at vi fortæller lidt om disse. Meget naturligt begynder vi med BASIC.
Og næsten lige så naturligt er det, at at vi har fået lektor Børge R. Christensen i Tønder til at tage sig af dette – B.R.Christensen har nemlig udviklet en avanceret BASIC – STRUCTURED BASIC – som han for tiden programmerer en fortolker til. Samme fortolker er iøvrigt beregnet til Z-80.
Så der bliver noget for selvbyggerne at glæde sig til.

Artikel: Lektor Børge R. Christensen

INDLEDNING

Programmeringssproget BASIC (**B**egin**e**rs **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) blev beskrevet i 1964 af J. Kemeny og T. Kurtz ved Dartmouth College i USA. Sproget blev hurtigt meget populært og er i dag uhyre udbredt. Det vil formodentligt være svært at finde en datamat i 1978, som ikke er udstyret med en eller anden BASIC-udgave.
Sproget er let at lære - det vil læseren erfare i denne og de følgende artikler - og egner sig udmærket til løsning af mindre programmeringsopgaver.
Ved programmering af mere omfattende og komplekse processor, hvor der kan indgå mange variable størrelser, og hvor der skal udføres mange indbyrdes afhængige delprocessor, er sproget imidlertid for simpelt, og der er da også i tiden siden 1964 blevet udviklet en mængde forskellige udvidelser af BASIC, hvilket har resulteret i, at der nu findes flere hundrede versioner af sproget.

Ved nærværende gennemgang vil vi først rette opmærksomheden mod grundversionen af BASIC og derefter gøre rede for nogle af de vigtigste udvidelser af den. Den hidtil stærkeste udgave af BASIC, den såkaldte RC BASIC, er forøvrigt udviklet for en dansk datamat, nemlig A/S Regnecentralens RC 3600 (RC 7000).
Ved udførelsen af BASIC-programmer kan man vælge mellem to principielt forskellige måder: Man kan enten lade en BASIC–**oversætter** (compiler) transformere brugerens program til maskinkode og derpå lade datamaten udføre denne, eller man kan lade et stort program, som til stadighed ligger i maskinens lager, **fortolke** BASIC-programmet uden at ændre på det. De fleste systemer anvender det sidstnævnte princip, idet dette i almindelighed gør det lettere for brugeren at skrive og rette sine programmer. I denne fremstilling vil jeg gå ud fra, at der anvendes en BASIC-**fortolker**, men det skal dog understreges, at dette ingen indflydelse har på sproget som sådant.

1. TAL OG VARIABLE.

1. Tal

De fleste BASIC-udgaver tillader mindst 6 betydende cifre i et tal. Der skelnes som regel ikke mellem hele tal og decimalbrøker, idet disse behandles på samme måde af fortolkeren. Det har således ikke nogen indflydelse på regnehastigheden eller nøjagtigheden, om man regner med den ene eller den anden slags tal. Her er nogle eksempler på tal, skrevet i BASIC-notation:

234 234567 3.14159 4.5 .78

Læg mærk til, at skilletegnet mellem heldel og brøker i den decimalbrøk altid er et **punktum**, og at man ikke behøver sætte et nul foran punktummet i en ægte brøk.
De ovenfor anførte tal er alle **positive**. Hvis et tal er **negativt**, skal der sættes et **minustegn** foran det:

−567 −3.44 −89.4567 −.78

Til angivelse af meget store og meget små tal kan man bruge den såkaldte **eksponentialnotation**, idet bogstavet E kan bruges i betydningen "potens af 10". Vi har fx.:

38E5 = 38 gange 10^5 =
38 gange 100000 = 3800000

38E−5 = 38 gange 10^{-5} =
38 gange .00001 = .00038

.4E2 = .4 gange 10^2 =
.4 gange 100 = 40

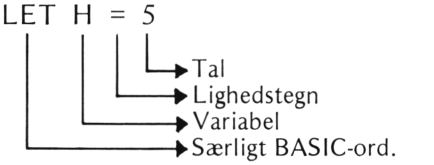
Bemærk, at der altid skal skrives en talværdi foran E'et, når det angiver potens af 10. Hvis man fx. vil angive tallet 10^3 (1000), kan man skrive: 1E3, hvorimod man ikke må skrive E3. Dette sidste bliver forstået af BASIC-fortolkeren på en helt anden måde (se nedenfor).

2. Variable. Værditildelinger.

Variable bruges til at betegne **tal** eller **tekster**. Vi skal først se på variable, der kan bruges til at angive tal. I de fleste BASIC- udgaver betegnes en variabel ved enten : (1) et **enkelt bogstav** eller (2) et **enkelt bogstav** og et **enkelt ciffer**. Som eksempel kan vi tage:

B G H C1 D7 Q8

Som nævnt kan variable knyttes sammen med talværdier. Lad os fx. se på følgende BASIC-sætning:



Når denne sætning **udføres**, bliver tallet 5 gemt i datamatens arbejdslager under navnet H. Man kan forstille sig, at tallet 5 "flytter ind" i lageret og samtidig får tildelt "adressen" H:

H: 5

Der kan naturligvis aldrig "bo" mere en et tal ad gangen på "adressen" H. Hvis et nyt tal skal "flytte ind", må det gamle tal "rykke ud". Man siger også, at den gamle talværdi bliver **overskrevet** af den nye. Hvis fx. denne sætning bliver udført:

LET H=24.6

bliver indholdet af H udskiftet med tallet 24.6:

H: 24.6

Det er vigtigt, at læseren er klar over, at variable i virkeligheden er adresser på områder i lageret, hvor de tilsvarende talværdier er gemt (lageret). Vi burde sådan set sige, at fx. tallet 24.6 "er lager i H", men det er almindeligt, at man simpelthen siger "H **er lig med** 24.6".
Denne sprogbrug er naturligves hentet fra matematikken, hvorfra vi også har

udtrykket, at "H **bliver sat lig med** 24.6". Indenfor datalæren er det dog også almindeligt at sige, at "H **får tildelt** værdien 24.6".

3. Udtryk.

Fra matematik og regning kender vi til **udtryk** (eller **formler**) som fx.:

$\frac{1}{2}h(a + b)$ $2a^3 + 3b$ $3\sin(x) - \cos(x)$.

For at kunne skrive sådanne udtryk har vi brug for bl. a. nogle tegn til at udtrykke de forskellige **regningsarter**. I BASIC råder man over følgende fem **regnetegn (aritmetiske operatorer)**:

+ og **—** for hhv. **addition** og **subtraktion,**

***** og **/** for hhv. **multiplikation** og **division**, samt

↑ for **potensopløftning.**

Her er nogle eksempler på udtryk, skrevet i BASIC-notation:

(A+B)*H/2 2*A*3+3↑B

(A+B)/(A—B)

Bemærk, at gangetegnet * aldrig må underforstås, således som det er almindeligt i sædvanlig matematisk skrivemåde. Eventuelle paranteser har helt samme betydning som i almindelig matematik og regning.
Lad os tænke os, at de variable H og G har fået tildelt værdierne hhv. 10 og 15, og at der udføres følgende sætning:

```
LET A=H G/2
```

Denne sætning skal forstås således: "Sæt A lig med det resultat, der fremkommer, når udtrykket på højre side af lighedstegnet udregnes". Resultatet af udregningen bliver naturligvis 10.15/2 = 75, og denne værdi bliver altså den variable A.

Særlig interessante er tildelinger som fx. denne:

```
LET T=T+1
```

hvor den samme variabel forekommer både på venstre og højre side af lighedstegnet. For at forstå denne sætning, er det nødvendigt at huske, at den bliver læst således af BASIC-fortolkeren: "Tag det tal, der er tildelt T. Læg 1 til det, og lad det herved fremkomne tal være ny værdi for T". Lad os tænke os, at T har værdien 8, **før** vi udfører sætningen. T vil da have værdien 9 (nemlig 8+1), **efter** at sætningen er udført.
Bemærk, at tegnet / er et divisionstegn og **ikke en brøkstreg**. Således betyder fx.:

(A+B)/A—B

at værdien af A skal adderes til værdien af B, hvorpå man skal dividere med A og til slut subtrahere B, mens derimod:

(A+B)/(A—B)

betyder det samme som brøkstregen:

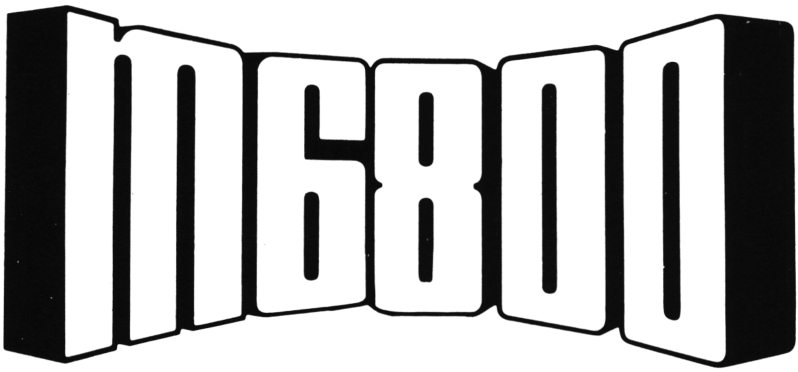
$$\frac{a + b}{a - b}.$$

I sidste tilfælde kan man altså **ikke undvære parantesen** om A—B, når der skrives i BASIC-notation. BRC

Eksempel på en stump af et BASIC-program fotograferet fra skærmen af en PET 2001. De forskellige instruktioner forklares i denne og kommende afsnit af artiklen.

```
 310 PRINTT/100;"SEKUND":PRINT
 320 K=K+1
 330 IFK<10THEN50:PRINT
 340 PRINT"DIT GENNEMSNIT FOR 10 FORSØG"

 350 PRINT"BLEV ";P/1000;" SEKUND"
 351 IFP/1000>3THENPRINT"DET ER VIST IKK
E DIN DAG!"
 352 IFINT(P/1000)=2THENPRINT"DU VAR VIS
T UHELDIG - PRØV IGEN"
 353 IFINT(P/1000)=1THENPRINT"SLET IKKE
GALT AF EN BEGYNDER!"
 354 PRINT
 355 IFTIME>54000THENPRINT"ER I KLAR OVE
R, AT VI HAR ARBEJDET I ET HELT KVARTER?
"
 356 IFTIME>54000THENTIME$="000000"
 357 IFP/1000<1THENPRINT"BRAVO - GODT KL
ARET!"
 358 IFP/1000<.9THENPRINT"EXCELLENT!!!"
 359 IFP/1000<.8THENPRINT"KAN IKKE GØRES
BEDRE!!!!!"
 360 GOTO45
READY.
```



TOTAL DEVELOPMENT SYSTEM

KOMPLET KEYBOARD

5 ELLER 9 TOMMER SKÆRM (EL. EGET TV)

BÅND INTERFACE

ASSEMBLER OG EDITOR PÅ PROM

BASIC TOLKER PÅ PROM

8K ELLER 16K LAGER (ELLER MERE)

PROM PROGRAMMERER

POWER SUPPLY OG TDS KABINET

BILLIG LINJE PRINTER 30 TEGN / SEK

ALT ER SAMLET OG AFPRØVET

START MED KR. 6509,00 – BYG 6800 MODULER PÅ

(03) 38 57 16

gds-henckel aps GDS

a franchised **MOTOROLA** *Semiconductor* distributor

PET 2001

I forrige nummer af HFD fortalte vi, at PET 2001 nu kom til det danske marked. Denne gang har vi haft fat i et eksemplar fra den første sending, og vi iler med at bringe en nærmere beskrivelse. Foto af PET 2001 er på side M33, og vi bringer på side B37 et program skrevet til denne datamat. Illustrationerne i programmet Reaktionstid er fotograferet fra en PET 2001.

PET 2001 er en positiv overraskelse.
Ikke sådan at forstå, at der intet negativt kan siges om den, men fordi kombinationen af pris og anvendelse i sammenligning med det hidtidige udbud på det danske marked så absolut sætter denne datamat i særklasse.

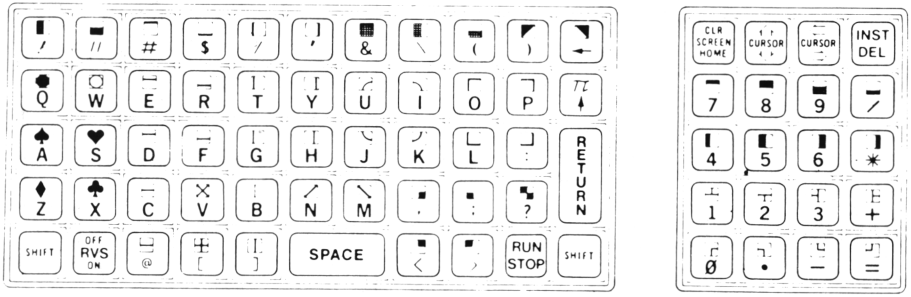
MINUSSERNE

Inden vi går igang med de mange positive sider, vil vi lige opremse de negative - så er det overstået.
PET 2001 er nok den billigste hobbydatamat på markedet til dato, men den samlede pris på ca. kr. 10.000,– er stadig for høj i sammenligning med, hvad en sådan datamat rent faktisk kan produceres for idag.
En stor del af prisstigningen fra den amerikanske pris skyldes luftfragt, told, statsafgift og moms, mens importøren yderligere må sikre sig et vist provenu til mulig service. Alligevel synes man, at prisen burde kunne sænkes en smule.
Den nuværende pris vil nemlig for mange betyde, at PET 2001 placeres udenfor hobby-området, og som vi senere skal se, har den næppe kapacitet nok til forretningsformål.
Tastaturet er for småt til at tillade hurtig indtastning, hvilket for en rutineret typist er særdeles irriterende. Dette er selvfølgelig et prisspørgsmål, men da man kan få glimrende tastaturer i fuld størrelse til under kr. 500,–, kan forskellen ikke have været afskrækkende.
Commodore siger selv, at de vil fremkomme med en større PET med større tastatur - og til en højere pris.
Kassettebåndoptageren har ikke båndtæller. Når flere programmer eller data er placeret på samme bånd, kan PET godt nok finde frem til disse, men hvis man hver gang skal begynde at søge forfra, tager det en frygtelig tid. Løsningen vil for de fleste nok blive et større bibliotek, så man hver gang kan begynde forrest på en kassette.
TV-skærmen er lige lille nok, men til de fleste formål vil den være tilstrækkelig.
Lageret på 8K er i virkeligheden kun 7.167 byte a 8 bit, hvilket er for småt til mere alvorlig programmering. Hvis maskinen hovedsagelig skal bruges til spil og personlige programmer, vil 8K være tilstrækkelig, men man kan ikke lave meget forretningskørsel med så begrænset hukommelse.
Tilslutningerne falder udenfor de hidtil gængse normer, og der er således ikke umidelbar mulighed for tilslutning af større hukommelse. Det kræver godt kendskab til hardware selv at udbygge PET 2001 til en mere alvorlig datamat.

PLUSSERNE

Det kunne lyde som en række ret alvorlige indvendinger, men der er faktisk så mange plusser, at vi stadig vil hævde, at PET 2001 er i særklasse på markedet.

Således er den indbyggede BASIC den hurtigste, vi endnu har set i en datamat i denne størrelse.
Vi lavede en lille sammenligning med en Motorola TDS, hvis BASIC fylder nogenlunde det samme. Begge maskiner blev sat til at kalde en subrutine 1.000 gange. Subrutinen adderede 1 til en variabel.
TDS-datamaten var 30,5 sekund om de 1.000 subrutine-kald. PET 2001 klarede det samme på nøjagtig 6 sekunder!
Samme BASIC, som er baseret på en fortolker fra MITS, indeholder en lang række instruktioner, som gør det muligt at lave virkelig effektiv programmering.
Tastaturet udnytter sit specielle format til at bringe en lang række grafiske tegn indenfor rækkevidde - streger, kryds, kortsignaturer etc. etc. Der kan laves søjler, diagrammer og almindelige tegninger, ja - maskinen er så hurtig, at det må være mulig at lave en simpel tegnefilm (En hånd, der vinker o.lign).
Tastaturet giver fuld kontrol med placering af udskrift i positiv eller negativ tekst eller grafisk symbol. Der er faktisk ingen grænser for mulighederne i udskrift på skærmen.
Commodore vil senere bringe en række ydre enheder på markedet, og det forlyder, at en printer og en floppy-disc vil dukke op i løbet af året til hver ca. kr. 6.000,—. Det vil være en velkommen udvidelse. Ligeledes vil yderligere RAM blive tilbudt.
(Som følge af in/output strukturen kan der højst tilsluttes 32K bruger-RAM).
PET 2001 er færdig til brug, når den kommer ind ad døren - sæt den i stikkontakter, tænd — og du er igang!
Den indbyggede båndoptager kan benyttes til lagring af både programmer og data, og der er en lang række smarte instruktioner til håndtering af dette. Blot en skam, at manualen er lidt „tung" på dette område.
PET 2001 har indbygget 2 elektroniske ure, som kan bruges i programmerne til mange formål. Det ene ur udlæser alm. tid i time, minut, sekund, mens det andet ur er en 60 Hz tæller. Man kan f.eks. benytte urene til at afprøve hastigheden på forskellige programløsninger etc.
Maskinen er kompakt og handy, og selvom dens udseende ikke er det meget smarte strømlinede, som oprindelig blev stillet i udsigt, er den afgjort meget nydelig med glimrende finish overalt.
Den er tilstrækkelig robust i opbygning til, at man uden større uro kan flytte den i f.eks. en bil.
Da skærmen er relativ lille, har man fornuftigt valgt at udlæse 24 linier a 40 karakterer, så disse bliver rimeligt store og letlæselige. Der benyttes en 8 x 8 punkt matrix til symboler og 7 x 7 til tal og karakterer, så der er glimrende opløsning.
Som følge af den effektive BASIC vil et program i PET'en ofte fylde mindre end i maskiner med en mere simpel BASIC.
Der skelnes således mellem integer og flydende punkt variable, hvorved man



Tastaturet til PET 2001. Som det ses, er tastaturet 2-delt med tal og aritmetiske funktioner i en særskilt afdeling til højre. I brug finder man det særdeles praktisk, at der er tilgang til alle bogstaver og tegn i lower-case, mens de grafiske symboler befinder sig i upper-case.

ofte kan nøjes med 2 byte til hver variabel.
Tekst-variable fylder ikke mere, end nødvendigt for det indtastede - mange andre programmer sætter fast plads af til tekst-variable.
Der kan tilsluttes en ekstra båndoptager, så der kan laves temmelig effektiv behandling af data udenfor maskinen - når dette styres korrekt, vil man opdage, at der kan behandles endog ret store mængder data på blot 8K (Der er dog næppe mange amatører, som kan overse endsige programmere et effektivt databehandlingsprogram).
Der findes en række konnektorer for tilslutning af ydre enheder, og selvom det for de fleste vil være en vanskelig opgave at fremstille de nødvendige mellemkoblinger (interface), er det faktisk muligt at tilslutte praktisk talt alt eksisterende elektronisk udstyr. Det vil f.eks. være muligt at frakoble den indbyggede kassettebåndoptager og istedet tilkoble 2 båndoptagere med elektrisk fjernbetjening, så frem- og tilbagespoling kan styres fra datamaten!
Mere realistisk for de fleste vil nok være brugen af floppy-disc, og hvis prisen på kr. 6.000,– kan holde, vil det unægtelig være lidt af et plaster på såret.

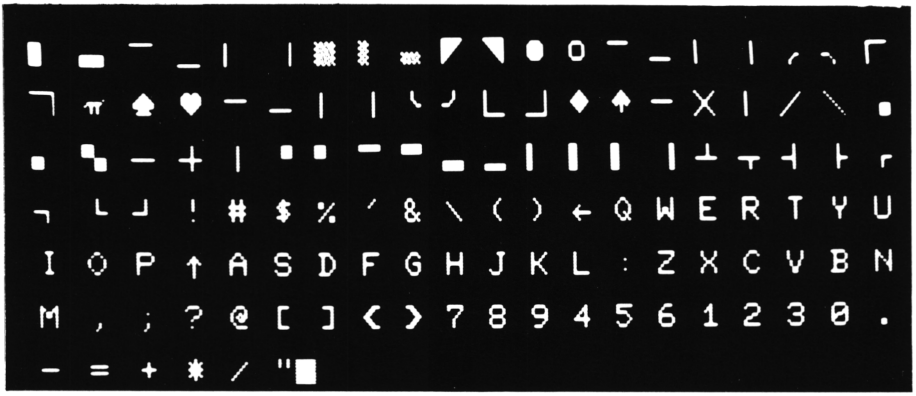
KONKLUSION

Lad det være sagt med det samme: Jeg kan godt lide PET 2001.
Den har selvfølgelig en række minusser, men kombinationen af dens pris i forhold til de nuværende konkurrenter og dens virkelig gode og effektive BASIC giver et produkt, som er afgjort attraktivt.
Hvis jeg havde haft kr. 10.000,– til overs, ville jeg straks have stillet mig op i køen, for de få timer, jeg havde sammen med PET 2001, var nogle af de mest underholdende til dato.
De, der kan afse det nødvendige beløb, kan se frem til et positivt bekendtskab.
Hvem der blot havde eget firma, så man kunne placere udgiften lidt mere fornuftigt!

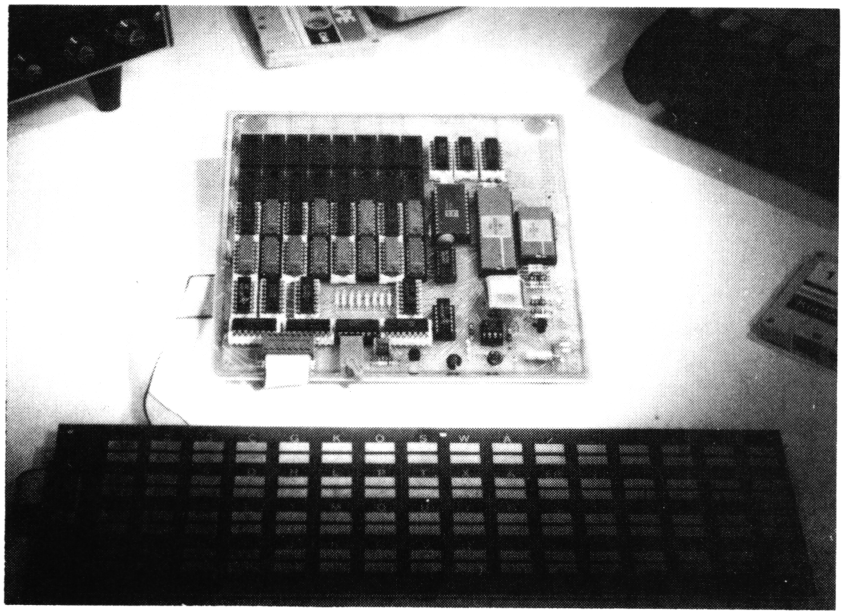
PH

```
*** COMMODORE BASIC ***
 7167 BYTES FREE
READY.
LOAD"REAKTIONSTID"
PRESS PLAY ON TAPE #1
OK
SEARCHING FOR REAKTIONSTID
FOUND REAKTIONSTID
LOADING
READY.
```

Illustrationen øverst til højre viser dels startinformationen, når PET 2001 tændes, og dels resultatet, når den beordres til at finde et bestemt program på den indbyggede kassettebåndoptager – i dette tilfælde REAKTIONSTID.
Herunder er samtlige tegn, som PET 2001 kan vise på sin skærm - mulighederne er dog i virkeligheden fordoblede, idet alle tegn også kan vises i negativ.



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.**

INTRODUKTIONSPRIS

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

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

TV-modulator til kanal 36 (UHF) kr. 75,–

piezodan aps.

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

Z-80 SIO-kort

Her følger beskrivelsen på det sidste kort til Z-80 selvbyggerdatamaten i denne omgang. På et senere tidspunkt vil vi bringe omtale af de mere specielle kort, som bliver fremstillet.

STYKLISTE

1 stk. IC3	1489
1 stk. IC4	1488
1 stk. IC6	8251
1 stk. IC8	4702
1 stk. IC10	74LS139
1 stk. IC11	25LS2521
2 stk. IC12, 14	74LS245
1 stk. IC13	74LS240
1 stk. R1	1Mohm, 1/4 W
1 stk. R2	6-polet SIL-modstandsværk, 2,2 kohm
3 stk. C1, 2, 3	390 pF
2 stk. C4, 5	56 pF
6 stk. CX	0,1 µF
1 stk. S1	6-polet DIL-switch
1 stk. S3	4-polet DIL-switch
1 stk. krystal	2,4576 MHz
1 stk. DC-DC conv.	Reliability V5 R 12-12
1 stk. printkort	78043

SIO–KORTET

Vi har i de forløbne artikler talt meget om, hvordan de forskellige dele af datamaten "taler" med hinanden. Denne gang skal vi se nærmere på, hvorledes *vi* kan "tale" med datamaten.

Det giver os samtidig anledning til at se lidt nærmere på det problem, der nok i øjeblikket er datamatamatørens største, nemlig de perifere enheder.

INPUT–OUTPUT

Nogle mennesker synes, der er vældig interessant at tænke på de nuller og ettaller, der i form af spændinger suser rundt i en datamat; men den store betydning har det ikke, før det kan udnyttes. Vi må altså på en eller anden måde etablere en forbindelse mellem datamaten og omverdenen. Sagt på nu-dansk: den må forsynes med in- og output.

Det lyder så enkelt, men i praksis er det ofte denne funktion, som lægger beslag på størstedelen af en datamats konstruktionstid, fordi den er så stærk afhængig af, hvad maskinen skal anvendes til.

Vi vil dog - i denne omgang - indskrænke emnet væsentligt, idet vi vil koncentrere os om forbindelsen menneske – datamat.

Dette vil i praksis sige, at vi skal se nærmere på, hvorledes vi skaber forbindelse mellem CPU'en og den ydre enhed, der danner de tegn, som vi mennesker kan opfatte.

Som bekendt er vor Z–80'er en 8-bit maskine, som altså arbejder med 8 bit i parallel. Dette betyder, at når vi beder CPU'en om at udføre en I/O-funktion, vil det ske med alle 8 bit på en gang.

Hvis nu den tilsluttede enhed også er beregnet til at arbejde med 8 bit i parallel, passer tingene sammen, og vi har en parallel I/O-port.

Men hvad nu, hvis der er langt mellem den ydre enhed og datamaten? Ja, så kan man for eksempel "binde" dem sammen ved hjælp af en telefonlinie; men på en sådan linie kan der kun overføres 1 bit ad gangen. Hvis alle 8 bit skal overføres, må de derfor sendes efter hinanden, det vil sige i serie.

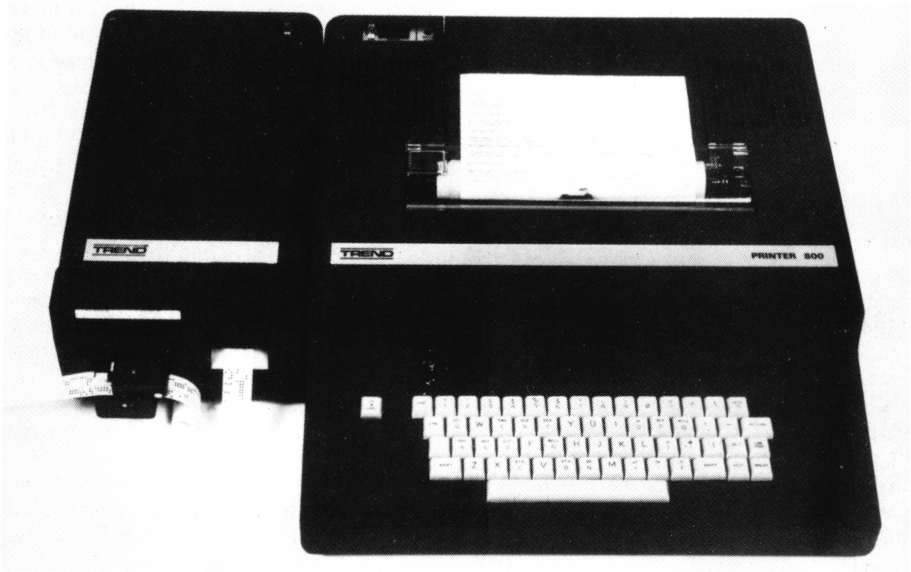
Det lyder enkelt, men er faktisk en temmelig kompliceret sag; først og fremmest fordi metoden kræver en eller anden form for synkronisering mellem sender og modtager. Alligevel er det nok den mest brugte form for dataoverførelse, og det er denne form, vi denne gang skal beskrive et kort til. Først må vi dog se lidt på, hvad det er for ydre enheder, der kan være tale om.

PERIFERE ENHEDER

Brochurer fra store EDB-firmaer indeholder næsten altid et farvebillede af en eller anden stor installation. Som regel er hovedparten af udstyret på disse billeder det, som man kalder perifere - dvs. ydre enheder. Selve datamaten er gerne en beskeden kasse i baggrunden.
De ydre enheder har mange former. Det kan være båndstationer, disk, dataskærme, linieprintere og meget andet.
For de fleste amatører er det nok en drøm at få fat i nogle af disse, men man får desværre en brat opvågning, når man hører prisen. Den billigeste dataskærm koster således omkring 7000 kroner - og ligger dermed nok udenfor de flestes rækkevidde.

Dette har ført til en søgen efter andre muligheder; en søgen, som må siges at være kronet med held, idet der idag er en række muligheder.
En af de bedste - og desværre lidt foragtede - er en telexmaskine. Disse enhede har i en årrække været brugt blandt radioamatører og handles ofte til priser omkring 400-500 kroner.
Fordelen ved disse maskiner er, at de skriver på papir, og at man derfor kan få en samlet udskrift af sit program. Desuden er mange af dem forsynet med papertape-puncher, så programmer kan gemmes og genindlæses i datamaten efter behov.
Udover at de er langsommere, har disse maskiner andre ulemper. De anvender en 5-bit kode (Baudot) og har et andet karaktersæt, end den anvendte standard, som er ASCII. Passende software tager dog let højde for noget sådant.
Vi har selv i flere år haft fornøjelse af denne mulighed, og finder, at det er en god måde at starte på.

Professionelt perifert udstyr som f.eks. denne printer med tastatur, er desværre stadig ret kostbart. Denne engelske terminal, som distribueres af Fa. Trend i Køge, koster kr. 12.600 excl. moms. Dejlig, men dyr.



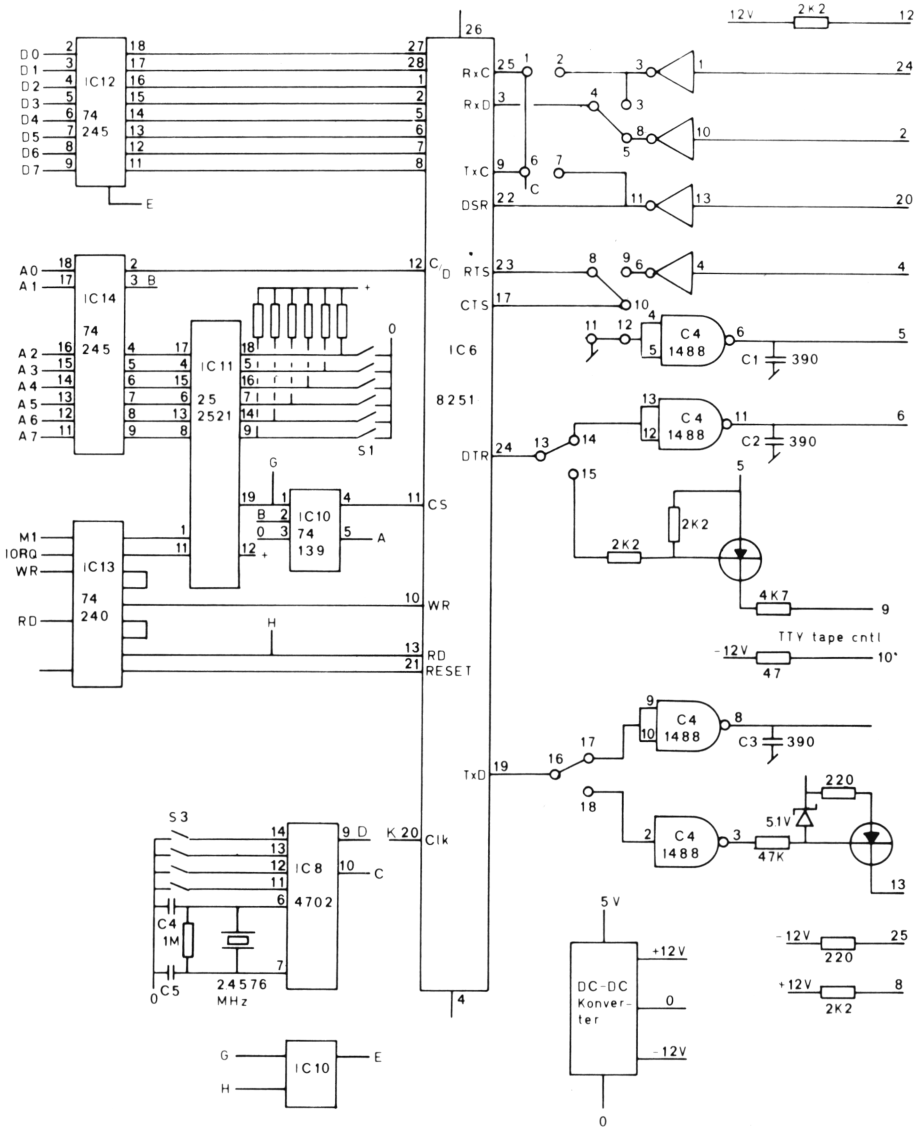
Mange mennesker har et fjernsyn stående. Da principperne i dette er meget lig dem, der anvendes i dataskærme, var det nærliggende at søge disse anvendt.
Desværre betyder den begrænsede båndbredde, at det er vanskeligt at få mere end 16 linier med hver 32 eller 64 karakterer på skærmen, hvilket gør det vanskeligt at bevare oversigten i blot et lidt større program.
Fordelen ved disse enheder er, at de er hurtige og relativt billige, men for den seriøst arbejdende kan de ikke erstatte muligheden for en samlet udskrift.
Endelig er der blinkende lamper, som ses på en del datamater. De ser jo meget imponerende ud og kan være til stor hjælp, nå man afprøver hardware, men til programafvikling anser vi dem for at være helt uegnede.
Dette var de nærliggende muligheder, men der findes også andre, f.eks. en brugt flexowriter, som mange firmaer i øjeblikket smider ud, fordi man går over til EDB.
Lad os et øjeblik vende tilbage til telexmaskinen. Den er beregnet til at arbejde med data på serieform, og det er grunden til, at vi denne gang ser nærmere på SIO-kortet.
Kortet kan dog ikke direkte styre en telex, da disse er beregnet til at arbejde med 110V på datalinierne. Der må derfor indskydes et par transistorer mellem kortet og telexmaskinen.
Selve kortets udgang er udført efter standarden RS-232, som også kaldes V24+V28.

PRINCIP

Datamaten arbejder jo i parallel, men vor udgang skal sende bittene efter hinanden. Der må altså ske en omsætning. Dette kan man lade datamaten selv udføre ved at skifte alle bittene sidelæns f. eks. hen i DO og bruge denne som serieudgang, men det forhindrer CPU'en i at udføre andet nyttigt arbjede så længe.
En bedre løsning er derfor at bruge en såkaldte USART, hvilket betyder Universal Synchronous/asynchronous Receiver/Transmitter. Denne enhed er beregnet til at modtage data fra CPU'en i parallel, omsætte dem til serieform, tilføje start- og stopbit og udsende det hele.
Tilsvarende kan den modtage data på serieform, fjerne start- og stopbit og præsentere data i parallel form til CPU'en.
Der findes en del forskellige af denne kredstype på markedet, men den mest naturlige at anvende i vor datamat, er den kreds, som hører sammen med Z-80 CPU'en. Dette er en fremragende kreds, som kun har én fejl: Den koster næsten kr. 600,—.
I denne omgang bruger vi derfor en anden kreds, nemlig 8251, som koster under kr. 100,— og leveres i et antal forskellige fabrikater.
Denne kreds er den såkaldte programmerbare type, hvilket betyder, at den efter ordre fra CPU'en kan udføre omsætningen til/fra serieform efter forskellige standarder.
Kredsen kan umiddelbart benyttes sammen med Z-80, men der er dog ét problem, idet den ikke direkte kan indpasses i Daisy-chain princippet. Vi har derfor helt fjernet interrupt fra dette kort, og CPU'en må spørge på portens status, før en I/O operation kan udføres.
Kortet indeholder 2 komplette kanaler, men kun én behøver at blive monteret. Den anden kan så færdiggøres, når man får behov for en kanal mere.

DIAGRAMMET

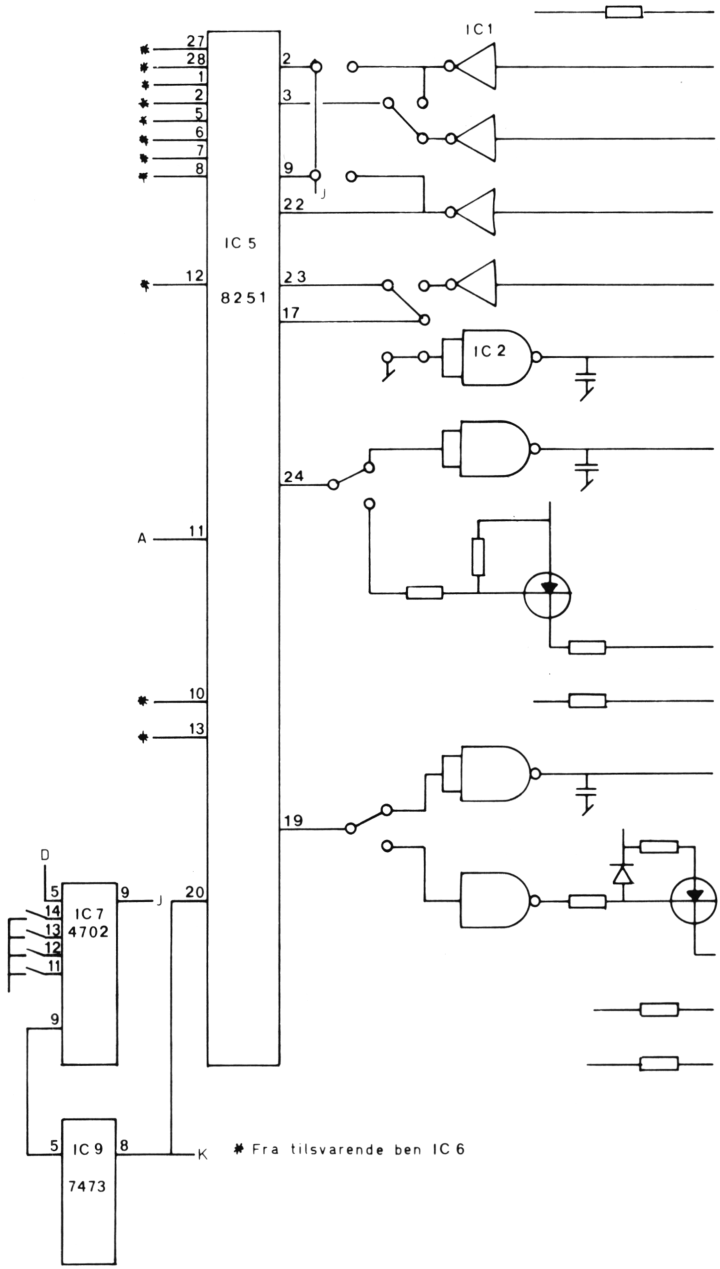
For den, der har fulgt denne serie, byder diagrammet nok ikke på de store overraskelser. Data-, adresse- og kontrollinier føres gennem en driver ind på kortet. Datalinierne, adresselinier A0, RD og WR samt RESET føres direkte til de 2 USART'er. Adresselinierne A2-A7 føres til en komparator, hvor de sammen med kontrollinierne M1 og IORQ afgør, når kortet skal anvendes.
Hvilken af de 2 USART'er, der skal benyttes, afgøres af adresselinien A1 sammen med udgangen fra komparatoren. Disse linier føres til en 2-4 linier dekoder (74LS139), hvis udgang 4 og 5 føres til CS på de 2 USART'er.



Disse har hver sin clock-generator, som styres af et fælles krystal. Herved opnås, at de 2 udgange kan arbejde med hver sin hastighed (baud-rate). Dette indstilles med en DIL-omskifter.

Styringen af datadriveren sker med den anden halvdel af 2-4 linier dekoderen, som omstilles, når kortet er udpeget og RD aktiv, altså i nul.

Dette var processor-siden af kortet. Den anden side – mod den perifere enhed – har måske et lidt rodet udseende. Det



skyldes, at man ved hjælp af lus kan skifte mellem transmissionsstandarder, nemlig RS-232, current loop og synkron. Når du modtager kortet, er det første „indbygget" i det trykte kredsløb, og disse forbindelser må derfor overskæres, når en af de andre former skal i brug.

RS-232 kræver ± 12 volt. For ikke at skulle lave specielle strømforsyninger til

disse spændinger, er der på kortet plads til en DC-DC konverter, som klarer problemet. Kortet skal altså blot forsynes med + 5 volt.
Hvis du kun vil montere én kanal, skal det være den side med IC6. Desuden skal der i IC7 lægges en lus mellem ben 5 og ben 9.

KONSTRUKTION

Denne udføres som for de andre kort, og det er næppe nødvendigt at gentage instruktionerne.
Dog skal siges, at 825 og 4702 er MOS-kredse, og de skal derfor behandles forsigtigt.

AFPRØVNINGEN

Også denne udføres som på de andre kort. Du begynder altså med at konstatere, at alle linier arbejder frit og er i overensstemmelse med bussen. Et særligt problem er clock-generatoren. Den arbejder ved 2,4 MHz, og man kan derfor kun kontrollere den med oscilloscope.
Dette signal deles med 2 i IC9 og bruges af 8251 på ben 20.
Når du er sikker på, at kortet er iorden, kan du montere krystallet på CPU-kortet. Anbring en afbryder mellem backplanens reset-linie og stel. Så kan datamaten startes. Resten afhænger af den soft-ware, som er indlagt.

EFTERSKRIFT

Overskriften betyder ikke, at vi er færdige med datamaten. Vi har stadig mange kort, vi skal igennem, men nu, hvor vi er færdige med de grundliggende funktioner, er det måske godt at stoppe op et øjeblik.
Vi har i den forløbne tid beskrevet et antal kort, som tilsammen danner en komplet datamat. Disse kort er udvalgt efter, hvad der giver den billigste start på en fleksibel amatør-datamat.
Fremover vil vi tage mindre hensyn til korts færdige pris og lægge vægt på funktionen. Det er ikke fordi, at vi venter, at mange amatører vil bygge f.eks. 32K RAM-kortet, men dels er prisudviklingen umulig at forudsige, og det kan om et år være billigere at bygge et 32K kort end det tidligere beskrevne 4K RAM kort. Endvidere finder vi, at det er vigtigt, at de, som går i gang med denne datamat, kan vælge en opbygning, der passer bedst til den ønskede anvendelse.
Hvor mange kort, der ialt bliver, ved ingen, men det skal ikke være nogen hemmelighed, at det er vort mål at gøre denne datamat til den stærkeste – og mest fleksible – amatør-datamat på markedet. Så må tiden vise, om det lykkes!

SOFTWARE

Udviklingen har på dette punkt været med Z-80. Der er i den sidste tid fremkommet et spændende udvalg af nye programmer, som let kan gøre indehavere af andre CPU'er misundelige. De fleste af disse programmer arbejder både på 8080 og Z-90, men er hurtigst og fylder mindst på sidstnævnte.
Det foreløbige højdepunkt er højniveausproget PASCAL, som nu er tilgængeligt, men også FORTRAN kan fås. Dette er en udvikling, som rummer helt nye muligheder. MP/SS

Som efterskriften antyder, er dette den foreløbige afslutning på det spændende Z-80 selvbyggerprojekt. Det betyder dog ikke, at projektet står stille, da der fortsat arbejdes ihærdigt med både hard- og software. Således er der en interperter til struktureret BASIC undervejs, og der arbejdes med muligheden af at levere færdigsamlede kort til amatører, som gerne vil have en effektiv datamat, men som helt ikke selv vil bygge den. Interesserede bedes fortsat kontakte Mogens Pelle, Birkhøj Terrasserne 416 C, 3520 Farum.

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: . . . . . .

. . . . . .

BREVKORT

Husk afsender

Porto
100
øre

Til:

Telepress ApS

Greve Strandvej 42
2670 Greve Strand

KLIP LANGS STREGERNE HELT TIL BLADETS KANT

BREVKORT

Husk afsender

Porto
100
øre

Til:

Telepress ApS

Greve Strandvej 42
2670 Greve Strand