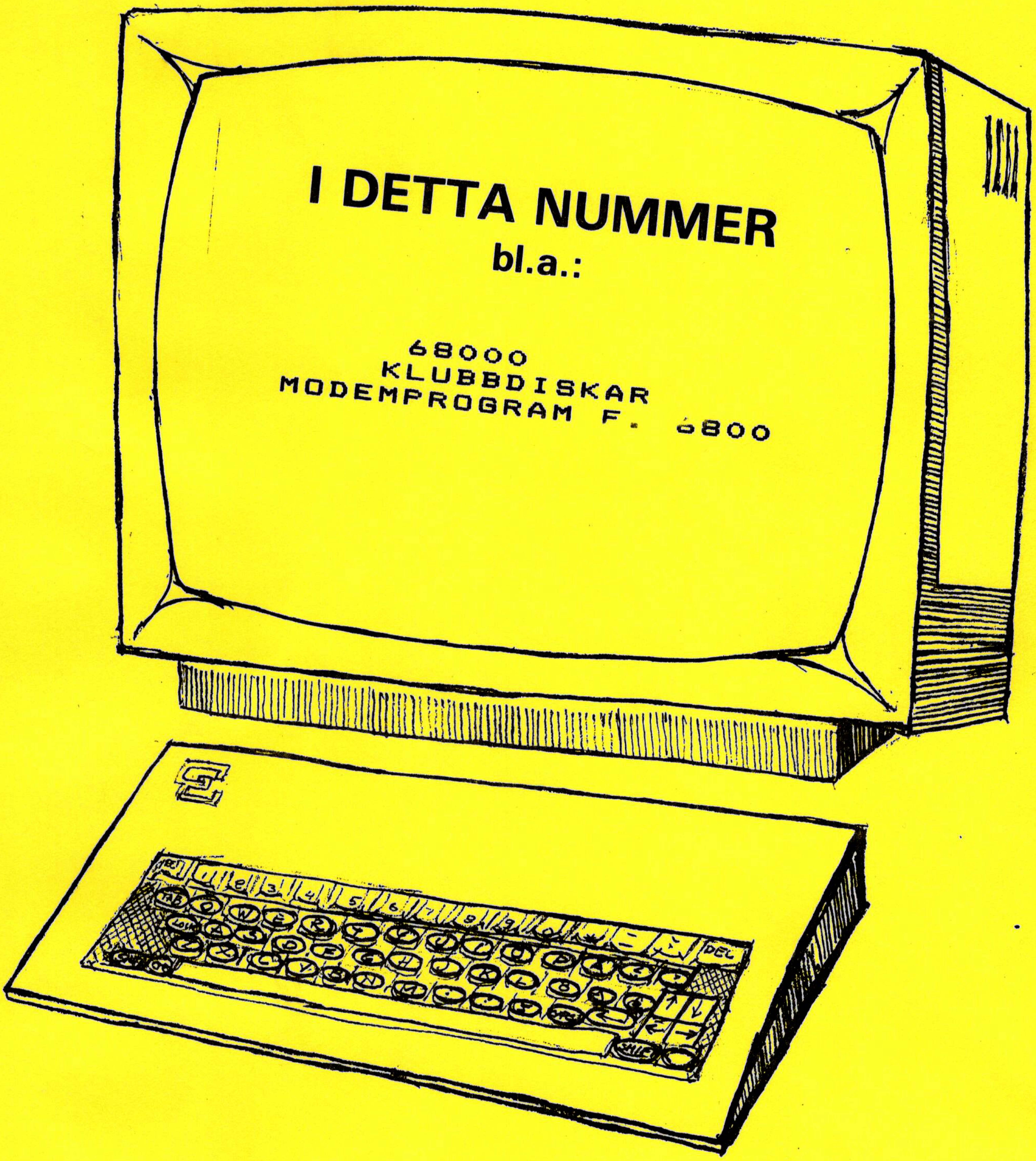
MPU-laren. Nr 3 1984

Tidskrift för Privatdataklubben PD68 Årgång 7



DEXTRAFERM
har donerat ett 75/1200 modem
till PD68.

PD68 klubbdator telefon:
75/1200 = 08 - 479 949
300 = 08 - 998 716

PRIVATDATAKLUBBEN
PD 68

BOX 1098, 122 21 ENSKEDE PG: 96 04 68-7

REDAKTION

Bernt Petzäll (BS) Red. Chef
Franserudsvägen 56
666 00 BENGTSFORS 0513-12466
(FLEX 2/9)

Roger Holmstrand (RH) Annons Chef
Rapsodivägen 168
142 00 TRÅNGSUND 08-7713852
(FLEX 9, OS9)

Gunnar Ejemoh (GE)
Terapivägen 14B 7tr
141 56 HUDDINGE 08-7747760
(FLEX 9)

Lennart Uusitalo (LUU)
Skrakvägen 20
141 72 HUDDINGE
(FLEX 9)

Sonny Taberman (SJT)
Ripstigen 5
171 73 SOLNA 08-853054
(FLEX 9, APPLE 2&3)

Anders Holm (LH) 019-96375
Ingelstavägen 11
710 15 VINTROSA

STYRELSE

Sven-Erik Ringström (SER) Ordf.
Högbyvägen 18
175 46 JÄRFÄLLA 0758-16228

Bo-Erik Sandholm V.ordf.
Lingvägen 217
123 59 FARSTA 08-933266

Ingemar Skarpås Sekr.
Töjnavägen 42A
191 44 SOLLENTUNA 08-969696

Leif Ulfström (LUL) Kassör
Fornhöjdsvägen 70
151 58 SÖDERTÄLJE 0755-18772

Ulf Holm V.sekr.
Artillerigatan 69B 3tr
114 45 STOCKHOLM 08-7830501

Tommy Bladh (TBL) Suppl.
Norrholmsvägen 106
132 00 SALTSJÖ-BOO

Åke Holm (ÅH) Suppl.
Långsjövägen 15B
135 54 TYRESÖ

Roger Holmstrand (RH) Suppl.
Rapsodivägen 168
142 00 TRÅNGSUND 08-7713852

ANNONSER

Helsida	500:-
Halvsida	300:-
Kvartssida	200:-
Bilaga	Enl.ö.k.

Priserna gäller tryckfärdigt material. Medlemmar får sätta in privata radannonser gratis

NYA MEDLEMMAR

Du som vill bli medlem i PD68 !
Sätt in 125:- på postgiro 960468-7.
Glöm ej skriva namn, adress och tel.

SNART ÅRSMÖTE

DU som känner dig hågad att ställa upp som kandidat till någon post inom klubben, SKRIV till valberedningen.

MATERIAL TILL TIDNINGEN.

Alla bidrag STORA, små, korta eller långa är välkomna.

Om möjligt, skriv spalten med 40 kolumner !
Ni som har möjlighet bör skicka materialet på flexskiva.
FORMAT: Enkel/dubbelsidigt, enkel packningstäthet och 35/40 spår.
Flexskivan retuneras.

KLUBBDATOR

Tel.nr. till klubbdatorn:

08-998 716 300 baud fdx.
08-479 949 75/1200 baud fdx.

OBS! KRÄVER LÖSENORD.

Synpunkter på klubbdatorn sändes till:
PD68
Klubbdatorgruppen
BOX 1098
122 21 ENSKEDE

ORDFÖRANDEN HAR ORDET

Ordföranden har ordet.

1984 ett årtal med en speciell innebörd tack vare Orwell. Farhågorna har förstås inte besannats. Bästa sättet att undvika datoriseringens nackdelar är kunskap åt så många som möjligt. Kunskap om vad datorer är. Kunskap om program och systemuppbyggnad.

För PD 68 blev inte året så aktivt. Ett mindre antal aktivister håll-er skutan flytande. En av dem är Gunnar Ejemoh, som nu tagit ett fast grepp om 68k projektet för vilket han gör en jätteinsats. Intresset för den snabba beställningen före jul, blev över förväntan stort (70st). Därför ser det ut som 1985 skulle bli 68000 året för PD 68.

Ett önskemål från många medlemmar, har varit en 75/1200 anslutning till klubbdatorn. Denna finns nu på plats. Detta föranleder ett tack även till klubbdatorgruppen för stronga insatser under året.

SER

INNEHALLSFÖRTECKNING

68000

MJ

PROJEKTET.

Anmälningstiden för köp av 68000-systemet har nu gått ut. För att hålla er ajour, kommer här lite information :

Vi hade i försiktig optimism räknat med 8-10 beställningar i första omgången. Detta dels beroende på att det är ett alldeles nytt projekt, dels för att det skulle bli lätt att administrera.

Döm om vår förvåning (och fasa) när det ramlar in 73 !!! beställningar.

Här finns tydligen ett enormt uppdämt sug efter något nytt projekt inom klubben.

Nåväl, detta har inneburit att vi måste lägga upp hela proceduren på ett helt annat sätt, inte minst administrativt, för att skydda beställaren.

Steg ett blev att vi (Gunnar Ejemoh och Anders Karlsson) fick sätta oss i bilen och skumpa iväg till leverantören (en resa på 132 mil !!).
Detta för att diskutera igenom alla punkter och för att kunna sätta upp ett köpeavtal som ger oss en riktig leverans.

Vi testkörde systemet varvid vi fann några mindre problem som leverantören dock lovade åtgärda innan vi undertecknade köpeavtalet.
På detta sätt hoppas vi slippa problem längre fram.

I övrigt verkar erbjudandet mycket prisvärt med en kraftfull programvara.

Det stora köpintresset har dock skapat vissa leveransproblem av korten, men c:a 35-40 kort kommer att levereras senast vecka 5 och resten c:a vecka 9.

För de som inte hann vara med i omgång 1 av 68000-köpet kommer en ny omgång att ordnas. Intresserade bör höra av sig till Gunnar Ejemoh senast 850220.
Leveranstid för denna omgång (vid rimligt antal) c:a 3-4 veckor.

*** ANNONS - SÄLJES ***

N-Y-Å-R-S-R-E-A.
================

1/ RT-dator i 19 tums svart Vero-låda med fläkt. 57 K-byte RAM, serieinterface: 2 portar, printerkort, timerkort, promkort, klockkort, 2 st floppyinterface - en för enkel och en för enkel/dubbel density, en inbyggd Shugart-drive (plats finns för två). Pris kr 3900:-

2/ Terminal I: Best. av Alfaskop monitor och 75-19200 baud terminalkort. Grafik 480X240 som option med härför lämpligt prom. Även text/blockgrafik. Pris kr 1400:-

3/ Terminal II: Alfaskop monitor,75-19200 baud terminalkort enligt ovan samt separat tangentbord (extra låg modell). Pris kr 1950:-

4/ Terminal III: CA-elektroniks RT-terminal med grafik. Kortet inbyggd i låda med "Keytronic" tgbord. Komplett med nätanslutning. Utan monitor. Pris kr 750:-

5/ Terminal IV: Grafikterminalkort "TERM" på Europakort 100x160 mm.Tangentbord- och datoringång för RS-232. Passar därför de flesta datorer. Upplösning: 512 x 256. Kan även samtidigt anv. som skrivterminal med 24 rader, 80 tecken/rad utan prombyte. 8+2 K-byte RAM (4164,6116) Säljes utan 8 K-byte RAM. Uförlig manual medföljer. Pris Kr 1350:-

6/ Monitor Philips, V7001 grön display. Med ljudingång. Använd högst 10 timmar. Garantitid kvar. Pris kr 700:-

7/ Mjukvara: UCSD-PASCAL-SYSTEM. Har eget operativsystem anpassat för FLEX-6809. Compiler, flera editorer till ex. en förstklassig lineeditor med "GOTOXY-rutiner" som även används av kompilatorn vid felangivning. Även kompilerande basic och kalkylatorprogram m.m. Alltsammans i 4st 5 1/4" disketter och manual av tegelstensformat. Priset i USA: Dollar 419:-
Säljes för kr 1700:-

8/ Diverse 68xx-system-IC: 40- och 24-benare (t ex 2 st MC 14411), kristaller. Billigt!

Pris kan diskuteras vid komb. av ovanstående alternativ, spec komb 1 o 6 (Välfungerande måttsydd komb.).

Gunnar Lovius, tel 08 - 777 15 60.

BOKFöRINGSPROGRAM I BASIC

Här är ett litet basicprogram för dokföring. Programmet heter BOKB, och lagrar information på filen "UTG.DAT". Om inte lagringsfilen existerar skapas den första gången man väljer inmatning vid huvudmenyn. Därav följer att man SKALL välja INMATNING vid första tillfället man kör BOKB.

Programmet är tänkt att användas för lagring av information och för att få en snabb summering av lika poster för en viss tidsperiod.

Programmet är uppbyggt med menyer enl:

MENY

1. INMATNING
2. LISTNING
3. SUMMERING
4. äNDRING
9. AVSLUTA

Under INMATNING följer sedan interaktiv inmatning av : DATUM
POSTRUBrik
BELLOP

LISTNING ger som resultat att allt som är lagrat i "UTG.DAT" listas på skärmen

SUMMERING innehåller en undermeny enl:

1. TOTALA SUMMAN
2. SUMMAN AV LIKA POSTER - ALLA
3. SUMMAN AV LIKA POSTER - VISSA

Samtliga SUMMERINGsrutiner innehåller en förfrågan om vilken tidsperiod som avses.
Punkt ett talar för sig själv.
Punkt två ger som resultat att alla poster skrivs ut på skärmen och om två eller flera poster är lika summeras deras bellop och den posten skrivs bara ut en gång med det summerade bellopet.

Under punkt tre får man skriva in de vissa poster man vill ha utskrivna, och de övriga kommer summerade under övrigt.

Totala summan skrivs även ut på punkt två och tre.

Trycker man på 4 i huvudmenyn för äNDRING frågas efter vilket datum man vill ändra i och sedan skrivs det gamla ut på skärmen varefter man får skriva in det nya.

★

```
10 Y=84
20 DIM H¤(365)
30 GOTO 180
40 REM ** ERROR-RUTIN **
50 IF ERR=2 THEN PRINT "Filen redan
öppnad"
60 IF ERR=4 THEN PRINT "Filen existerar
ej"
70 IF ERR=7 THEN PRINT "Skiva är fylld"
80 PRINT "Fel"; ERR;" på rad";ERL
90 REM ** SLUT ERROR-RUTIN **
100 RESUME 180
110 REM ** SUBRUTIN F
120 REM ** AV LAGRINGSFILENS FÄLT **
130 REM ** INDELNING.            **
140 REM
150 FIELD #1, R*18 AS Z¤, 6 AS D¤, 8 AS
R¤, 4 AS B¤
160 REM ** SLUT SUBRUTIN         **
170 RETURN
180 PRINT : PRINT "    MENY"
190 PRINT
200 PRINT "1. Inmatning"
210 PRINT "2. Listning"
```

```
220 PRINT "3. Summering"
230 PRINT "4. Ändring"
240 PRINT "9. Avsluta"
250 PRINT : INPUT "Välj siffra ";S
260 IF S<1 OR S>9 GOTO 180
270 ON S GOSUB 290,920,1180,3760,4170,4180,4190,4195,4200
280 GOTO 180
290 REM ** SUBRUTIN  INMATNING **
300 REM **                     **
310 OPEN OLD "UTG.DAT" AS 1
320 ON ERROR GOTO 670
330 R=0
340 GOSUB 140
350 GET #1,RECORD 1
360 A=VAL(D¤)
370 SE=INT(A/14+.94)
380 R=A-(SE-1)*14
390 REM
400 IF R<=13 THEN 480
410 SE=SE+1
420 FOR R=0 TO 13
430 LSET D¤="" : LSET R¤="" : LSET B¤=""
440 PUT #1, RECORD SE+1
450 NEXT R
460 R=0
470 PUT #1, RECORD SE
480 PRINT "Ge datum (ex. 840909) S=Slut"
490 INPUT D1¤
500 IF D1¤="S" OR D1¤="s" GOTO 820
510 IF VAL(RIGHT¤(D1¤,2)) > 31 OR VAL(RIGHT¤(D1¤,2)) < 1 THEN 480
520 IF VAL(MID¤(D1¤,3,2)) > 12 OR VAL(MID¤(D1¤,3,2)) < 1 THEN 480
530 PRINT "Skriv postrubrik (Max 8 tecken)"
540 INPUT R1¤
550 IF LEN(R1¤) > 8 GOTO 530
560 PRINT "Ge bellop (1-9999)"
570 INPUT B1¤
580 IF VAL(B1¤)> 9999 OR VAL(B1¤) < 1 THEN 560
590 GOSUB 140
600 GET #1, RECORD SE
610 LSET D¤=D1¤
620 LSET R¤=R1¤
630 LSET B¤=B1¤
640 PUT #1, RECORD SE
650 R=R+1
660 GOTO 390
670 IF ERR<>4 THEN 40
680 PRINT : PRINT "Skapar en ny fil !"
690 CLOSE 1
700 OPEN NEW "UTG.DAT" AS 1
710 R=0
720 GOSUB 140
730 LSET D¤=STR¤(1)
740 PUT #1, RECORD 1
750 FOR R=0 TO 13
760 LSET D¤="" : LSET R¤="" : LSET B¤=""
770 PUT #1, RECORD 2
780 NEXT R
790 R=0
800 A=1
810 GOTO 370
820 REM ** AVSLUTNING AV INMATNING **
830 RE=R : R=0
840 AN=(SE-1)*14+RE
850 GOSUB 140
860 GET #1, RECORD 1
870 LSET D¤=STR¤(AN)
880 PUT #1, RECORD 1
890 CLOSE 1
900 REM ** SLUT SUBRUTIN INMATNING **
910 RETURN
920 REM ** SUBRUTIN LISTNING **
930 PRINT CHR¤(27);CHR¤(5);"  LISTNING"
940 OPEN OLD "UTG.DAT" AS 1
950 ON ERROR GOTO 1140
960 R=0
970 GOSUB 140
980 GET #1, RECORD 1
990 PRINT :PRINT "Dat.";TAB(12);"Post";TAB(22);"Bellop"
1000 R=VAL(D¤)
1010 R1=R
1020 SE=INT(R/14+.93)
1030 PRINT
1040 FOR J=1 TO SE
1050 IF J=SE THEN RE=R1-1-(SE-1)*14 ELSE RE=13
1060 FOR I=0 TO RE
1070 IF J=1 AND I=0 THEN 1110
1080 R=I : GOSUB 140
1090 GET #1, RECORD J
1100 PRINT D¤;TAB(12);R¤;TAB(23);B¤
1110 NEXT I
1120 NEXT J
1130 GOTO 1150
1140 GOSUB 40
1150 CLOSE 1
1160 REM ** SLUT SUBRUTIN LISTNING **
1170 RETURN
1180 REM ** SUMMERING **
1190 PRINT CHR¤(27);CHR¤(5);"SUMMERING"
1200 PRINT : PRINT
1210 PRINT "1. Enbart totala summan"
1220 PRINT "2. Summan av lika poster - ALLA"
1230 PRINT "3. Summan av lika poster - VISSA"
1240 PRINT "4. Slut summering"
1250 S=0
1260 PRINT : INPUT "Siffra ";W
1270 IF W>4 OR W<1 THEN 1260
```

```
1280 IF W=4 GOTO 1320
1290 ON W GOSUB 1880, 2320, 3020
1300 REM ** SLUT SUBRUTIN SUMMERING **
1310 GOTO 1200
1320 RETURN
1330 REM ** subrutin för inmatning **
1340 REM ** av intervall           **
1350 REM ** TILLDELAR F1 och A1    **
1360 PRINT : PRINT "Skriv första datumet
i intervallet"
1370 INPUT F
1380  IF  VAL(LEFT¤(STR¤(F),3))  < 84 OR
VAL(LEFT¤(STR¤(F),3)) > 84 THEN 1360
1390  IF  VAL(MID¤(STR¤(F),4,2))  <1  OR
VAL(MID¤(STR¤(F),4,2)) >12 THEN 1360
1400  IF  VAL(RIGHT¤(STR¤(F),3))  <1  OR
VAL(RIGHT¤(STR¤(F),3)) >31 THEN 1360
1410 PRINT : PRINT "Skriv andra datumet"
1420 INPUT A
1430  IF  VAL(LEFT¤(STR¤(A),3))  <  1 OR
VAL(LEFT¤(STR¤(A),3)) > Y THEN 1410
1440  IF  VAL(MID¤(STR¤(A),4,2))  <1  OR
VAL(MID¤(STR¤(A),4,2)) >12 THEN 1410
1450  IF  VAL(RIGHT¤(STR¤(A),3))  <1  OR
VAL(RIGHT¤(STR¤(A),3)) >31 THEN 1410
1460 IF A<F THEN 1410
1470 OPEN OLD "UTG.DAT" AS 1
1480 ON ERROR GOTO 1820
1490 R=0
1500 GOSUB 140
1510 GET #1, RECORD 1
1520 AN=VAL(D¤)
1530 IF AN<10 GOTO 1540
1540 SE=INT(AN/14+.94)
1550 FOR I=1 TO SE
1560 IF I=SE THEN RE=AN-1-(SE-1)*14 ELSE
RE=13
1570 FOR R=0 TO RE
1580 IF I=1 AND R=0 THEN 1620
1590 GOSUB 140
1600 GET #1, RECORD I
1610 IF F=VAL(D¤) THEN 1660
1620 NEXT R
1630 NEXT I
1640 PRINT "Inget så sent datum"
1650 GOTO 1860
1660 F1=I
1670 R1=R
1680 FOR I=F1 TO SE
1690 IF I=F1 THEN ST=R1 ELSE ST=0
1700 IF I=SE THEN RE=AN-1-(SE-1)*14 ELSE
RE=13
1710 FOR R=ST TO RE
1720 GOSUB 140
1730 GET #1, RECORD I
1740 IF A=VAL(D¤) THEN 1790
1750 NEXT R
1760 NEXT I
1770 PRINT "ingen A1"
1780 GOTO 1860
1790 A1=I
1800 R2=R
1810 GOTO 1860
1820 REM ** ANROPA ERROR-RUTIN **
1830 GOSUB 40
1840 REM ** slut subrutin för *
1850 REM ** inmatning av interv. **
1860 CLOSE 1
1870 RETURN
1880 REM ** SUBRUTIN F
1890 REM ** SUMMAN                **
1900 PRINT CHR¤(27);CHR¤(5)
1910 PRINT "Totala summan"
1920 REM ** ANROPA SUBRUTIN F
1930 REM ** TILLDELNING AV VAR. **
1940 REM ** F1 och A1           **
1950 GOSUB 1330
1960 OPEN OLD "UTG.DAT" AS 1
1970 ON ERROR GOTO 2250
1980 FL=0
1985 S=0
1990 FOR K=F1 TO A1
2000 K1=K
2010 REM ** HÄMTA RS & RF  **
2020 GOSUB 2050
2030 GOTO 2160
2040 REM ** SUBRUTIN F
2050 IF F1<>A1 THEN 2080
2060 RF=R1 : RS=R2
2070 GOTO 2150
2080 IF K1<>F1 THEN 2110
2090 RF=R1 : RS=13
2100 GOTO 2150
2110 IF K1=A1 THEN 2140
2120 RF=0 : RS=13
2130 GOTO 2150
2140 RF=0 : RS=R2
2150 RETURN
2160 FOR R=RF TO RS
2170 GOSUB 140
2180 GET #1 ,RECORD K
2190 S=S+VAL(B¤)
2200 NEXT R
2210 NEXT K
2215 IF FL=1 THEN 2300
2220 PRINT
2230 PRINT "Totala summan = ";S
2240 GOTO 2270
2250 REM ** ANROP AERROR-RUTIN **
2260 GOSUB 40
2270 REM
2280 REM ** SLUT SUBRUTIN SUMMERING **
2290 REM ** AV SAMTLIGA POSTER      **
2300 CLOSE 1
2310 RETURN
2320 REM ** SUBRUTIN F
```

```
2330 REM ** AV ALLA LIKA POSTER      **
2340 PRINT CHR¤(27);CHR¤(5)
2350 PRINT "Summan av alla lika poster"
2360 PRINT
2370 REM ** anropa subrutin som *
2380 REM ** ger parametrarna F1 *
2390 REM ** och A1, samt R1 & R2 **
2400 GOSUB 1360
2430 PRINT
2440 PRINT "Post","Summa"
2450 PRINT
2460 OPEN OLD "UTG.DAT" AS 1
2470 ON ERROR GOTO 2950
2480 R=0 : GOSUB 140
2490 GET #1, RECORD 1
2500 AN=VAL(D¤)-1
2510 IF AN<365 GOTO 2540
2520 PRINT "för stort intervall !"
2530 GOTO 3090
2540 SE=INT(AN/14+.94)
2550 FOR I=0 TO AN
2560 H¤(I)=""
2570 NEXT I
2580 S=0
2590 H=0
2600 FOR K=F1 TO A1
2610 REM ** HÄMTA RS & RF      **
2620 K1=K
2630 GOSUB 2050
2640 FOR RG=RF TO RS
2650 R=RG
2660 GOSUB 140
2670 GET #1, RECORD K
2680 R1¤=R¤
2690 FOR M=0 TO H
2700 IF R1¤=H¤(M) THEN 2910
2710 NEXT M
2720 REM ** R¤ finns inte i H¤ **
2730 REM ** alltså ej printad. **
2740 REM ** Då printar vi den då **
2750 FOR L=F1 TO A1
2760 REM ** hämta RS & RF   **
2770 K1=L
2780 GOSUB 2050
2790 FOR RE=RF TO RS
2800 R=RE
2810 GOSUB 140
2820 GET #1, RECORD L
2830 IF R1¤<>R¤ THEN 2860
2840 S=S+VAL(B¤)
2850 H¤(H)=R¤
2860 NEXT RE
2870 NEXT L
2880 H=H+1
2890 PRINT R1¤, S
2900 S=0
2910 NEXT RG
2920 NEXT K
2930 GOSUB 1980
2940 GOTO 2970
2950 REM ** ANROPA ERROR-RUTIN **
2960 GOSUB 40
2970 PRINT : PRINT
2980 REM ** SLUT SUBRUTIN F
2990 REM ** SUMMERING ALLA LIKA **
3000 REM ** POSTER.             **
3010 RETURN
3020 REM ** SUMMERING AV VISSA POSTER **
3030 REM ** ANROPA SUBRUTIN F
3040 REM ** TILLDELNING AV        **
3050 REM ** A1, F1, R1 & R2       **
3060 GOSUB 1360
3070 OPEN OLD "UTG.DAT" AS 1
3080 ON ERROR GOTO 3700
3090 R=0 : GOSUB 130
3100 GET #1, RECORD 1
3110 AN=VAL(D¤)
3120 IF AN < 365 THEN 3150
3130 PRINT "För stort intervall !"
3140 GOTO 3720
3150 SE=INT(AN/14+.94)
3160 H=0
3170 INPUT "Skriv post (S=Stopp)";H¤(H)
3180 IF H¤(H)="S" OR H¤(H)="s" THEN 3210
3190 H=H+1
3200 GOTO 3170
3210 H=H-1
3220  PRINT  :  PRINT  "Post","Summa  ";
PRINT
3230 PRINT
3235 S1=0
3240 FOR K=F1 TO A1
3250 REM ** HÄMTA RS & RF  **
3350 K1=K
3360 GOSUB 2050
3370 FOR RG=RF TO RS
3380 R=RG
3390 GOSUB 140
3400 GET #1, RECORD K
3430 R1¤=R¤
3440 FOR M=0 TO H
3450 IF R1¤=H¤(M) THEN 3490
3460 NEXT M
3480 GOTO 3650
3490 H¤(M)=""
3500 FOR L=F1 TO A1
3510 REM ** HÄMTA RS & RF  **
3520 K1=L
3530 GOSUB 2050
3540 FOR RE=RF TO RS
3550 R=RE
3560 GOSUB 140
3570 GET #1, RECORD L
3580 IF R1¤<>R¤ THEN 3600
3590 S=S+VAL(B¤)
3600 NEXT RE
```

```
3610 NEXT L
3620 PRINT R1¤,S
3624 S1=S1+S
3630 S=0
3640 REM
3650 NEXT RG
3660 NEXT K
3662 FL=1
3664 GOSUB 1985
3670 PRINT "
3680 PRINT : PRINT "Totala summan =";S
3690 GOTO 3740
3700 IF ERR<> 4 THEN 50
3710 PRINT "Filen existerar ej !"
3720 CLOSE 1
3730 RESUME 180
3740 REM ** SLUT SUM VISSA POSTER  **
3750 RETURN
3760 REM ** ÄNDRING  **
3770 INPUT "Viket datum vill du ändra";D
3780  IF  VAL(LEFT¤(STR¤(D),3))  <1  OR
VAL(LEFT¤(STR¤(D),3)) >Y THEN 3770
3790  IF  VAL(MID¤(STR¤(D),4,2))  <1  OR
VAL(MID¤(STR¤(D),4,2)) >12 THEN 3760
3800  IF  VAL(RIGHT¤(STR¤(D),3))  <1  OR
VAL(RIGHT¤(STR¤(D),3)) >31 THEN 3770
3810 OPEN OLD "UTG.DAT" AS 1
3820 ON ERROR GOTO 4120
3830 R=0 : GOSUB 130
3840 GET #1 , RECORD 1
3850 AN=VAL(D¤)
3860 SE=INT(AN/14+.94)
3870 PRINT
3880 FOR I=1 TO SE
3890 IF I=1 THEN RF=1 ELSE RF=0
3900 IF I=SE THEN RS=AN-1-(SE-1)*14 ELSE
RS=13
3910 FOR R=RF TO RS
3920 GOSUB 130
3930 GET #1, RECORD I
3940 IF VAL(D¤)=D THEN 3990
3950 NEXT R
3960 NEXT I
3970 PRINT "Datumet finns ej !"
3980 GOTO 4150
3990 PRINT "Postrubrik"
4000 PRINT "Gammalt:";R¤;" Nytt:";
4010 INPUT R1¤
4020 IF LEN(R1¤)>8 THEN 4000
4030 PRINT "Bellop"
4040 PRINT "Gammalt:";B¤;" Nytt:";
4050 INPUT B1¤
4060 IF VAL(B1¤)>9999 THEN 4040
4070 LSET R¤=R1¤
4080 LSET B¤=B1¤
4090 GOSUB 130
4100 PUT #1, RECORD I
4110 GOTO 4150
4120 IF ERR<>4 THEN 40
4130 PRINT "Filen existerar ej !"
4140 RESUME 180
4150 REM ** SLUT ÄNDRING  **
4160 CLOSE 1
4170 RETURN
4180 RETURN
4190 RETURN
4195 RETURN
4200 REM close
4210 END
```

*

MODEMPROGRAM FöR 6800

Äntligen har det kommit !!

På följande sidor finns en programlista för ett modemprogram för 6800 (FLEX 2).
Ett efterlängtat program som tidigare varit få förunnat (dvs. endast 6809-innehavare) kan nu också nyttjas av medlemmar med 6800-processorer.

RED.

```
                  *      MODEMPROGRAM för ELFI-6800.
                  *
                  *      Januari 1984
                  *
                  *      SYNTAX:  MODEM
                  *
                  *  Interna  kommandon  skall föregås av (ESC). Nästa tecken tolkas då som ett kommando och sänds ej.
                  *  Om  (ESC)  skall  sändas  ,  svara  med  (ESC) på frågan 174488"Kommando?".
                  *
                  * Kommandon:  E = Ekoswitch
                  *             F = Retur till FLEX
                  *             S = SAVE. Spara inkommande text i RAM.
                  *             T = TRANSMIT. Sänd en diskfil.
                  *             W = WRITE. Spela in på skivan.
                  *             X = Avbryt.
                  *
AD03              WARMS   EQU   ¤AD03
AD1E               PSTRNG  EQU   ¤AD1E
DC2A              MODEM   EQU   ¤DC2A
E820              ACIAT   EQU   ¤E820
E000              IRQVEC  EQU   ¤E000     ADRESS TILL IRQ-RUTIN
AD18              PUTCHR  EQU   ¤AD18     FLEX char.output routine
FBE3              INCHP   EQU   ¤FBE3
AD1B              INBUFF  EQU   ¤AD1B
AD33              SETEXT  EQU   ¤AD33
AD2D              GETFIL  EQU   ¤AD2D
B403              FMSCLS  EQU   ¤B403
B406              FMS     EQU   ¤B406
A840              SYSFCB  EQU   ¤A840     FILE CONTROL BLOCK
AD3F              RPTERR  EQU   ¤AD3F
AD24              PCRLF   EQU   ¤AD24
FAF8              LFCR    EQU   ¤FAF8
0000              BUFSTA  EQU   ¤0000
9FFF              MEMEND  EQU   ¤9FFF
                  *
A100                      ORG   ¤A100
A100 20 16        START   BRA   STARTO
A102 01                   FCB   1         VER NR
A103 45           ECHO    FCB   'E        E  FöR EKOSWITCH
A104 54           TRANSM  FCB   'T        T  SäND DISKFIL
A105 53           SAVE    FCB   'S        S  SPAR INKOMMANDE TEXT.
A106 57           WRITE   FCB   'W        W  SPAR INKOMMEN TEXT.
A107 58           BREAK   FCB   'X        X  AVBRYT ALLT
A108 46           RETUR   FCB   'F        F  RETUR TILL FLEX.
A109 1B           ESCAPE  FCB   ¤1B
A10A              IRQSAV  RMB   2
A10C              FCBPEK  RMB   2         PEKARE I FCB
A10E              BUFPEK  RMB   2         PEKARE I BUFFERT
A110              TRADR   RMB   2         TRANSFER ADRESS VID RTI
A112 00           EFLG    FCB   ¤0        EJ EKO=0, EKO=FF
A113 00           FNKFLG  FCB   ¤0        FUNKTIONSFLAGGA
A114 00           WAITM   FCB   ¤0        Max time for waiting on MODEM.
A115 00 00        POINTER FDB   BUFSTA
A117 FF           SAVFLG  FCB   ¤FF       0=SPARA INKOMMEN TEXT.
                  *
                  * Initiering och funktionsval.
```

```
                   *
A118 FE E000 STARTO  LDX     IRQVEC
A11B FF A10A         STX     IRQSAV     SPARA IRQ-VEKTORN
A11E CE A224         LDX     #INTER
A121 FF E000         STX     IRQVEC     INTERRUPTRUTIN
A124 86 03           LDA A   #3
A126 B7 E820         STA A   ACIAT
A129 B7 DC2A         STA A   MODEM
A12C 86 95   START1  LDA A   #%10010101 Enable interupt, freq./16 ,
 A12E  B7 DC2A          STA A  MODEM     8 bitar+1 stoppbit, Udda paritet,
(105=1)
A131 86 81           LDA A   #%10000001
A133 B7 E820         STA A   ACIAT      7 bitar+2 stoppbitar, jämn paritet
A136 B6 E821         LDA A   ACIAT+1    DUMMY READ
A139 B6 DC2B         LDA A   MODEM+1    -"-
A13C CE A44C         LDX     #STATXT
A13F BD AD1E         JSR     PSTRNG     Skriv ut menyn
A142 BD AD24         JSR     PCRLF
A145 3E      LOOP    WAI                Vänta på interrupt
A146 20 FD           BRA     LOOP
A148 7E A334 ERR1    JMP     ERROR
             *
             * Sänd en fil från en skiva.
             *
A14B CE A399 DISFIL  LDX     #DISTXT
 A14E  BD  AD1E         JSR     PSTRNG    Skriv:"Namn på diskfil som skall
sändas?"
A151 C6 01           LDA B   #¤01
A153 F7 E820         STA B   ACIAT      INGEN IRQ TANGENTBORD
A156 BD AD24         JSR     PCRLF
A159 BD AD1B         JSR     INBUFF     Hämta in filnamn från terminalen.
A15C C6 81           LDA B   #%10000001
A15E F7 E820         STA B   ACIAT      IRQ IGEN
A161 CE A840         LDX     #SYSFCB
A164 BD AD2D         JSR     GETFIL
A167 25 DF           BCS     ERR1
A169 86 01           LDA A   #1         EXT=TXT
A16B BD AD33         JSR     SETEXT
A16E 86 01           LDA A   #1
A170 A7 00           STA A   0,X
A172 BD B406         JSR     FMS        OPEN FOR READ
A175 26 D1           BNE     ERR1
A177 BD B406 RELOOP  JSR     FMS        GET CHARACTER
A17A 26 7E           BNE     ERR
A17C 7D A112         TST     EFLG
A17F 27 0E           BEQ     LOP
A181 BD AD18         JSR     PUTCHR     UT TILL TERMINAL
A184 81 0D           CMP A   #¤D
A186 26 07           BNE     LOP
A188 86 0A           LDA A   #¤A
A18A BD AD18         JSR     PUTCHR     GE LINEFEED
A18D 86 0D           LDA A   #¤D
A18F F6 DC2A LOP     LDA B   MODEM
A192 C4 02           AND B   #%00000010
A194 27 F9           BEQ     LOP        EJ KLAR
A196 B7 DC2B         STA A   MODEM+1
A199 20 DC           BRA     RELOOP
             *
```

```
                 * Rutin för att spela in mottaget data på skiva.
                 *
A19B CE A49B WRIDIS LDX    #WRITXT    SKRIV TEXT TILL DISK
A19E BD AD1E         JSR    PSTRNG     SKRIV:"Skriver sparad text till disk"
A1A1 C6 01           LDA B  #%0000001
A1A3 F7 E820         STA B  ACIAT      EJ TERMINAL IRQ
A1A6 BD AD1B         JSR    INBUFF
A1A9 C6 81           LDA B  #%10000001
A1AB F7 E820         STA B  ACIAT      TERMINAL IRQ IGEN
A1AE CE A840         LDX    #SYSFCB
A1B1 BD AD2D         JSR    GETFIL
A1B4 25 44           BCS    ERR
A1B6 86 01           LDA A  #1
A1B8 BD AD33         JSR    SETEXT     EXT=TXT
A1BB 86 02           LDA A  #2
A1BD A7 00           STA A  0,X
A1BF BD B406         JSR    FMS        OPEN FOR WRITE
A1C2 26 36           BNE    ERR
A1C4 FF A10C         STX    FCBPEK
A1C7 CE 0000         LDX    #BUFSTA
A1CA BC A115 WLOP    CPX    POINTER    Hela texten på skivan?
A1CD 27 16           BEQ    WSTOP      I så fall avsluta skrivningen.
A1CF A6 00           LDA A  0,X
A1D1 08              INX
A1D2 FF A10E         STX    BUFPEK
A1D5 FE A10C         LDX    FCBPEK
A1D8 BD B406         JSR    FMS        SKRIV TILL DISK
A1DB 26 1D           BNE    ERR
A1DD FF A10C         STX    FCBPEK
A1E0 FE A10E         LDX    BUFPEK
A1E3 20 E5           BRA    WLOP
A1E5 86 04   WSTOP   LDA A  #4
A1E7 FE A10C         LDX    FCBPEK
A1EA A7 00           STA A  0,X
A1EC BD B406         JSR    FMS        CLOSE FILE
A1EF 26 09           BNE    ERR
A1F1 CE 0000         LDX    #BUFSTA
A1F4 FF A115         STX    POINTER    NOLLSTÄLL BUFFER
A1F7 7E A12C         JMP    START1
A1FA 7E A334 ERR     JMP    ERROR
                 *
A1FD 86 03   EXIT    LDA A  #3
A1FF B7 E820         STA A  ACIAT
A202 86 01           LDA A  #¤1
A204 B7 E820         STA A  ACIAT
A207 B7 DC2A         STA A  MODEM
A20A FE A10A         LDX    IRQSAV
A20D FF E000         STX    IRQVEC     ÅTERSTÄLL SYSTEM IRQ
A210 7E AD03         JMP    WARMS
                 *
A213 BD B403 BREAK1  JSR    FMSCLS     STÄNG ALLA FILER
A216 86 FF           LDA A  #¤FF
A218 B7 A117         STA A  SAVFLG     SPARA EJ INKOMMEN TEXT
A21B CE 0000         LDX    #BUFSTA
A21E FF A115         STX    POINTER    ÅTERSTÄLL TEXTPEKAREN
A221 7E A12C         JMP    START1     ÅTERGÅ TILL URSPRUNGSRUTIN
                 *
```

```
                     *       Interruptrutin
                     *
A224 F6 DC2A INTER   LDA B   MODEM      Gav modemet interrupt?
A227 2A 5B           BPL     CHRIN      Om inte,kolla terminalen
A229 B6 DC2B         LDA A   MODEM+1    Ta in tecken från modemet
A22C 84 7F           AND A   #¤7F       Maska bort paritetsbiten
A22E C4 70           AND B   #%01110000
A230 26 23           BNE     REDERR     Om ett läsfel detekterats.
A232 81 1F           CMP A   #¤1F       Kontrolltecken?
A234 22 08           BHI     INT2
A236 81 0A           CMP A   #¤0A
A238 27 04           BEQ     INT2
A23A 81 0D           CMP A   #¤0D
A23C 26 03           BNE     INTE
A23E BD AD18 INT2    JSR     PUTCHR     Skriv tecknet på terminalen
A241 7D A117 INTE    TST     SAVFLG     Skall tecknet sparas?
A244 26 31           BNE     EROUT      Om inte, avsluta
A246 FE A115         LDX     POINTER
A249 8C 9FFF         CPX     #MEMEND    Minnet fullt?
A24C 27 2A           BEQ     WRISTO     I så fall, avbryt
A24E A7 00           STA A   0,X        Lägg tecknet i minnet
A250 08              INX
A251 FF A115         STX     POINTER    Stega fram pekaren
A254 3B              RTI                Nu ligger tecknet i minnet!
                     *
A255 37      REDERR  PSH B
A256 C4 10           AND B   #¤10
A258 27 06           BEQ     ERR2
A25A CE A35A         LDX     #ERTXT1
A25D BD AD1E         JSR     PSTRNG
A260 33      ERR2    PUL B
A261 37              PSH B
A262 C4 20           AND B   #¤20
A264 27 06           BEQ     ERR3
A266 CE A363         LDX     #ERTXT2
A269 BD AD1E         JSR     PSTRNG
A26C 33      ERR3    PUL B
A26D C4 40           AND B   #¤40
A26F 27 06           BEQ     EROUT
A271 CE A36D         LDX     #ERTXT3
A274 BD AD1E         JSR     PSTRNG
A277 3B      EROUT   RTI
                     *
A278 CE A388 WRISTO  LDX     #FULTXT
A27B BD AD1E         JSR     PSTRNG
A27E CE A12C         LDX     #START1
A281 7E A30D         JMP     BRANCH
                     *
                     * Interruptrutin för terminalen.
                     *
A284 F6 E820 CHRIN   LDA B   ACIAT
A287 2A 71           BPL     RTI
A289 B6 E821         LDA A   ACIAT+1    Ta in tecken från terminalen
A28C 84 7F           AND A   #¤7F       Maska bort paritetsbiten
A28E 7F A114         CLR     WAITM
A291 7D A113         TST     FNKFLG     Intern funktion?
A294 26 0F           BNE     COMAND
```

```
A296 B1 A109        CMP A  ESCAPE      Funktionsswitch?
A299 26 3E          BNE    SEND
A29B 73 A113        COM    FNKFLG      Sätt funktionsflaggan.
A29E CE A3CF        LDX    #COMTXT
A2A1 BD AD1E        JSR    PSTRNG      Fråga efter kommando
A2A4 3B             RTI
A2A5 7F A113 COMAND CLR    FNKFLG      Återställ funktionsflaggan
A2A8 B1 A109        CMP A  ESCAPE      Skall ESC. sändas?
A2AB 27 34          BEQ    OUT
A2AD B7 E821        STA A  ACIAT+1
A2B0 BD FAF8        JSR    LFCR
A2B3 84 5F          AND A  #¤5F        Omvandla till stor bokstav
A2B5 B1 A108        CMP A  RETUR       Retur till FLEX?
A2B8 27 41          BEQ    RET         I så fall gå till FLEX
A2BA B1 A103        CMP A  ECHO        Skall tecknen ekas?
A2BD 27 5D          BEQ    EKOSET
A2BF B1 A105        CMP A  SAVE        Skall texten läggas i minnet?
A2C2 27 5D          BEQ    SAVSET
A2C4 B1 A104        CMP A  TRANSM      Skall diskfil sändas?
A2C7 27 37          BEQ    TRAMIT
A2C9 B1 A106        CMP A  WRITE       Skall den inkomna texten sparas på skiva?
A2CC 27 37          BEQ    DISWRI
A2CE B1 A107        CMP A  BREAK       Skall rutinen avbrytas?
A2D1 27 37          BEQ    BRYT
A2D3 86 3F          LDA A  #'?
A2D5 BD AD18        JSR    PUTCHR      Skriv frågetecken
A2D8 3B             RTI
                 *
A2D9 7D A112 SEND   TST    EFLG        EKA?
A2DC 27 03          BEQ    OUT
A2DE BD AD18        JSR    PUTCHR      I så fall, eka till egna terminalen
A2E1 F6 DC2A OUT    LDA B  MODEM
A2E4 C4 02          AND B  #%00000010 Är ACIAN redo?
A2E6 26 0F          BNE    OUT1
A2E8 7C A114        INC    WAITM
A2EB 7D A114        TST    WAITM
A2EE 26 F1          BNE    OUT
A2F0 CE A37B        LDX    #ERTXT4
A2F3 BD AD1E        JSR    PSTRNG      Skriv: * NOT SENT *
A2F6 3B             RTI
A2F7 B7 DC2B OUT1   STA A  MODEM+1     Sänd tecknet
A2FA 3B      RTI    RTI
                 *
                 * Ändra återhoppsadressen till den valda rutinen.
                 *
A2FB CE A1FD RET    LDX    #EXIT
A2FE 20 0D          BRA    BRANCH
A300 CE A14B TRAMIT LDX    #DISFIL
A303 20 08          BRA    BRANCH
A305 CE A19B DISWRI LDX    #WRIDIS
A308 20 03          BRA    BRANCH
A30A CE A213 BRYT   LDX    #BREAK1
A30D FF A110 BRANCH STX    TRADR
A310 30             TSX
A311 B6 A110        LDA A  TRADR
A314 A7 05          STA A  5,X
A316 B6 A111        LDA A  TRADR+1
A319 A7 06          STA A  6,X          NY RETURADRESS
```

```
A31B 3B              RTI               Hoppa till den valda rutinen.
                *
                *
A31C 73 A112 EKOSET  COM     EFLG
A31F 20 D9           BRA     RTI
A321 7F A117 SAVSET  CLR     SAVFLG
A324 CE A477         LDX     #SAVTXT
A327 BD AD1E         JSR     PSTRNG
A32A BD AD24         JSR     PCRLF
A32D CE 0000         LDX     #BUFSTA
A330 FF A115         STX     POINTER    NOLLSTÄLL BUFFER
A333 3B              RTI
                *
A334 A6 01     ERROR LDA A   1,X        ERROR BYTE
A336 81 08           CMP A   #8
A338 26 17           BNE     STOP
A33A 86 04           LDA A   #4
A33C A7 00           STA A   0,X
A33E BD B406         JSR     FMS        CLOSE FILE
A341 A6 01           LDA A   1,X
A343 26 0C           BNE     STOP
A345 CE A3BD         LDX     #REDTXT
A348 BD AD1E         JSR     PSTRNG
A34B BD AD24         JSR     PCRLF
A34E 7E A12C         JMP     START1
A351 BD AD3F STOP    JSR     RPTERR
A354 BD B403         JSR     FMSCLS
A357 7E A12C         JMP     START1
                *
A35A 07        ERTXT1 FCB    ¤07
A35B 2A              FCC     /* RAM */
A362 04              FCB     ¤04
A363 07        ERTXT2 FCB    ¤07
A364 2A              FCC     /* LOST */
A36C 04              FCB     ¤04
A36D 07        ERTXT3 FCB    ¤07
A36E 2A              FCC     /* PAR.ERR. */


A37A 0
A36C 04              FCB     ¤04
A36D 07        ERTXT3 FCB    ¤07
A36E 2A              FCC     /* PAR.ERR. */
A37A 04              FCB     ¤04
A37B 07        ERTXT4 FCB    ¤07
A37C 2A              FCC     /* EJ SÄNT */
A387 04              FCB     ¤04
A388 07        FULTXT FCB    ¤07
A389 4D              FCC     /Minnet fullt !!/
A398 04              FCB     ¤04
A399 07        DISTXT FCB    ¤07
A39A 4E              FCC     /Namn på diskfil som skall sändas ?/
A3BC 04              FCB     ¤04
A3BD 07        REDTXT FCB    ¤07
A3BE 46              FCC     /Filsändning klar/
A3CE 04              FCB     ¤04
A3CF 0A        COMTXT FCB    ¤0A,¤0D
A3D1 54              FCC     /T=Sänd diskfil, S=Spara in-text, W=Text till disk/
```

```
337  A402 0A           FCB    ¤0A,¤0D
338  A404 45           FCC    /E=Ekoswitch,     F=Retur till FLEX, X=Avbryt,
339  A43E 0A           FCB    ¤0A,¤0D           ESC=Sänd ESC./
340  A440 4B           FCC    /KOMMANDO? :/
341  A44B 04           FCB    ¤04
342  A44C 4D   STATXT  FCC    /MODEMET ON LINE!  (ESC)=Kommando följer./
343  A474 0A           FCB    ¤0A,¤0D,¤04
344  A477 07   SAVTXT  FCB    ¤07
345  A478 53           FCC    /Sparar inkommande text i buffer/
346  A497 04           FCB    ¤04
347  A498 07   WRITXT  FCB    ¤07
348  A499 53           FCC    /Skriver sparad text till disk. FILNAMN ? /
349  A4C2 04           FCB    ¤04
350                    END

No error(s) detected
```

*

ANNONSER   ANNONSER   ANNONSER

Till Salu: SWTCP S50 och S30 bussen. CPU 6809 1500:- STAT RAM 64K bestyckat med 48K 1800:- DYN RAM 64K 800:- 16K RAM/PROM 400:- 168K PROM 400:- SERIE 2 ACIA 200:- PRINTER och KLOCKA 150:- FLOPPY Dubbel Density 600:- MODERKORT 16 adr med BAUDRATE 250:- eller allt för 5000:- ALF ERIKSSON VALLAVÄGEN 129 13641 HANDEN 08/7450598 Kretskort för Floppykontroller Dubbel Density till RT Datorn ett fåtal finnes pris 225:- Ring anders 0758/40518 eller Affe 08/7450598

BORTSKÄNKES

FRIDEN PROGRAMATIC FLEXOWRITER TELETYPE MED SVENSKA BOKSTÄVER, PLUS PAPPERS REMSOR OCH MANUAL.

MED VÄNLIG HÄLSNING
PHILIP WOODBERRY
HÄSTHAGSV. 15
123 52 FARSTA
TEL 08-745 93 98

PRINTER SPOOLING

FLEX HAR JU EN FUNKTION KALLAD PRINTER-SPOOLING, MEN FöR ATT DEN SKALL FUNGERA FORDRAS ETT KORT SOM GENERERAR IRQ-SIGNALER VAR TIONDE MILLISEKUND.
FLEX FöRUTSÄTTER ATT ETT 'INTERRUPT TIMER BOARD' FINNS INKOPPLAT. DETTA ÄR VAD JAG FöRSTÅR UPPBYGGT AV EN PIA OCH EN MONOVIPPA.
DÅ JAG REDAN HADE EN TIMER-KRETS 6840 ANVÄNDER JAG DEN ISTÄLLET OCH LYCKADES KNAPA IHOP FöLJANDE PROGRAM, SOM T.EX LADDAS IN GENOM STARTUP-RUTINEN:

```
                                * INTERRUPT HANDLING WITH PTM MC6840

               D3E1   PTMADR   EQU    ¤D3E1       PTM ADDRESS LOCATION
D3A7                           ORG    ¤D3A7
                                * TIMER INITIALIZE ROUTINE

D3A7 BE   D3E1                 LDX    PTMADR      GET PTM ADDRESS
D3AA 86   03                   LDA    #¤03
D3AC A7   01                   STA    1,X
D3AE 86   43                   LDA    #¤43
D3B0 A7   84                   STA    0,X
D3B2 86   28                   LDA    #¤28
D3B4 A7   02                   STA    2,X
D3B6 A7   03                   STA    3,X
D3B8 39                        RTS


                                * TIMER ON ROUTINE

D3BD                           ORG    ¤D3BD
D3BD 86   42                   LDA    #¤42        TURN ON TIMER
D3BF A7   9F D3E1              STA    ÄPTMADRÅ
D3C3 39                        RTS


                                * TIMER OFF ROUTINE

D3C4                           ORG    ¤D3C4
D3C4 86   02                   LDA    #¤02        TURN OFF TIMER
D3C6 A7   9F D3E1              STA    ÄPTMADRÅ
D3CA 39                        RTS


                                * IRQ INTERRUPT HANDLER ROUTINE

D3CB                           ORG    ¤D3CB
D3CB 86   43                   LDA    #¤43        RESET INTERRUPTS
D3CD A7   9F D3E1              STA    ÄPTMADRÅ
D3D1 86   42                   LDA    #¤42
D3D3 A7   9F D3E1              STA    ÄPTMADRÅ
D3D7 7E   C700                 JMP    ¤C700       GO TO SPOOLER
                                * PTM ADDRESS

D3E1                           ORG    PTMADR
D3E1 E040                      FDB    ¤E040       TIMER PTM ADDRESS
                               END
```

KLUBBDISKAR

Klubben har köpt in ett antal program från 68-micro för att kunna distribuera till medlemmarna.

Denna distribution kommer att ske till självkostnadspris, d.v.s. 39:-/diskett.
På detta sätt får beställaren en disk som han/hon kan behålla istället för det gamla krångligare sättet där man först måste sända in en skiva varefter man fick den i retur med programmet på.

Proceduren för programbeställning är alltså:

1. Skriv på ett inbetalningskort : Namn, Adress och medlemsnr.
2. Anteckna vilka diskar du är intresserad av.
3. Betala in 39:- för varje diskett det gäller.
4. Skicka detta inbetalningskort till klubben.

RED.

*

DISK: M6800 NR:1
FORMATTERAD: 20-SEP-84

FIL NR	NAMN	TYP	R	STORL.	SKYDD
1	CHANGE	.CMD		4	
2	CHANGE	.TXT		23	
3	DISK-EXP	.TXT		12	
4	DISKFIX	.CMD		8	
5	DISKFIX1	.TXT		37	
6	DISKFIX2	.TXT		40	
7	DISKSAVE	.CMD		7	
8	DISKSAVE	.TXT		44	
9	FINDDAT	.CMD		1	
10	FINDDAT	.TXT		7	
11	INFO-CNG	.TXT		4	
12	PRMOD	.TXT		23	
13	TESTDISK	.CMD		6	
14	TESTDISK	.TXT		35	

FILER=14, SEKTORER=251,
STÖRST=44, LEDIGA=139

* *

DISK: BASIC NR:1
FORMATTERAD: 20-SEP-84

FIL NR	NAMN	TYP	R	STORL.	SKYDD
1	LIFETIME	.BAS		39	
2	POET	.BAS		7	
3	DIET	.BAS		19	
4	FOODLIST	.TXT		10	
5	SNOOPY2	.BAS		9	
6	FOOTBALL	.BAS		25	
7	HEXPAWN	.BAS		20	
8	SPACEWAR	.BAS		58	
9	COVLET	.BAS		10	
10	ENVELOPE	.BAS		8	
11	SEARCVLT	.BAS		13	
12	SEARENVL	.BAS		12	
13	SREXAMZP	.BAS		8	
14	SRCVLTZP	.BAS		11	
15	SRENVLZP	.BAS		10	
16	DATCOMP	.BAS		5	
17	MAILING	.TXT		12	
18	LETTER	.BAS		4	
19	LOVESIGN	.BAS		6	
20	BOWLING	.BAS		13	
21	BLACKJAK	.BAS		26	

FILER=21, SEKTORER=325,
STÖRST=58, LEDIGA=65

```
DISK: BASIC  NR:2
FORMATTERAD: 20-SEP-84

FIL NR  NAMN      TYP R STORL.   SKYDD


     1  INFO-PO .TXT      18
     2  PO      .TXT      37
     3  PO      .BAC      24
     4  ADDRESS .DAT       4
     5  SIX27111.DAT       1
     6  ALP27111.DAT       1

    FILER=6, SEKTORER=85,
    STöRST=37, LEDIGA=305
```

*

```
DISK: KLUBBDAT  NR:1
FORMATTERAD: 20-SEP-84

FIL NR  NAMN      TYP R STORL.  SKYDD


     1  CURPOS  .V16      16
     2  CURPOS  .V19      17
     3  CURPOS  .XPC      44
     4  HEXDEC  .TXT       8
     5  PACK    .INF       6
     6  PACK    .TXT      49
     7  REPEAT  .TXT       6
     8  TEST    .CUR      19

    FILER=8, SEKTORER=165,
    STöRST=49, LEDIGA=225
```

*

```
DISK: STANDARD  NR:120
FORMATTERAD: 26-MAJ-84

FIL NR  NAMN      TYP R STORL.  SKYDD


     2  COPY    .CMD      15
     3  COPY    .DOC      69
     4  COPY    .TXT     114
     5  CSSTABS .TXT      15
     6  MEMEX   .TXT      32
     7  P       .TXT       7
     8  STANDARD.TXT      65

    FILER=7, SEKTORER=317,
    STöRST=114, LEDIGA=23
```

```
DISK: M6809-1  NR:1
FORMATTERAD: 20-SEP-84

FIL NR  NAMN      TYP R STORL. _ SKYDD


     1  FILESORT.CMD       9
     2  FILESORT.TXT     159
     3  MINICAT .TXT      10
     4  MINICOPY.TXT      19
     5  MINIFMS .TXT      44
     6  MINIFMS .BIN       9
     7  DISKEDIT.TXT      85
     8  DISKEDIT.CMD      12
     9  PRIME   .TXT      15
    10  DISKEDIT.REP       4
    11  INSTRUC .TXT      20

    FILER=11, SEKTORER=386,
    STöRST=159, LEDIGA=4
```

*

```
DISK: M6809-2  NR:2
FORMATTERAD: 20-SEP-84

FIL NR  NAMN      TYP R STORL. SKYDD


     1  CBUG09  .TXT      18
     2  SEC1    .TXT      76
     3  SEC2    .TXT      90
     4  FIND    .TXT       8
     5  TABLE2  .TXT       5
     6  INTEXT  .CMD       3
     7  INTEXT  .TXT      20
     8  DISK-EXP.TXT      12
     9  INDEX   .CMD       6
    10  INDEX   .TXT      33
    11  SORTSPEC.BIN       2
    12  CRTSET  .CMD       7
    13  CRTSET  .SRC      32
    14  CRTSET  .PR        7
    15  CRTSET  .TXT       8
    16  READ_ME .TXT       2
    17  ASMFLEX .LIB       2
    18  DATECOPY.TXT       5
    19  DATECOPY.LTR       4

    FILER=19, SEKTORER=340,
    STöRST=90, LEDIGA=50
```

*
*

DISK: M6809-3 NR:3
FORMATTERAD: 20-SEP-84

FIL NR	NAMN	TYP	R	STORL.	SKYDD
1	LOADER	.ASM		4	
2	FDUMP	.CMD		1	
3	RLOAD	.CMD		18	
4	MODULE	.MAC		12	
5	ABSPRO	.REL		2	
6	ASCHEX	.REL		1	
7	COPY	.REL		1	
8	DOCMD	.REL		2	
9	ENTER	.REL		1	
10	ENTPRO	.REL		3	
11	EXCHAN	.REL		1	
12	EXTPRO	.REL		2	
13	EXT2	.REL		2	
14	GETNAM	.REL		2	
15	GETTOK	.REL		2	
16	MODPRO	.REL		2	
17	NTERPS	.REL		3	
18	OSLINK	.REL		3	
19	PASS1	.REL		3	
20	PASS2	.REL		4	
21	PSTRNG	.REL		1	
22	RDBNRC	.REL		1	
23	RLOAD	.REL		2	
24	SCOMPR	.REL		1	
25	SEARCH	.REL		1	
26	SORT	.REL		2	
27	SYMTAB	.REL		1	
28	WRBNRC	.REL		1	
29	ZCOPY	.REL		1	
30	ABSPRO	.TXT		9	
31	ASCHEX	.TXT		5	
32	COPY	.TXT		3	
33	DOCMD	.TXT		10	
34	ENTER	.TXT		6	
35	ENTPRO	.TXT		13	
36	EXCHAN	.TXT		4	
37	EXTPRO	.TXT		6	
38	EXT2	.TXT		8	
39	FCBDEF	.TXT		2	
40	FDUMP	.TXT		3	
41	FLEXDEF	.TXT		4	
42	FORMAT	.TXT		1	
43	GETTOK	.TXT		10	
44	LDRDEF	.TXT		4	
45	LOADFILE	.TXT		2	
46	MODPRO	.TXT		8	
47	NTERPS	.TXT		16	
48	OSLINK	.TXT		12	
49	PASS1	.TXT		18	
50	PASS2	.TXT		19	
51	PSTRNG	.TXT		3	
52	RDBNRC	.TXT		6	
53	RLOAD	.TXT		8	
54	SCOMPR	.TXT		4	
55	SEARCH	.TXT		4	
56	SORT	.TXT		17	
57	SYMTAB	.TXT		2	
58	WRBNRC	.TXT		5	
59	ZCOPY	.TXT		4	
60	DISKFIX9	.CMD		7	
61	DISKFIX9	.LTR		9	
62	DISKFIX9	.TXT		48	

FILER=62, SEKTORER=360,
STÖRST=48, LEDIGA=29

****

*** ANNONS – SÄLJES ***

1 ST FLOPPY BASF 8101 40 SPAR SS 800:-
1 ST TERMINAL CA-790 INKL. LADA, TANGENTBORD OCH KONTAKTER 800:-
1 ST 28 K EXTRA MINNESKORT VIC-20 250:-
1 ST BOK VIC REVEALD VIC20 260 S ENG 100:-
1 ST BOK VIC20 BYGG UT DIN VIC 90 S SVENSKA 50:-
1 ST BOK VIC20 PROGRAMMERINGSHANDBOK 160 S SVENSKA 50:-
1 ST BOK VIC64 PROGRAMMERS REFERENCE MANUAL 480 S ENG 100:-

MARTIN ERLANDSON
ANDERSPRÄSTGATAN 1
442 36 KUNGÄLV
0303-181 29 KVÄLLAR

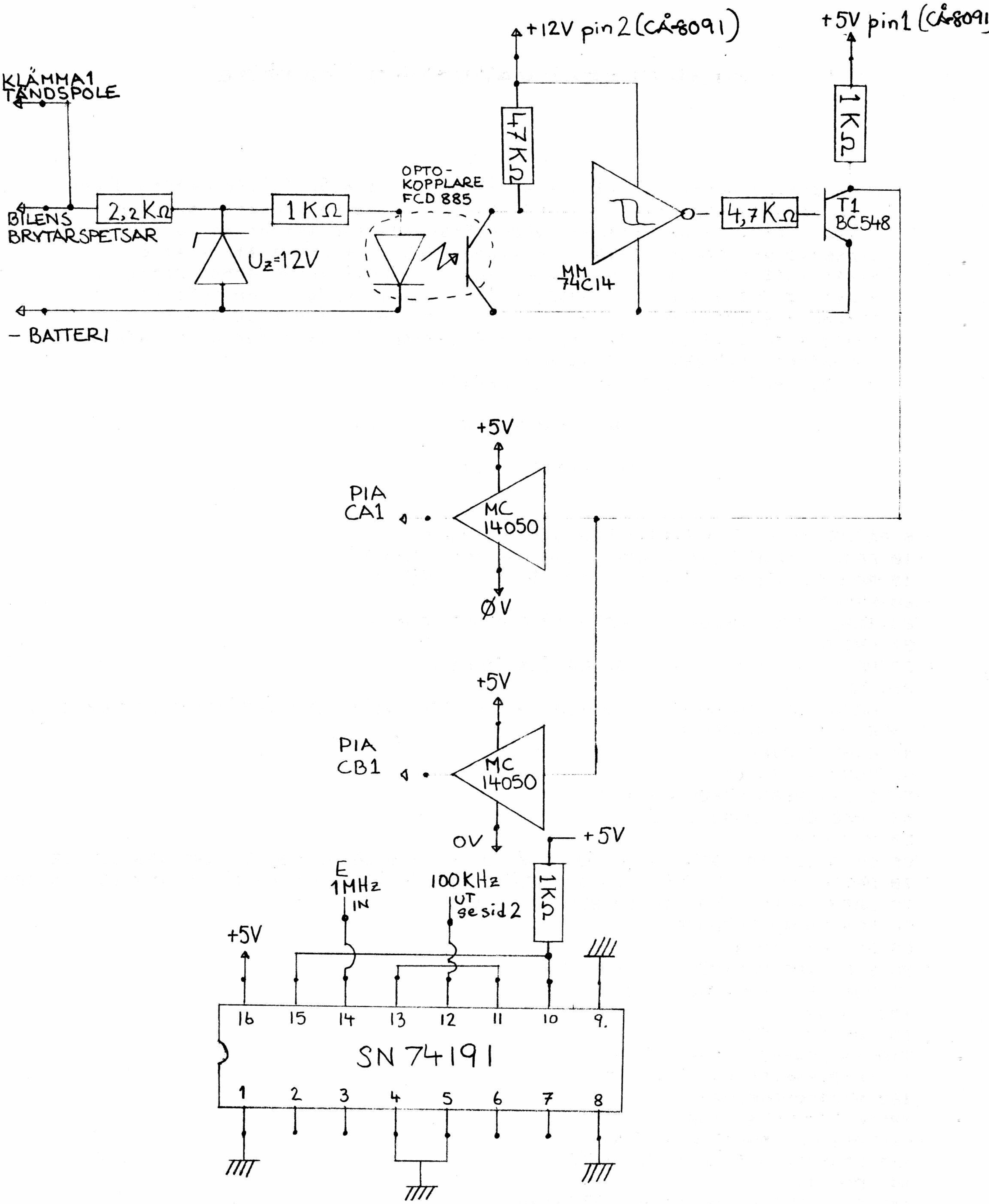
KAMVINKEL/VARVTALS-MÄTARE

En optokopplares lysdiodsida anslutes,via några skyddsmotstånd och en zenerdiod,paralellt med brytarspetsarna.En schmittrigger ansluts till optokopplarens transistorsida för att ge en ren fyrkantsvåg på schmittriggerns utgång.Transistorn t1 återställer nivån till 5 volt innan signalen,via 2 buffrar(MC14050),matas in på pians CA1 och CB1 ingångar.Datorns klocksignal omvandlas till 100khz i en 74191 krets och matas sedan in på 4 kaskadkopplade 74191 kretsar.Dessa kretsars räkneutgångar matas sedan in på pians PA0-PA7 resp PB0-PB7 ingångar
Jag bifogar kretsscheman och program

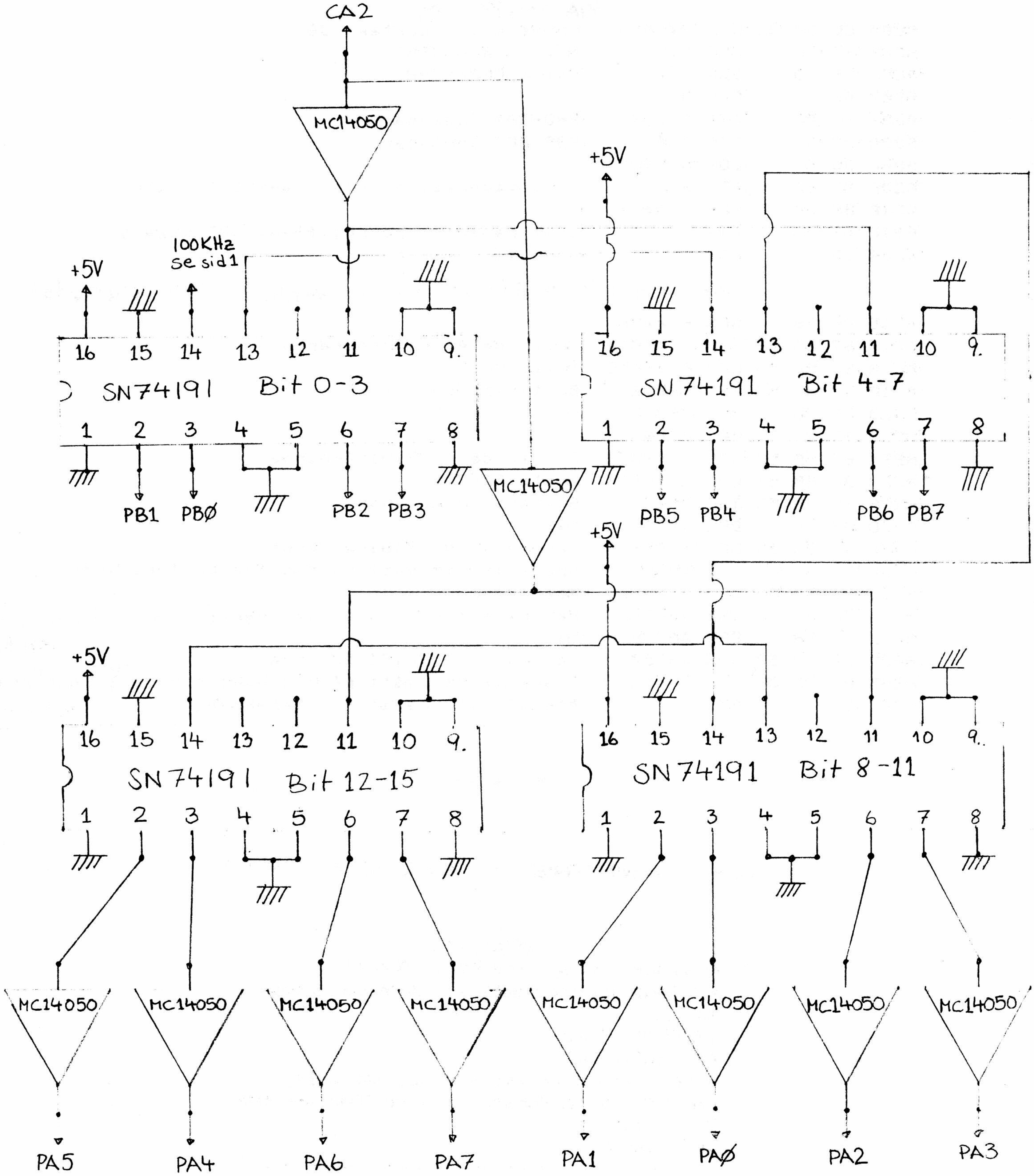
Med PD68 hälsningar från

Nils-Gösta Elfström

```
5 PRINT CHR$(27);CHR$(21);CHR$(27);CHR$(5);
10 PRINT "MED DETTA PROGRAM KAN MAN MÄTA KAMVINKEL"
15 PRINT "ELLER VARVTAL PÅ EN 4TAKT OTTOMOTOR"
20 PRINT
25 INPUT "HUR MÅNGA CYLINDRAR HAR DIN MOTOR";A
27 PRINT
28 PRINT "TRYCK PÅ GODTYCKLIG GRÅ TANGENT"
29 INPUT A$
30 DATA 206,128,80,111,2,111,3,79,167,0,167,1,134,54,167,2,134,52,167,3,57
35 REM PIA INITIERING
40 B=HEX("A1FF")
45 FOR X=1 TO 21
50 B=B+1:READ C:POKE B,C:NEXT X
55 POKE 68,162:POKE 69,0
60 Y=USR(0)
65 DATA 134,54,183,128,82,182,128,80,125,128,82,42,251,134,62,183,128,82
70 DATA 182,128,81,125,128,83,42,251,254,128,80,255,163,0,182,128,80,125,128,82
75 DATA 42,251,254,128,80,255,163,2,57
80 REM HÄMTA MÄTDATA
85 FOR X=1 TO 47
90 B=B+1:READ C:POKE B,C:NEXT X
95 POKE 68,162:POKE 69,21
100 Y=USR(0)
105 D=HEX("A300")
110 E=256*PEEK(D)+PEEK(D+1)
115 F=256*PEEK(D+2)+PEEK(D+3)
120 V=200000*60/A*F
125 K=360*(F-E)/A*F
130 PRINT CHR$(27);CHR$(5);
135 PRINT "VARVTALET=";V;"R/M"
140 PRINT
145 PRINT "KAMVINKELN=";K;"GRADER"
150 PRINT
155 PRINT "ÖNSKAS NYA MÄTVÄRDEN.SVARA JA ELLER NEJ"
160 INPUT A$
165 IF A$="JA" THEN 95
170 END
```



LABKORT FÖR VARVTAL OCH KAMVINKEL = CÅ-8091



PIANS stift 23 ansluts till J6 stift 21
—— " —— 36 —— " —— 16
—— " —— 35 —— " —— 15
för övrigt se sidan 77 i Bygg själv datorer.

Subrutiner till kamvinkel varvtalsprogram

PIA initiering

```
A200 CE 80 50 LDX #$8050   Kretskort i kontakt J6
A203 6F 02    CLR 2,X      Nolla CRA0-CRA7
A205 6F 03    CLR 3,X      Nolla CRB0-CRB7
A207 4F       CLR A
A208 A7 00    STA A 0,X    PA0-PA7 ingångar
A20A A7 01    STA A 1,X    PB0-PB7 ingångar
A20C 86 36    LDA A #$36
A20E A7 02    STA A 2,X    CRA7 påverkas på pos.flank; CA2 utgång
A210 86 34    LDA A #$34
A212 A7 03    STA A 3,X    CRB7 påverkas på neg.flank; CB2 utgång
A214 39       RTS          Åter till basic
```

subrutin för mätning av kamvinkel eller varvtal

```
A215 86 36    LDA A #$36
A217 B7 80 52 STA A $8052  Nolla de 4 74191räknarna
A21A B6 80 50 LDA A $8050  Nolla CRA7
A21D 7D 80 52 TST $8052    Brytarspetsar isär?
A220 2A FB    BPL $A21D    Nix
A222 86 3E    LDA A #$3E
A224 B7 80 52 STA A $8052  Starta de 4 74191räknarna
*A227 B6 80 51 LDA A $8051 Nolla CRB7
A22A 7D 80 53 TST $8053    Brytarspetsar ihop?
A22D 2A FB    BPL $A22A    Nix
A22F FE 80 50 LDX $8050    Läs av de 4 74191räknarna
A232 FF A3 00 STX $A300    Spara ix=ett mått på tid för brytare isär
A235 B6 80 50 LDA A $8050  Nolla CRA7
A238 7D 80 52 TST $8050    Har kamaxeln roterat 1/4 varv (1/6 varv för 6 cylindrig motor
A23B 2A FB    BPL $A238    Nix
A23D FE 80 50 LDX $8050    Läs av de 4 74191räknarna
A240 FF A3 02 STX $A302    Spara ix=ett mått på 1/4 kamaxelvarv (1/6 varv för 6 cylindrig motor
A243 39       RTS          Återgå till basic för beräkning
```

*****

*** ANNONS – SÄLJES ***

MODEMKORT

Modemkort uppbygt kring AM7910 (AM7911)
Fristående kort med autosvar med A och B kanal
300 Baud
1200/75(150) Baud
(150)75/1200 Baud
Passar vilken dator som helst med V24 utgång
Kretskort med telefonanpassad trafo 300kr per styck
Ian Birchman
Tel 08/205719

DISKDRIVERS

1st ny 40spår BMI dubbelsidig drive
1st ny 80spår dubbelsidig SHUGART drive
Ian Birchman
tel 08/205719

DIREKTRONIK

BRA PRISER!

Kvalitetsdon
från H & T Components,
direkt utbytbara mot de flesta
kontaktdonen på marknaden (MIL C 83503).

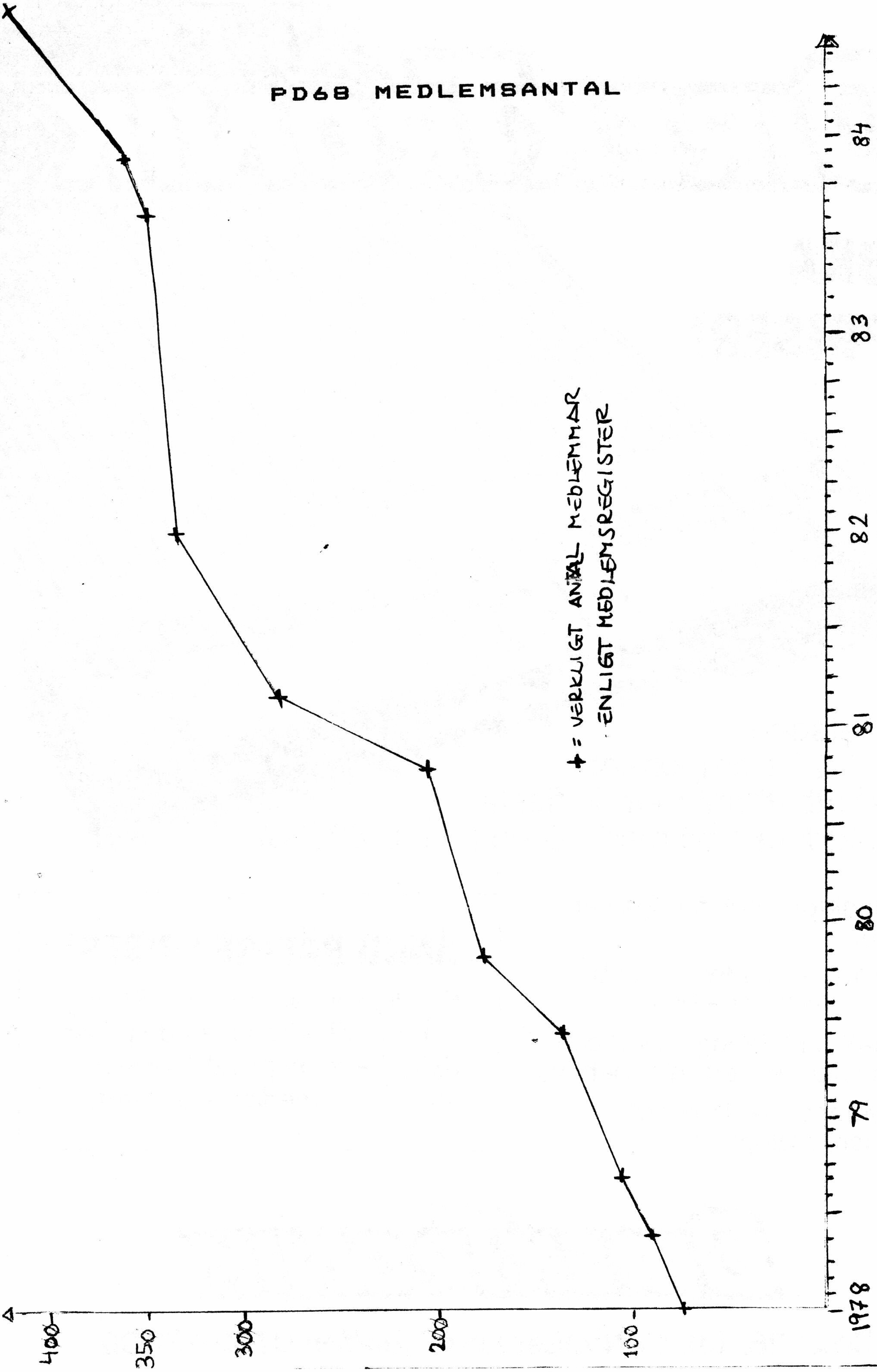
- Kontaktdon och färdiga kablage.
- Socket – Header – D-sub – DIP – Kortkant – PCB.
- Selektiv plätering – guld och tenn där det verkligen behövs.
- Leverans "på direkten" – begär katalog!

ÄNNU BÄTTRE PRISER!

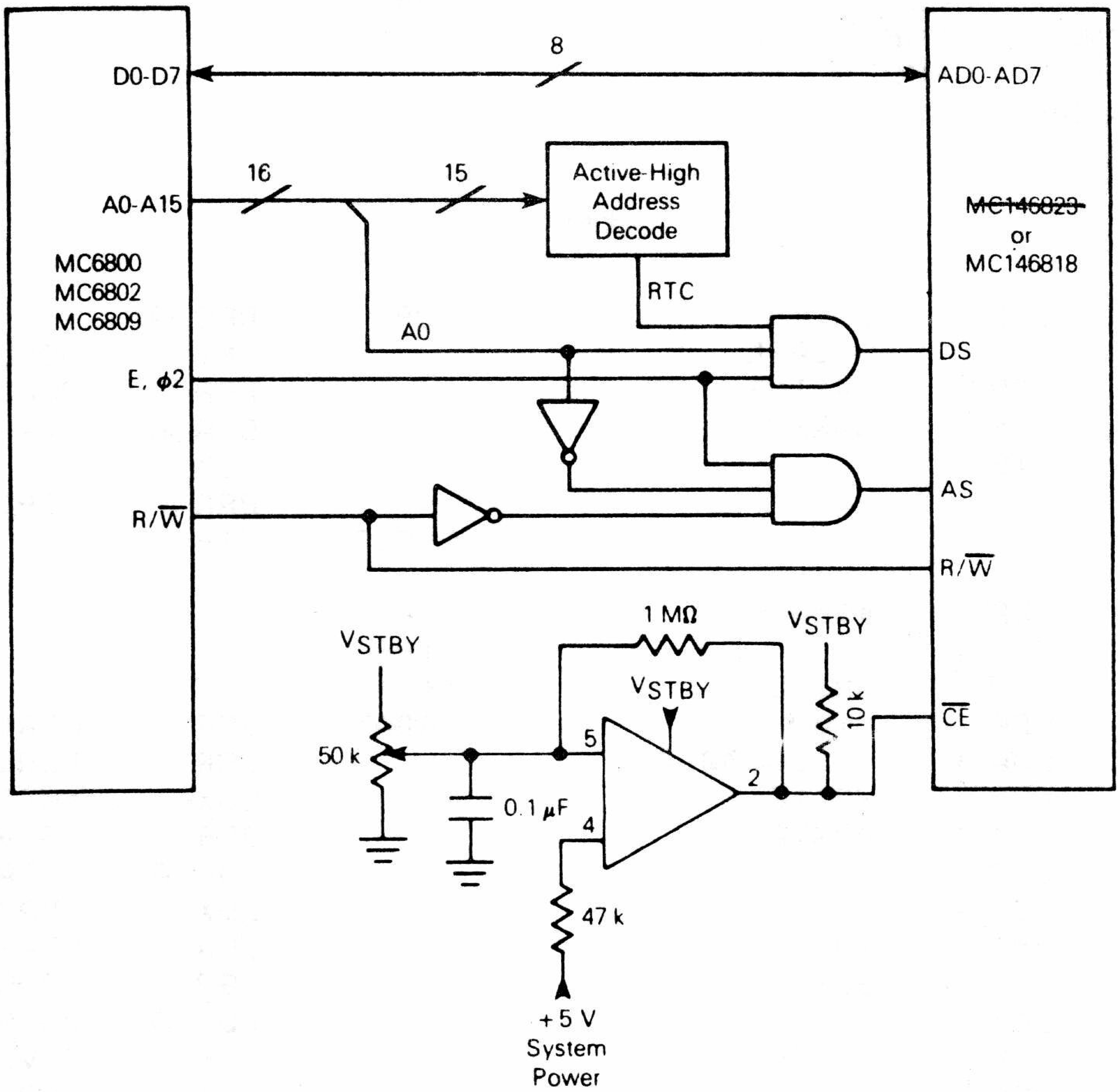
För dig som är medlem i PD 68 har vi kraftigt reducerade priser. (Gäller endast privatpersoner som finns i medlemsregistret)

DIREKTRONIK

Box 126, 149 01 Nynäshamn. Telefon 0752-182 00.


PD68 MEDLEMSANTAL
+ = VERKLIGT ANTAL MEDLEMMAR
ENLIGT MEDLEMSREGISTER
1978
79
80
81
82
83
84
100
200
300
350
400

REALTIDSKLOCKA FöR 6800/6809



****

```
RTC      EQU    $DC40
OUTHEX   EQU    $AD3C
WARMS    EQU    $AD03
PUTCHR   EQU    $AD18
PCRLF    EQU    $AD24
PSTRNG   EQU    $AD1E

         ORG    $E200
STAR     BRA    TIME
         BRA    INT1

****
****
****
DATA     FCB    01          SECONDS
         FCB    01          SECONDS ALARM
         FCB    01          MINUTES
         FCB    01          MINUTES ALARM
         FCB    01          HOUR
```

```
        FCB     01              HOUR ALARM
        FCB     01              DAY OF WEEK
        FCB     01              DAY OF MONTHS
        FCB     01              MONTHS
        FCB     01              YEAR
TEMP1   RMB     1
TEMP2   RMB     1
TADDR   RMB     1
```

****

```
****
INT1    JSR     INTCK
TIME    JSR     PRDATE
        JSR     PRTIME
        JMP     WARMS
*       RTS

****
****
INTCK   LDA A   #$0A
        CLR B
        BSR     WRITE
        LDA A   #$0B
        LDA B   #%10000010
        BSR     WRITE
        LDX     #DATA
        DEC A
LOP1    DEC A
        BMI     INTUT
        LDA B   9,X
        DEX
        BSR     WRITE
        BRA     LOP1
INTUT   LDA A   #$0B
        LDA B   #%00000010
        BSR     WRITE
        RTS

****

****
READ    STA A   RTC
        LDA B   RTC+1
        RTS

WRITE   STA A   RTC
        STA B   RTC+1
        RTS

****
****
GDATA   LDX     #DATA
CHECH   LDA A   #$0A
        STA A   RTC
        LDA B   RTC+1
        BMI     CHECH
```

```
LOP2    DEC A
        BMI     GDOUT
        BSR     READ
        STA B   9,X
        DEX
        BRA     LOP2
GDOUT   RTS

****
****
****
PRDATE  JSR     GDATA
        JSR     PCRLF
        LDX     #MESS2
        JSR     PSTRNG
        LDA A   DATA+6
        LDX     #DAYS
        JSR     START
        JSR     SPACE
        LDA A   DATA+8
        JSR     COVBCD
        LDX     #MONTHS
        JSR     START
        JSR     SPACE
        LDX     #DATA+7
        JSR     OUTHEX
        LDX     #YEAR
        JSR     PSTRNG
        RTS

****
****
****

PRTIME  JSR     PCRLF
        LDX     #MESS1
        JSR     PSTRNG
        LDX     #DATA+4
        JSR     OUTHEX
        JSR     COLLON
        LDX     #DATA+2
        JSR     OUTHEX
        JSR     COLLON
        LDX     #DATA
        JSR     OUTHEX
```

```
        JSR     PCRLF
        RTS

SPACE   PSH A
        LDA A   #$20
        JSR     PUTCHR
        PUL A
        RTS

COLLON  PSH A
        LDA A   #$3A
        JSR     PUTCHR
        PUL A
        RTS

COVBCD  PSH B
        TAB
        AND B   #$0F
        AND A   #$F0
        BEQ     OUT1
CONV1   ADD B   #10
        SUB A   #$10
        BNE     CONV1
OUT1    TBA
        PUL B
        RTS

***  PRINT TEXT DATE
***
START   TAB
SCHT1   DECB                 SEARCH
        BEQ     SCH2    TABLE FOR TEXT
SCH1    INX
        LDA A   0,X
        BNE     SCH1
        BRA     SCHT1
SCH2    INX
        LDA A   0,X          LOAD
        BEQ     SCH3     CHARACTER
        JSR     PUTCHR       SEND
        BRA     SCH2     TO TERMINAL
SCH3    RTS
```

```
MONTHS  FCC     ' JAN'
        FCB     0
        FCC     'FEB'
        FCB     0
        FCC     'MAR'
        FCB     0
        FCC     'APR'
        FCB     0
        FCC     'MAY'
        FCB     0
        FCC     'JUN'
        FCB     0
        FCC     'JUL'
        FCB     0
        FCC     'AUG'
        FCB     0

        FCC     'SEP'
        FCB     0
        FCC     'OCT'
        FCB     0
        FCC     'NOV'
        FCB     0
        FCC     'DEC'
        FCB     0
DAYS    FCC     ' SUNDAY'
        FCB     0
        FCC     'MONDAY'
        FCB     0
        FCC     'TUESDAY'
        FCB     0
        FCC     'WEDNESDAY'
        FCB     0
        FCC     'THURSDAY'
        FCB     0
        FCC     'FRIDAY'
        FCB     0
        FCC     'SATURDAY'
        FCB     0
YEAR    FCC     ' 1984'
        FCB     4
MESS1   FCC     ' TIME '
        FCB     4
MESS2   FCC     ' DATE '
        FCB     4
        END     STAR
```

****

*** MERA ANNONSER ***

RT-DATOR med följande :

CPU-kort m. 6800/6809
Minneskort 48k dynamiskt
ACIA-kort
Lab-kort
Timerkort
Alltsammans inbyggt i 19" rack med kraftig nätdel.
Pris : 2900:-

Jan Nordström
Tel : 0753 - 44451 kvällar

Några få böcker av AN ASSEMBLY LANGUAGE COURSE återstår. Boken finns beskriven i tidigare nr av MPU-laren. Pris 195:-

Ingemar Skarpås
Tel : 08 - 969696

KöPES

1 st diskdrive

Bengt Garell
Tel : 0498 - 19383 dagtid

****

KOMMUNIKATIONSPROGRAM FöR 68000-PROJEKTET

För att tillgodose våra ivriga 68000-byggare har vi härmed nöjet att presentera program nr. 1.
Programmet är det välkända kommunikationsprogrammet KERMIT.
Naturligtvis kommer 68000-byggarna att skicka in massor med program till MPU-laren (eller ??) så att vi får en stadig avdelning i tidningen med program, byggtips m.m. m.m.
Lycka till önskar RED.

UNIX KERMIT

A sample, working implementation of the Kermit "kernel" was written in the C language, and widely distributed in the Kermit Protocol Manual. This kernel was intended merely to illustrate the protocol, and did not include a "user interface", nor some of the fancy features like server support, 8-bit quoting, file warning, timeouts, etc. Several sites have added the necessary trappings to make this a production version of Kermit, usually under the UNIX operating system.

The COMND style of user/program interaction favored by Kermit (program types prompt, user types command followed by operands, program types another prompt, etc) is contrary to the UNIX style, so UNIX implementations have a style more familiar to UNIX users. C versions of Kermit are running successfully on VAX and PDP-11 UNIX systems, and the SUN Microsystems MC68000-based workstation.

UNIX filespecs are of the form

```
dir1/dir2/dir3/ ... /filename
```

where the tokens delimited by slashes form a path name, and by convention are each limited to 14 characters in length. The final token in a path is the actual file name. By convention, it is of the form name.type, but there is nothing special about the dot separating name and type; to UNIX it's just another character, and there may be many dots in a filename.

Columbia UNIX Kermit

The version in use at Columbia was adapted from the Kermit kernel by Chris Maio and Bob Cattani of the Columbia University Computer Science Department for the SUN and VAX under UNIX, and is described below by Chris Maio. It is a version 1 Kermit (no server functions).

This version more or less conforms to the Unix command parsing conventions, and has the ability to time out.

The new command syntax isn't as user friendly as the that of the other implementations, but it is closer to what most Unix users will expect. Roughly, this is how it works: You can only do one thing with any one invocation of Kermit; either connect, send, or receive. The format is basically like that of the "dump" or "tar" tape utilities, in that the first command line argument is a word made up of letters which specify the desired function and options. This "word" is then followed by the arguments for the options, in the order in which the options appear in the command "word," and then a list of files to be sent, if the send command is being invoked. The commands are:

```
c    connect
s    send
r    receive
```

and the options are:

```
d   debug mode (no argument)
i   image (8-bit) mode
    No mapping from LF to CRLF, and all eight bits of each byte
    are sent or received. if this option is not specified, LF
    is mapped to CRLF, and only 7 bit bytes are sent or
    received.

l   line (the next arg is a tty name, e.g. /dev/tty13)
b   baud rate (the next arg is a decimal number, e.g. 1800)
e   escape char
    (next arg is a decimal number, interpreted as the ascii
    code for the desired escape character)
```

The defaults are no debug mode, no external tty line (i.e. "host" mode -- remote -- operation), the system default baud rate, and ÜD as the escape character. For example, assume that a user on a micro running Unix wants to transfer files between her machine and a DEC-20 over a hardwire line, say, /dev/tty13, at 4800 baud, using ÜB (ASCII code 2) as her escape character. Note that the order in which arguments are given must match the order in which corresponding letters appear in the first "word":

```
% kermit clbe /dev/tty13 4800 2
Kermit: connected
É
É; now we're on the -20
Ékermit
Kermit-20>send *.c
                                  (user types ÜB to get back to Unix)
Kermit: disconnected
%
% kermit rlb /dev/tty13 4800    (no escape character needed here)
Receiving CONNECT.C
Receiving MAIN.C
Receiving RECEIVE.C
Receiving SEND.C
Receiving UTILS.C
OK

%                                 (now, let's send to the DEC-20)
% kermit clbe /dev/tty13 4800 2
Kermit: connected.

                                          (now we're on the -20)
Kermit-20>receive
ÜB                                (user types ÜB again to get back)
Kermit: disconnected.
%
% kermit slb /dev/tty13 4800 *.pl
Sending learn.pl
Sending parse.pl
OK
%
```

While the format shown above might seem verbose, it's easy to set up shell scripts or command macros so that, for a given tty line, all one need type is "connect", "send file1 file2 ..." or "receive."

"Host" (remote) mode operation is much simpler. When running Kermit in host mode, nothing about the tty line need be specified, so the format reduces to

```
% kermit r
```

to receive files sent by the "remote" Kermit, and

```
% kermit s file1 file2 file3 ...
```

to send files back to your local system. Unfortunately, this command syntax doesn't allow you to own the tty line from the local machine to the "host" continuously, but this shouldn't be a problem for most users.

The current version has been used to transfer files between VAXes running Berkeley 4.1bsd Unix, a Sun workstation running Unisoft Unix (basically Unix Version 7 with Berkeley extensions), and a DEC-20, over dialups at 1200 baud, the PACX at 4800 baud, and hardwire lines at 9600 baud, all with no noticable problems. None of the special features of Berkeley Unix are used, so only minimal, straightforward changes should be necessary to get Kermit up and running on other flavors of Unix.

Unix Kermit presently does not send or process error-message packets.

***

OBS !!! SISTA DATUM FöR MATERIEL TILL NÄSTA NUMMER AV MPU-LAREN ÄR REDAN 850220

***

The following program uses decimal notation for numbers, and tochar() rather than char() for integer-to-character conversion.

```
/*
 *  K e r m i t   File Transfer Utility
 *
 *  UNIX Kermit, Columbia University, 1981, 1982, 1983
 *      Bill Catchings, Bob Cattani, Chris Maio, Frank da Cruz, Alan Crosswell
 *
 *  Also:    Jim Guyton, Rand Corporation
 *           Walter Underwood, Ford Aerospace
 *
 *  usage:   kermit c Älbe line baud escapecharÅ        to connect
 *           kermit s Äd..iflb line baudÅ file ...      to send files
 *           kermit r Äd..iflb line baudÅ               to receive files
 *
 *  where    c=connect, s=send, r=receive,
 *           d=debug, i=image mode, f=no filename conversion, l=tty line,
 *           b=baud rate, e=escape char.
 *
 *  For remote Kermit, format is either:
 *           kermit r                                   to receive files
 *  or       kermit s file ...                          to send files
 *
 *
 *  Modification History:
 *
 *  Oct. 17 Included fixes from Alan Crosswell (CUCCA) for IBM_UTS:
 *           - Changed MYEOL character from Ön to Ör.
 *           - Change char to int in bufill so getc would return -1 on
 *             EOF instead of 255 (-1 truncated to 8 bits)
 *           - Added read() in rpack to eat the EOL character
 *           - Added fflush() call in printmsg to force the output
 *           NOTE: The last three changes are not conditionally compiled
 *                 since they should work equally well on any system.
 *
 *           Changed Berkeley 4.x conditional compilation flag from
 *               UNIX4X to UCB4X.
 *           Added support for error packets and cleaned up the printing
 *               routines.
 */

#include <stdio.h>              /* Standard UNIX definitions */

/* Conditional compilation for different machines/operating systems */
/* One and only one of the following lines should be 1 */

#define UCB4X       1           /* Berkeley 4.x UNIX */
#define TOPS_20     0           /* TOPS-20 */
#define IBM_UTS     0           /* Amdahl UTS on IBM systems */
#define VAX_VMS     0           /* VAX/VMS (not yet implemented) */

/* Conditional compilation for the different Unix variants */
/* 0 means don't compile it, nonzero means do */

#if UCB4X
```

```
#define V6_LIBS      0          /* Dont't use retrofit libraries */
#define NO_FIONREAD  0          /* We have ioctl(FIONREAD,...) for flushinput() */
#define NO_TANDEM    0          /* We have TANDEM line discipline (xon/xoff) */
#endif

#if IBM_UTS
#define V6_LIBS      0          /* Don't use retrofit libraries */
#define NO_FIONREAD  1          /* No ioctl(FIONREAD,...) for flushinput() */
#define NO_TANDEM    1          /* No TANDEM line discipline (xon/xoff) */
#endif

#if V6_LIBS
#include <retrofit/sgtty.h>
#include <retrofit/signal.h>
#include <retrofit/setjmp.h>
#else
#include <sgtty.h>
#include <signal.h>
#include <setjmp.h>
#endif
#if NO_TANDEM
#define TANDEM       0          /* define it to be nothing if it's unsupported */
#endif


/* Symbol Definitions */

#define MAXPACKSIZ  94          /* Maximum packet size */
#define SOH         1           /* Start of header */
#define CR          13          /* ASCII Carriage Return */
#define SP          32          /* ASCII space */
#define DEL         127         /* Delete (rubout) */
#define ESCCHR      'Ü'         /* Default escape character for CONNECT */

#define MAXTRY      10          /* Times to retry a packet */
#define MYQUOTE     '#'         /* Quote character I will use */
#define MYPAD       0           /* Number of padding characters I will need */
#define MYPCHAR     0           /* Padding character I need (NULL) */

#if IBM_UTS
#define MYEOL       'Ör'        /* End-Of-Line character for UTS systems */
#else
#define MYEOL       'Ön'        /* End-Of-Line character I need */
#endif

#define MYTIME      10          /* Seconds after which I should be timed out */
#define MAXTIM      60          /* Maximum timeout interval */
#define MINTIM      2           /* Minumum timeout interval */

#define TRUE        -1          /* Boolean constants */
#define FALSE       0


/* Macro Definitions */

/*
```

```
 * tochar: converts a control character to a printable one by adding a space.
 *
 * unchar: undoes tochar.
 *
 * ctl:    converts between control characters and printable characters by
 *         toggling the control bit (ie. ÜA becomes A and A becomes ÜA).
 */
#define tochar(ch)  ((ch) + ' ')
#define unchar(ch)  ((ch) - ' ')
#define ctl(ch)     ((ch) Ü 64 )


/* Global Variables */
int     size,                   /* Size of present data */
        rpsiz,                  /* Maximum receive packet size */
        spsiz,                  /* Maximum send packet size */
        pad,                    /* How much padding to send */
        timint,                 /* Timeout for foreign host on sends */
        n,                      /* Packet number */
        numtry,                 /* Times this packet retried */
        oldtry,                 /* Times previous packet retried */
        ttyfd,                  /* File descriptor of tty for I/O, 0 if remote */
        remote,                 /* -1 means we're a remote kermit */
        image,                  /* -1 means 8-bit mode */
        debug,                  /* indicates level of debugging output (0=none) */
        filnamcnv,              /* -1 means do file name case conversions */
        filecount;              /* Number of files left to send */

char    state,                  /* Present state of the automaton */
        padchar,                /* Padding character to send */
        eol,                    /* End-Of-Line character to send */
        escchr,                 /* Connect command escape character */
        quote,                  /* Quote character in incoming data */
        **filelist,             /* List of files to be sent */
        *filnam,                /* Current file name */
        recpktÄMAXPACKSIZÅ,     /* Receive packet buffer */
        packetÄMAXPACKSIZÅ;     /* Packet buffer */

FILE    *fp,                    /* File pointer for current disk file */
        *log;                   /* File pointer for Logfile */

jmp_buf env;                    /* Environment ptr for timeout longjump */


/*
 *  m a i n
 *
 *  Main routine - parse command and options, set up the
 *  tty lines, and dispatch to the appropriate routine.
 */

main(argc,argv)
int argc;                           /* Character pointers to and count of */
char **argv;                            /* command line arguments */
ä
    char *ttyname,                      /* tty name for LINE argument */
```

```
        *cp;                            /* char pointer */
    int speed,                          /* speed of assigned tty, */
        cflg, rflg, sflg;               /* flags for CONNECT, RECEIVE, SEND */

    struct sgttyb
        rawmode,                        /* Controlling tty raw mode */
        cookedmode,                     /* Controlling tty cooked mode */
        ttymode;                        /* mode of tty line in LINE option */

    if (argc < 2) usage();              /* Make sure there's a command line */

    cp = *++argv; argv++; argc -= 2;    /* Set up pointers to args */


/*  Initialize these values and hope the first packet will get across OK */

    eol = CR;                           /* EOL for outgoing packets */
    quote = '#';                        /* Standard control-quote char "#" */
    pad = 0;                            /* No padding */
    padchar = NULL;                     /* Use null if any padding wanted */

    speed = cflg = sflg = rflg = 0;     /* Turn off all parse flags */
    ttyname = 0;                        /* Default is remote mode */

#if UCB4X                               /* Default to 7-bit masking, CRLF */
    image = FALSE;                      /* translation and filename case */
    filnamcnv = TRUE;                   /* conversion for UNIX systems */
#else
    image = TRUE;                       /* Default to no processing for */
    filnamcnv = FALSE;                  /* non-UNIX systems */
#endif

    escchr = ESCCHR;                    /* Default escape character */

    while ((*cp) != NULL)               /* Parse characters in first arg. */
        switch (*cp++)
        ä
            case 'c': cflg++; break;    /* C = Connect command */
            case 's': sflg++; break;    /* S = Send command */
            case 'r': rflg++; break;    /* R = Receive command */

            case 'd':                   /* D = Increment debug mode count */
                debug++; break;

            case 'f':
                filnamcnv = FALSE;      /* F = don't do case conversion */
                break;                  /*     on filenames */

            case 'i':                   /* I = Image (8-bit) mode */
                image = TRUE; break;    /* (this is default for non-UNIX) */

            case 'l':                   /* L = specify tty line to use */
                if (argc--) ttyname = *argv++;
                else usage();
                if (debug) printf("Line to remote host is %sÖn",ttyname);
                break;
```

```
        case 'e':                       /* E = specify escape char */
            if (argc--) escchr = **argv++;
            else usage();
            if (debug) printf("Escape char is Ö"%cÖ"Ön",escchr);
            break;
        case 'b':                       /* B = specify baud rate */
#if UCB4X
            if (argc--) speed = atoi(*argv++);
            else usage();
            if (debug) printf("Line speed to remote host is %dÖn",speed);
            break;
#else
            printmsg("Speed setting implemented for Unix only.");
            exit(1);
#endif
    å

/* Done parsing */

    if ((cflg+sflg+rflg) != 1)          /* Only one command allowed */
        usage();


    if (ttyname)                        /* If LINE was specified, we */
    ä                                   /* operate in local mode */
        ttyfd = open(ttyname,2);        /* Open the tty line */
        if (ttyfd < 0)
        ä
            printmsg("Cannot open %s",ttyname);
            exit(1);
        å
        remote = FALSE;                 /* Indicate we're in local mode */
    å
    else                                /* No LINE specified so we operate */
    ä                                   /* in remote mode (ie. controlling */
        ttyfd = 0;                      /* tty is the communications line) */
        remote = TRUE;
    å


/* Put the proper tty into the correct mode */

    if (remote)                         /* If remote, use controlling tty */
    ä
        gtty(0,&cookedmode);            /* Save current mode so we can */
        gtty(0,&rawmode);               /* restore it later */
        rawmode.sg_flags ö= (RAWöTANDEM);
        rawmode.sg_flags &= ü(ECHOöCRMOD);
        stty(0,&rawmode);               /* Put tty in raw mode */
    å
    else                                /* Local, use assigned line */
    ä
        gtty(ttyfd,&ttymode);
        ttymode.sg_flags ö= (RAWöTANDEM);
        ttymode.sg_flags &= ü(ECHOöCRMOD);
```

```
#if UCB4X                                  /* Speed changing for UNIX only */
        if (speed)                         /* User specified a speed? */
        ä
            switch(speed)                  /* Get internal system code */
            ä
                case 110: speed = B110; break;
                case 150: speed = B150; break;
                case 300: speed = B300; break;
                case 1200: speed = B1200; break;
                case 2400: speed = B2400; break;
                case 4800: speed = B4800; break;
                case 9600: speed = B9600; break;

                default:
                    printmsg("Bad line speed.");
                    exit(1);
            å
            ttymode.sg_ispeed = speed;
            ttymode.sg_ospeed = speed;
        å
#endif /* UCB4X */

        stty(ttyfd,&ttymode);              /* Put asg'd tty in raw mode */
    å


/* All set up, now execute the command that was given. */

    if (debug)
    ä
        printf("Debugging level = %dÖnÖn",debug);

        if (cflg) printf("Connect commandÖnÖn");
        if (sflg) printf("Send commandÖnÖn");
        if (rflg) printf("Receive commandÖnÖn");
    å

    if (cflg) connect();                   /* Connect command */

    if (sflg)                              /* Send command */
    ä
        if (argc--) filnam = *argv++;      /* Get file to send */
        else
        ä   if (remote)
                stty(0,&cookedmode);       /* Restore controlling tty's modes */
            usage();                       /* and give error */
        å
        fp = NULL;                         /* Indicate no file open yet */
        filelist = argv;                   /* Set up the rest of the file list */
        filecount = argc;                  /* Number of files left to send */
        if (sendsw() == FALSE)             /* Send the file(s) */
            printmsg("Send failed.");      /* Report failure */
        else                               /*  or */
            printmsg("done.");             /* success */
    å
```

```
    if (rflg)                               /* Receive command */
    ä
        if (recsw() == FALSE)               /* Receive the file(s) */
            printmsg("Receive failed.");
        else                                /* Report failure */
            printmsg("done.");              /* or success */
    å

    if (remote) stty(0,&cookedmode);        /* Restore controlling tty's modes */
å


/*
 *  s e n d s w
 *
 *  Sendsw is the state table switcher for sending files.  It loops until
 *  either it finishes, or an error is encountered.  The routines called
 *  by sendsw are responsible for changing the state.
 *
 */

sendsw()
ä
    char sinit(), sfile(), sdata(), seof(), sbreak();

    state = 'S';                            /* Send initiate is the start state */
    n = 0;                                  /* Initialize message number */
    numtry = 0;                             /* Say no tries yet */
    while(TRUE)                             /* Do this as long as necessary */
    ä
        if (debug) printf("sendsw state: %cÖn",state);
        switch(state)
        ä
            case 'S':   state = sinit();  break; /* Send-Init */
            case 'F':   state = sfile();  break; /* Send-File */
            case 'D':   state = sdata();  break; /* Send-Data */
            case 'Z':   state = seof();   break; /* Send-End-of-File */
            case 'B':   state = sbreak(); break; /* Send-Break */
            case 'C':   return (TRUE);           /* Complete */
            case 'A':   return (FALSE);          /* "Abort" */
            default:    return (FALSE);          /* Unknown, fail */
        å
    å
å


/*
 *  s i n i t
 *
 *  Send Initiate: send this host's parameters and get other side's back.
 */

char sinit()
ä
    int num, len;                           /* Packet number, length */
    if (numtry++ > MAXTRY) return('A');     /* If too many tries, give up */
```

```
    spar(packet);                           /* Fill up init infö packet */

    flushinput();                           /* Flush pending input */

    spack('S',n,6,packet);                  /* Send an S packet */
    switch(rpack(&len,&num,recpkt))         /* What was the reply? */
    ä
        case 'N':  return(state);           /* NAK, try it again */

        case 'Y':                           /* ACK */
            if (n != num)                   /* If wrong ACK, stay in S state */
                return(state);              /* and try again */
            rpar(recpkt);                   /* Get other side's init info */

            if (eol == 0) eol = 'Ön';       /* Check and set defaults */
            if (quote == 0) quote = '#';

            numtry = 0;                     /* Reset try counter */
            n = (n+1)%64;                   /* Bump packet count */
            return('F');                    /* OK, switch state to F */

        case 'E':                           /* Error packet received */
            prerrpkt(recpkt);               /* Print it out and */
            return('A');                    /* abort */

        case FALSE: return(state);          /* Receive failure, try again */

        default: return('A');               /* Anything else, just "abort" */
    å
å


/*
 *  s f i l e
 *
 *  Send File Header.
 */

char sfile()
ä
    int num, len;                           /* Packet number, length */
    char filnam1Ä50Å,                       /* Converted file name */
        *newfilnam,                         /* Pointer to file name to send */
        *cp;                                /* char pointer */

    if (numtry++ > MAXTRY) return('A');     /* If too many tries, give up */
    if (fp == NULL)                         /* If not already open, */
    ä   if (debug) printf("   Opening %s for sending.Ön",filnam);
        fp = fopen(filnam,"r");             /* open the file to be sent */
        if (fp == NULL)                     /* If bad file pointer, give up */
        ä
            error("Cannot open file %s",filnam);
            return('A');
        å
    å
```

```
    strcpy(filnam1, filnam);                /* Copy file name */
    newfilnam = cp = filnam1;
    while (*cp != 'Ö0')                     /* Strip off all leading directory */
        if (*cp++ == '/')                   /* names (ie. up to the last /). */
            newfilnam = cp;

    if (filnamcnv)                          /* Convert lower case to upper  */
        for (cp = newfilnam; *cp != 'Ö0'; cp++)
            if (*cp >= 'a' && *cp <= 'z')
                *cp ü= 040;

    len = cp - newfilnam;                   /* Compute length of new filename */

    printmsg("Sending %s as %s",filnam,newfilnam);

    spack('F',n,len,newfilnam);             /* Send an F packet */
    switch(rpack(&len,&num,recpkt))         /* What was the reply? */
    ä
        case 'N':                           /* NAK, just stay in this state, */
            num = (--num<0 ? 63:num);       /* unless it's NAK for next packet */
            if (n != num)                   /* which is just like an ACK for */
                return(state);              /* this packet so fall thru to... */

        case 'Y':                           /* ACK */
            if (n != num) return(state); /* If wrong ACK, stay in F state */
            numtry = 0;                     /* Reset try counter */
            n = (n+1)%64;                   /* Bump packet count */
            size = bufill(packet);          /* Get first data from file */
            return('D');                    /* Switch state to D */

        case 'E':                           /* Error packet received */
            prerrpkt(recpkt);               /* Print it out and */
            return('A');                    /* abort */

        case FALSE: return(state);          /* Receive failure, stay in F state */

        default:    return('A');            /* Something else, just "abort" */
    å
å

/*
 *  s d a t a
 *
 *  Send File Data
 */

char sdata()
ä
    int num, len;                           /* Packet number, length */

    if (numtry++ > MAXTRY) return('A'); /* If too many tries, give up */

    spack('D',n,size,packet);               /* Send a D packet */
    switch(rpack(&len,&num,recpkt))         /* What was the reply? */
    ä
        case 'N':                           /* NAK, just stay in this state, */
```

```
            num = (--num<0 ? 63:num);        /* unless it's NAK for next packet */
            if (n != num)                    /* which is just like an ACK for */
                return(state);               /* this packet so fall thru to... */

        case 'Y':                            /* ACK */
            if (n != num) return(state);  /* If wrong ACK, fail */
            numtry = 0;                      /* Reset try counter */
            n = (n+1)%64;                    /* Bump packet count */
            if ((size = bufill(packet)) == EOF) /* Get data from file */
                return('Z');                 /* If EOF set state to that */
            return('D');                     /* Got data, stay in state D */

        case 'E':                            /* Error packet received */
            prerrpkt(recpkt);                /* Print it out and */
            return('A');                     /* abort */

        case FALSE: return(state);           /* Receive failure, stay in D */

        default:    return('A');             /* Anything else, "abort" */
    å
å


/*
 *  s e o f
 *
 *  Send End-Of-File.
 */

char seof()
ä
    int num, len;                            /* Packet number, length */
    if (numtry++ > MAXTRY) return('A');  /* If too many tries, "abort" */
    spack('Z',n,0,packet);                   /* Send a 'Z' packet */
    switch(rpack(&len,&num,recpkt))          /* What was the reply? */
    ä
        case 'N':                            /* NAK, just stay in this state, */
            num = (--num<0 ? 63:num);        /* unless it's NAK for next packet, */
            if (n != num)                    /* which is just like an ACK for */
                return(state);               /* this packet so fall thru to... */

        case 'Y':                            /* ACK */
            if (n != num) return(state);  /* If wrong ACK, hold out */
            numtry = 0;                      /* Reset try counter */
            n = (n+1)%64;                    /* and bump packet count */
            if (debug) printf("  Closing input file %s, ",filnam);
            fclose(fp);                      /* Close the input file */
            fp = NULL;                       /* Set flag indicating no file open */

            if (debug) printf("looking for next file...Ön");
            if (gnxtfl() == FALSE)           /* No more files go? */
                return('B');                 /* if not, break, EOT, all done */
            if (debug) printf("  New file is %sÖn",filnam);
            return('F');                     /* More files, switch state to F */

        case 'E':                            /* Error packet received */
```

```
            prerrpkt(recpkt);              /* Print it out and */
            return('A');                   /* abort */

        case FALSE: return(state);         /* Receive failure, stay in Z */

        default:    return('A');           /* Something else, "abort" */
    å
å


/*
 *  s b r e a k
 *
 *  Send Break (EOT)
 */

char sbreak()
ä
    int num, len;                          /* Packet number, length */
    if (numtry++ > MAXTRY) return('A');    /* If too many tries "abort" */

    spack('B',n,0,packet);                 /* Send a B packet */
    switch (rpack(&len,&num,recpkt))       /* What was the reply? */
    ä
        case 'N':                          /* NAK, just stay in this state, */
            num = (--num<0 ? 63:num);      /* unless NAK for previous packet, */
            if (n != num)                  /* which is just like an ACK for */
                return(state);             /* this packet so fall thru to... */
        case 'Y':                          /* ACK */
            if (n != num) return(state);   /* If wrong ACK, fail */
            numtry = 0;                    /* Reset try counter */
            n = (n+1)%64;                  /* and bump packet count */
            return('C');                   /* Switch state to Complete */

        case 'E':                          /* Error packet received */
            prerrpkt(recpkt);              /* Print it out and */
            return('A');                   /* abort */

        case FALSE: return(state);         /* Receive failure, stay in B */

        default:    return ('A');          /* Other, "abort" */
    å
å


/*
 *  r e c s w
 *
 *  This is the state table switcher for receiving files.
 */

recsw()
ä
    char rinit(), rfile(), rdata();        /* Use these procedures */

    state = 'R';                           /* Receive-Init is the start state */
```

```
    n = 0;                              /* Initialize message number */
    numtry = 0;                         /* Say no tries yet */

    while(TRUE)
    ä
        if (debug) printf(" recsw state: %cÖn",state);
        switch(state)                   /* Do until done */
        ä
            case 'R':   state = rinit(); break; /* Receive-Init */
            case 'F':   state = rfile(); break; /* Receive-File */
            case 'D':   state = rdata(); break; /* Receive-Data */
            case 'C':   return(TRUE);           /* Complete state */
            case 'A':   return(FALSE);          /* "Abort" state */
        å
    å
å


/*
 *  r i n i t
 *
 *  Receive Initialization
 */

char rinit()
ä
    int len, num;                       /* Packet length, number */
    if (numtry++ > MAXTRY) return('A'); /* If too many tries, "abort" */

    switch(rpack(&len,&num,packet))     /* Get a packet */
    ä
        case 'S':                       /* Send-Init */
            rpar(packet);               /* Get the other side's init data */
            spar(packet);               /* Fill up packet with my init info */
            spack('Y',n,6,packet);      /* ACK with my parameters */
            oldtry = numtry;            /* Save old try count */
            numtry = 0;                 /* Start a new counter */
            n = (n+1)%64;               /* Bump packet number, mod 64 */
            return('F');                /* Enter File-Receive state */

        case 'E':                       /* Error packet received */
            prerrpkt(recpkt);           /* Print it out and */
            return('A');                /* abort */

        case FALSE:                     /* Didn't get packet */
            spack('N',n,0,0);           /* Return a NAK */
            return(state);              /* Keep trying */

        default:    return('A');        /* Some other packet type, "abort" */
    å
å


/*
 *  r f i l e
 *
```

```
 *  Receive File Header
 */

char rfile()
ä
    int num, len;                   /* Packet number, length */
    char filnam1Ä50Å;               /* Holds the converted file name */

    if (numtry++ > MAXTRY) return('A'); /* "abort" if too many tries */

    switch(rpack(&len,&num,packet))     /* Get a packet */
    ä
        case 'S':                   /* Send-Init, maybe our ACK lost */
            if (oldtry++ > MAXTRY) return('A'); /* If too many tries "abort" */
            if (num == ((n==0) ? 63:n-1)) /* Previous packet, mod 64? */
            ä                       /* Yes, ACK it again with  */
                spar(packet);       /* our Send-Init parameters */
                spack('Y',num,6,packet);
                numtry = 0;         /* Reset try counter */
                return(state);      /* Stay in this state */
            å
            else return('A');       /* Not previous packet, "abort" */
        case 'Z':                   /* End-Of-File */
            if (oldtry++ > MAXTRY) return('A');
            if (num == ((n==0) ? 63:n-1)) /* Previous packet, mod 64? */
            ä                       /* Yes, ACK it again. */
                spack('Y',num,0,0);
                numtry = 0;
                return(state);      /* Stay in this state */
            å
            else return('A');       /* Not previous packet, "abort" */

        case 'F':                   /* File Header (just what we want) */
            if (num != n) return('A');  /* The packet number must be right */
            strcpy(filnam1, packet);    /* Copy the file name */

            if (filnamcnv)              /* Convert upper case to lower */
                for (filnam=filnam1; *filnam != 'Ö0'; filnam++)
                    if (*filnam >= 'A' && *filnam <= 'Z')
                        *filnam ö= 040;

            if ((fp=fopen(filnam1,"w"))==NULL) /* Try to open a new file */
            ä
                error("Cannot create %s",filnam1); /* Give up if can't */
                return('A');
            å
            else                        /* OK, give message */
                printmsg("Receiving %s as %s",packet,filnam1);

            spack('Y',n,0,0);           /* Acknowledge the file header */
            oldtry = numtry;            /* Reset try counters */
            numtry = 0;                 /* ... */
            n = (n+1)%64;               /* Bump packet number, mod 64 */
            return('D');                /* Switch to Data state */

        case 'B':                       /* Break transmission (EOT) */
```

```
            if (num != n) return ('A'); /* Need right packet number here */
            spack('Y',n,0,0);           /* Say OK */
            return('C');                /* Go to complete state */

        case 'E':                       /* Error packet received */
            prerrpkt(recpkt);           /* Print it out and */
            return('A');                /* abort */

        case FALSE:                     /* Didn't get packet */
            spack('N',n,0,0);           /* Return a NAK */
            return(state);              /* Keep trying */

        default:     return ('A');      /* Some other packet, "abort" */
    å
å
/*
 *  r d a t a
 *
 *  Receive Data
 */

char rdata()
ä
    int num, len;                       /* Packet number, length */
    if (numtry++ > MAXTRY) return('A'); /* "abort" if too many tries */

    switch(rpack(&len,&num,packet))     /* Get packet */
    ä
        case 'D':                       /* Got Data packet */
            if (num != n)               /* Right packet? */
            ä                           /* No */
                if (oldtry++ > MAXTRY)
                    return('A');        /* If too many tries, abort */
                if (num == ((n==0) ? 63:n-1)) /* Else check packet number */
                ä                       /* Previous packet again? */
                    spack('Y',num,6,packet); /* Yes, re-ACK it */
                    numtry = 0;         /* Reset try counter */
                    return(state);      /* Don't write out data! */
                å
                else return('A');       /* sorry, wrong number */
            å
            /* Got data with right packet number */
            bufemp(packet,len);         /* Write the data to the file */
            spack('Y',n,0,0);           /* Acknowledge the packet */
            oldtry = numtry;            /* Reset the try counters */
            numtry = 0;                 /* ... */
            n = (n+1)%64;               /* Bump packet number, mod 64 */
            return('D');                /* Remain in data state */

        case 'F':                       /* Got a File Header */
            if (oldtry++ > MAXTRY)
                return('A');            /* If too many tries, "abort" */
            if (num == ((n==0) ? 63:n-1)) /* Else check packet number */
            ä                           /* It was the previous one */
                spack('Y',num,0,0);     /* ACK it again */
                numtry = 0;             /* Reset try counter */
```

```
                return(state);           /* Stay in Data state */
            å
            else return('A');            /* Not previous packet, "abort" */

        case 'Z':                        /* End-Of-File */
            if (num != n) return('A');   /* Must have right packet number */
            spack('Y',n,0,0);            /* OK, ACK it. */
            fclose(fp);                  /* Close the file */
            n = (n+1)%64;                /* Bump packet number */
            return('F');                 /* Go back to Receive File state */

        case 'E':                        /* Error packet received */
            prerrpkt(recpkt);            /* Print it out and */
            return('A');                 /* abort */
        case FALSE:                      /* Didn't get packet */
            spack('N',n,0,0);            /* Return a NAK */
            return(state);               /* Keep trying */

        default:    return('A');         /* Some other packet, "abort" */
    å
å


/*
 *  c o n n e c t
 *
 *  Establish a virtual terminal connection with the remote host, over an
 *  assigned tty line.
 */

connect()
ä
    int pid,                             /* Holds process id of child */
        connected;                       /* Boolean connect flag */
    char bel = 'ö07',
        c;

    struct sgttyb
        rawmode,                         /* Controlling tty raw mode */
        cookedmode;                      /* Controlling tty cooked mode */

    if (remote)                          /* Nothing to connect to in remote */
    ä                                    /* mode, so just return */
        printmsg("No line specified for connection.");
        return;
    å

    gtty(0,&cookedmode);                 /* Save current mode so we can */
    gtty(0,&rawmode);                    /* restore it later */
    rawmode.sg_flags ö= (RAWöTANDEM);
    rawmode.sg_flags &= ü(ECHOöCRMOD);
    stty(0,&rawmode);                    /* Put tty in raw mode */

    pid = fork();             /* Start fork to get typeout from remote host */
    if (pid)                       /* Parent: send type-in to remote host */
    ä
        printmsg("connected...ör");
```

```
        connected = TRUE;                   /* Put us in "connect mode" */
        while (connected)
        ä
            read(0,&c,1);                   /* Get a character */
            if ((c&0177) == escchr)         /* Check for escape character */
            ä
                read(0,&c,1);
                if ((c&0177) == escchr)
                    write(ttyfd,&c,1);
                else
                switch (c&0177)
                ä
                    case 'c':
                    case 'C':
                        connected = FALSE;
                        write(0,"ÖrÖn",2);
                        break;

                    case 'h':
                    case 'H':
                        write(0,"ÖrÖnYes, I'm still here...ÖrÖn",26);
                        break;

                    default:
                        write(0,&bel,1);
                        break;
                å
            å
            else
            ä                               /* If not escape charater, */
                write(ttyfd,&c,1);          /* write it out */
                c = NULL;                   /* Nullify it (why?) */
            å
        å
        kill(pid,9);                        /* Done, kill the child */
        wait(0);                            /* and bury him */
        stty(0,&cookedmode);                /* Restore tty mode */
        printmsg("disconnected.");
        return;                             /* Done */
    å
    else                  /* Child does the reading from the remote host */
    ä
        while(1)                            /* Do this forever */
        ä
            read(ttyfd,&c,1);
            write(1,&c,1);
        å
    å
å
/*
 *      KERMIT utilities.
 */

clkint()                                    /* Timer interrupt handler */
ä
    longjmp(env,TRUE);                      /* Tell rpack to give up */
```

```
å


/*
 *  s p a c k
 *
 *  Send a Packet
 */

spack(type,num,len,data)
char type, *data;
int num, len;
ä
    int i;                                  /* Character loop counter */
    char chksum, bufferÄ100Å;               /* Checksum, packet buffer */
    register char *bufp;                    /* Buffer pointer */

    if (debug>1)                            /* Display outgoing packet */
    ä
        if (data != NULL)
            dataÄlenÅ = 'Ö0';               /* Null-terminate data to print it */
        printf("  spack type: %cÖn",type);
        printf("         num:  %dÖn",num);
        printf("         len:  %dÖn",len);
        if (data != NULL)
            printf("        data: Ö"%sÖ"Ön",data);
    å

    bufp = buffer;                          /* Set up buffer pointer */
    for (i=1; i<=pad; i++) write(ttyfd,&padchar,1); /* Issue any padding */

    *bufp++ = SOH;                          /* Packet marker, ASCII 1 (SOH) */
    *bufp++ = tochar(len+3);                /* Send the character count */
    chksum  = tochar(len+3);                /* Initialize the checksum */
    *bufp++ = tochar(num);                  /* Packet number */
    chksum += tochar(num);                  /* Update checksum */
    *bufp++ = type;                         /* Packet type */
    chksum += type;                         /* Update checksum */
    for (i=0; i<len; i++)                   /* Loop for all data characters */
    ä
        *bufp++ = dataÄiÅ;                  /* Get a character */
        chksum += dataÄiÅ;                  /* Update checksum */
    å
    chksum = (((chksum&0300) >> 6)+chksum)&077; /* Compute final checksum */
    *bufp++ = tochar(chksum);               /* Put it in the packet */
    *bufp = eol;                            /* Extra-packet line terminator */
    write(ttyfd, buffer,bufp-buffer+1); /* Send the packet */
å

/*
 *  r p a c k
 *
 *  Read a Packet
 */

rpack(len,num,data)
```

```
int *len, *num;                         /* Packet length, number */
char *data;                             /* Packet data */
ä
    int i, done;                        /* Data character number, loop exit */
    char t,                             /* Current input character */
        type,                           /* Packet type */
        cchksum,                        /* Our (computed) checksum */
        rchksum;                        /* Checksum received from other host */

#if UCB4X                               /* TOPS-20 can't handle timeouts... */
    if (setjmp(env)) return FALSE;      /* Timed out, fail */
    signal(SIGALRM,clkint);             /* Setup the timeout */
    if ((timint > MAXTIM) öö (timint < MINTIM)) timint = MYTIME;
    alarm(timint);
#endif /* UCB4X */

    while (t != SOH)                    /* Wait for packet header */
    ä
        read(ttyfd,&t,1);
        t &= 0177;                      /* Handle parity */
    å

    done = FALSE;                       /* Got SOH, init loop */
    while (!done)                       /* Loop to get a packet */
    ä
        read(ttyfd,&t,1);               /* Get character */
        if (!image) t &= 0177;          /* Handle parity */
        if (t == SOH) continue;         /* Resynchronize if SOH */
        cchksum = t;                    /* Start the checksum */
        *len = unchar(t)-3;             /* Character count */

        read(ttyfd,&t,1);               /* Get character */
        if (!image) t &= 0177;          /* Handle parity */
        if (t == SOH) continue;         /* Resynchronize if SOH */
        cchksum = cchksum + t;          /* Update checksum */
        *num = unchar(t);               /* Packet number */
        read(ttyfd,&t,1);               /* Get character */
        if (!image) t &= 0177;          /* Handle parity */
        if (t == SOH) continue;         /* Resynchronize if SOH */
        cchksum = cchksum + t;          /* Update checksum */
        type = t;                       /* Packet type */

        for (i=0; i<*len; i++)          /* The data itself, if any */
        ä                               /* Loop for character count */
            read(ttyfd,&t,1);           /* Get character */
            if (!image) t &= 0177;      /* Handle parity */
            if (t == SOH) continue;     /* Resynch if SOH */
            cchksum = cchksum + t;      /* Update checksum */
            dataÄiÅ = t;                /* Put it in the data buffer */
        å
        dataÄ*lenÅ = 0;                 /* Mark the end of the data */

        read(ttyfd,&t,1);               /* Get last character (checksum) */
        rchksum = unchar(t);            /* Convert to numeric */
        read(ttyfd,&t,1);               /* get EOL character and toss it */
        if (!image) t &= 0177;          /* Handle parity */
```

```
        if (t == SOH) continue;             /* Resynchronize if SOH */
        done = TRUE;                        /* Got checksum, done */
    å

#if UCB4X
    alarm(0);                               /* Disable the timer interrupt */
#endif

    if (debug>1)                            /* Display incoming packet */
    ä
        if (data != NULL)
            dataÄ*lenÅ = 'Ö0';              /* Null-terminate data to print it */
        printf("  rpack type: %cÖn",type);
        printf("         num: %dÖn",*num);
        printf("         len: %dÖn",*len);
        if (data != NULL)
            printf("        data: Ö"%sÖ"Ön",data);
    å
                                            /* Fold in bits 7,8 to compute */
    cchksum = (((cchksum&0300) >> 6)+cchksum)&077; /* final checksum */

    if (cchksum != rchksum) return(FALSE);

    return(type);                           /* All OK, return packet type */
å


/*
 *  b u f i l l
 *
 *  Get a bufferful of data from the file that's being sent.
 *  Only control-quoting is done; 8-bit & repeat count prefixes are
 *  not handled.
 */
bufill(buffer)
char bufferÄÅ;                              /* Buffer */
ä
    int i,                                  /* Loop index */
        t;                                  /* Char read from file */
    char t7;                                /* 7-bit version of above */

    i = 0;                                  /* Init data buffer pointer */
    while((t = getc(fp)) != EOF)            /* Get the next character */
    ä
        t7 = t & 0177;                      /* Get low order 7 bits */

        if (t7 < SP öö t7==DEL öö t7==quote) /* Does this char require */
        ä                                   /* special handling? */
            if (t=='Ön' && !image)
            ä                               /* Do LF->CRLF mapping if !image */
                bufferÄi++Å = quote;
                bufferÄi++Å = ctl('Ör');
            å
            bufferÄi++Å = quote;            /* Quote the character */
            if (t7 != quote)
            ä
```

```
                t = ctl(t);                     /* and uncontrolify */
                t7 = ctl(t7);
            å
        å
        if (image)
            bufferÄi++Å = t;                    /* Deposit the character itself */
        else
            bufferÄi++Å = t7;

        if (i >= spsiz-8) return(i);            /* Check length */
    å
    if (i==0) return(EOF);                      /* Wind up here only on EOF */
    return(i);                                  /* Handle partial buffer */
å


/*
 *      b u f e m p
 *
 *  Put data from an incoming packet into a file.
 */

bufemp(buffer,len)
char  bufferÄÅ;                                 /* Buffer */
int   len;                                      /* Length */
ä
    int i;                                      /* Counter */
    char t;                                     /* Character holder */
    for (i=0; i<len; i++)                       /* Loop thru the data field */
    ä
        t = bufferÄiÅ;                          /* Get character */
        if (t == MYQUOTE)                       /* Control quote? */
        ä                                       /* Yes */
            t = bufferÄ++iÅ;                    /* Get the quoted character */
            if ((t & 0177) != MYQUOTE)          /* Low order bits match quote char? */
                t = ctl(t);                     /* No, uncontrollify it */
        å
        if (t==CR && !image)                    /* Don't pass CR if in image mode */
            continue;

        putc(t,fp);
    å
å


/*
 *  g n x t f l
 *
 *  Get next file in a file group
 */

gnxtfl()
ä
    if (debug) printf("   gnxtfl: filelist = Ö"%sÖ"Ön",*filelist);
    filnam = *(filelist++);
    if (filecount-- == 0) return FALSE; /* If no more, fail */
```

```
    else return TRUE;                       /* else succeed */
å


/*
 *  s p a r
 *
 *  Fill the data array with my send-init parameters
 *
 */

spar(data)
char dataÄÅ;
ä
    dataÄ0Å = tochar(MAXPACKSIZ);              /* Biggest packet I can receive */
    dataÄ1Å = tochar(MYTIME);               /* When I want to be timed out */
    dataÄ2Å = tochar(MYPAD);                /* How much padding I need */
    dataÄ3Å = ctl(MYPCHAR);                 /* Padding character I want */
    dataÄ4Å = tochar(MYEOL);                /* End-Of-Line character I want */
    dataÄ5Å = MYQUOTE;                      /* Control-Quote character I send */
å

/*  r p a r
 *
 *  Get the other host's send-init parameters
 *
 */

rpar(data)
char dataÄÅ;
ä
    spsiz = unchar(dataÄ0Å);                /* Maximum send packet size */
    timint = unchar(dataÄ1Å);               /* When I should time out */
    pad = unchar(dataÄ2Å);                  /* Number of pads to send */
    padchar = ctl(dataÄ3Å);                 /* Padding character to send */
    eol = unchar(dataÄ4Å);                  /* EOL character I must send */
    quote = dataÄ5Å;                        /* Incoming data quote character */
å


/*
 *  f l u s h i n p u t
 *
 *  Dump all pending input to clear stacked up NACK's.
 *  (Implemented only for Berkeley Unix at this time).
 */

#if UCB4X&(üNO_FIONREAD)
flushinput()
ä
    long int count;                         /* Number of bytes ready to read */
    long int i;                             /* Number of bytes to read in loop */

    ioctl(ttyfd, FIONREAD, &count);         /* See how many bytes pending read */
    if (!count) return;                     /* If zero, then no input to flush */
```

```
        printf("Ön");
        fflush(stdout);                        /* force output (UTS needs it) */
    å
å

/*
 *  e r r o r
 *
 *  Print error message.
 *
 *  If local, print error message with printmsg.
 *  If remote, send an error packet with the message.
 */

/*VARARGS1*/
error(fmt, a1, a2, a3, a4, a5)
char *fmt;
ä
    char msgÄ80Å;
    int len;

    if (remote)
    ä
        sprintf(msg,fmt,a1,a2,a3,a4,a5); /* Make it a string */
        len = strlen(msg);
        spack('E',n,len,msg);           /* Send the error packet */
    å
    else
        printmsg(fmt, a1, a2, a3, a4, a5);

    return;
å

/*
 *  p r e r r p k t
 *
 *  Print contents of error packet received from remote host.
 */
prerrpkt(msg)
char *msg;
ä
    printf("Kermit aborting with following error from remote host:Ön%sÖn",msg);
    return;
å
```

***   SLUT   ***

ALMANACKSPROGRAM

EFTERSOM DET REDAN HUNNIT BLI NYTT ÅR SÅ KOMMER HÄR ETT ALMANACKSPROGRAM SOM GER EN ALMANACKA FöR öNSKAT ÅRTAL.

* * * * * * * * * * * * * *

```
10 REM KALENDER FöR ETT VISST ÅR
20 INPUT "VILKET ÅR";Y
22 IF Y>99 GOTO 20
23 Y1=Y
24 IF Y1=0 THEN T=366 ELSE T=365
25 IF Y1<=0 GOTO 28
26 Y1=Y1-4
27 GOTO 24
28 Y=Y+1900
30 LET Y1=Y
32 M=1
33 D=1
37 LET M1=M
90 LET M=11
100 Y=Y-1
110 D1=INT(2.6*M-.2)+D+(Y-1900)+INT((Y-
1900)/4)
115 D1=D1+INT(19/4)-2*19
120 D1=D1-INT(D1/7)*7+1
121 D1=D1-1
122 D=D1*(-1)
130 DIM M(12)
135 FOR I=1 TO 6:PRINT CHR¤(10);:NEXT I
140 S=0
160 FOR N=0 TO 12:READ M(N):NEXT N
180 FOR N=1 TO 12
185 IF N<>2 GOTO 190
186 IF T=366 THEN M(N)=29
190 PRINT" ":PRINT" ":PRINT" ":PRINT"
":S=S+M(N-1)
200 PRINT "--";S;TAB(7);
210 FOR I=1 TO 18:PRINT"-";:NEXT I
220 ON N GOTO
230,240,250,260,270,280,290,300,310,320,
330,340
230 PRINT" JANUARI ";:GOTO 350
240 PRINT" FEBRUARI";:GOTO 350
250 PRINT"   MARS  ";:GOTO 350
260 PRINT"  APRIL  ";:GOTO 350
270 PRINT"   MAJ   ";:GOTO 350
280 PRINT"  JUNI   ";:GOTO 350
290 PRINT"  JULI   ";:GOTO 350
300 PRINT" AUGUSTI ";:GOTO 350
310 PRINT"SEPTEMBER";:GOTO 350
320 PRINT" OKTOBER ";:GOTO 350
330 PRINT"NOVEMBER ";:GOTO 350
340 PRINT"DECEMBER ";:GOTO 350
350 FOR I=1TO18:PRINT"-";:NEXT I
360 PRINT  T-S;"--";
380 PRINT CHR¤(10):PRINT"   SöN     MAN
TIS     ONS     TORS     FRE     LöR"
410 FOR I=1 TO 59:PRINT"-";:NEXT I
430 FOR W=1TO6
440 PRINT CHR¤(10)
450 PRINT TAB (4)
470 FOR G=1 TO 7
480 D=D+1
490 D2=D-S
500 IF D2>M(N) THEN 580
510 IF D2>0 THEN PRINT D2;
520 PRINT TAB(4+8*G);
530 NEXT G
550 IF D2=M(N) THEN 590
560 NEXT W
580 D=D-G
590 NEXT N
610 FOR I=1 TO 6:PRINT CHR¤(10);:NEXT I
620 DATA
0,31,28,31,30,31,30,31,31,30,31,30,31
640 END
```

* * * * * * * * * * * * * *

ALMANACKSPROGRAM

EFTERSOM DET REDAN HUNNIT BLI NYTT AR SA KOMMER HÄR ETT ALMANACKSPROGRAM SOM GER EN ALMANACKA FöR öNSKAT ARTAL.

* * * * * * * * * * * * * * *

```
10 REM KALENDER FöR ETT VISST AR
20 INPUT "VILKET AR";Y
22 IF Y>99 GOTO 20
23 Y1=Y
24 IF Y1=0 THEN T=366 ELSE T=365
25 IF Y1<=0 GOTO 28
26 Y1=Y1-4
27 GOTO 24
28 Y=Y+1900
30 LET Y1=Y
32 M=1
33 D=1
37 LET M1=M
90 LET M=11
100 Y=Y-1
110 D1=INT(2.6*M-.2)+D+(Y-1900)+INT((Y-
1900)/4)
115 D1=D1+INT(19/4)-2*19
120 D1=D1-INT(D1/7)*7+1
121 D1=D1-1
122 D=D1*(-1)
130 DIM M(12)
135 FOR I=1 TO 6:PRINT CHR¤(10);:NEXT I
140 S=0
160 FOR N=0 TO 12:READ M(N):NEXT N
180 FOR N=1 TO 12
185 IF N<>2 GOTO 190
186 IF T=366 THEN M(N)=29
190 PRINT" ":PRINT" ":PRINT" ":PRINT"
":S=S+M(N-1)
200 PRINT "--";S;TAB(7);
210 FOR I=1 TO 18:PRINT"-";:NEXT I
220 ON N GOTO
230,240,250,260,270,280,290,300,310,320,
330,340
230 PRINT" JANUARI ";:GOTO 350
240 PRINT" FEBRUARI";:GOTO 350
250 PRINT"   MARS   ";:GOTO 350
260 PRINT"  APRIL   ";:GOTO 350
270 PRINT"   MAJ    ";:GOTO 350
280 PRINT"  JUNI    ";:GOTO 350
290 PRINT"  JULI    ";:GOTO 350
300 PRINT" AUGUSTI ";:GOTO 350
310 PRINT"SEPTEMBER";:GOTO 350
320 PRINT" OKTOBER ";:GOTO 350
330 PRINT"NOVEMBER ";:GOTO 350
340 PRINT"DECEMBER ";:GOTO 350
350 FOR I=1TO18:PRINT"-";:NEXT I
360 PRINT  T-S;"--";
380 PRINT CHR¤(10):PRINT"   SöN      MAN
TIS      ONS      TORS     FRE      LöR"
410 FOR I=1 TO 59:PRINT"-";:NEXT I
430 FOR W=1TO6
440 PRINT CHR¤(10)
450 PRINT TAB (4)
470 FOR G=1 TO 7
480 D=D+1
490 D2=D-S
500 IF D2>M(N) THEN 580
510 IF D2>0 THEN PRINT D2;
520 PRINT TAB(4+8*G);
530 NEXT G
550 IF D2=M(N) THEN 590
560 NEXT W
580 D=D-G
590 NEXT N
610 FOR I=1 TO 6:PRINT CHR¤(10);:NEXT I
620 DATA
0,31,28,31,30,31,30,31,31,30,31,30,31
640 END
```

* * * * * * * * * * * * * * *