MPU-laren. Nr 4 1985

Tidskrift för Privatdataklubben PD68 Årgång 8



PRIVATDATAKLUBBEN
PD 68

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

REDAKTION

Gunnar Ejemoh (GE) — Red. Chef
Terapivägen 14B 7tr
141 56 HUDDINGE — 08-7747760

Ingemar Skarpås — Annons Chef
Töjnavägen 42A
191 44 SOLLENTUNA — 08-969696

STYRELSE

Bo-Erik Sandholm — Ordf.
Lingvägen 217
123 59 FARSTA — 08-93 32 66

Ulf Holm — Sekr.
Artillerigatan 69B 3tr
114 45 STOCKHOLM — 08-783 05 01

Leif Ulfström (LUL) — Kassör
Fornhöjdsvägen 70
151 58 SÖDERTÄLJE — 0755-187 72

Göran Anens — V. ordf.
Källbrinksvägen 38C
141 31 HUDDINGE — 08-711 17 87

Lars Joas — V. sekr.
Helsingforsgatan 21
163 26 SPÅNGA — 08-751 30 55

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

Mauritz Lahti — Suppl.
Terapivägen 14C, 11tr.
141 56 HUDDINGE

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.

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 returneras omgående.

KLUBBDATOR

Tel. nr. till klubbdatorn:

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

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

INFöR ETT NYTT AR.

Vi i styrelsen tackar alla medlemmar för det gångna året och hoppas att vi får se er som medlemmar även under nästa år. Vi hoppas att vi kan få en ökad aktivitet under nästa år. Vi har sålt ca 20 st Danskdatorer, och vi tror på en fortsatt aktivitet med FLEX.

Vi har igen lyckats få ihop ett nummer av MPUlaren. Ett stort tack till Gunnar Ejemo som lägger ned mycket arbete för att vi skall få ut alla nummer av MPUlaren.

68 000: Klubben har köpt ett 68k kit med 10Mb Winchester av en medlem. 68k maskinen kommer att användas för anpassing av program till IDRIS samt för kopiering och distribution av program.

Vi har tillgång till ett antal public domain program för UNIX. Om någon känner för att hjälpa till med översättningen till IDRIS, skriv ett brev till klubben märkt IDRIS.

BYGGPROJEKT: Ett projekt blir troligen en billig terminal med lite grafik, den bygger på 6845 och en 6809 eller 6803.

Styrelsen har kontakt med en konstruktör av ett avancerat grafikkort. Det skall kunna bestyckas efter plånbokens tjocklek.

Seriell ingång och/eller VME buss. 2048*2048 intern ritarea, 800*590 visas i ett flyttbart fönster. Max 2Mb bildminne ger 16 färger ur en palett på 256k. Den billigaste varianten ger enbart svart/vitt och kan användas med mindre minne.

BES

INNEHÅLLSFöRTECKNING

REDAKTöR'N HAR ORDET..4
68000-TIPS.....5
WINCH. O. FLOPPYPROGR.7
AUTOLOGPROGRAM.....27
VT100-PROGRAM.....33
SMALL-C.....39
CA-790 TERM.PROGRAM..40

REDAKTöR'N HAR ORDET

DET ÄR MIG ETT SANT NöJE ATT FÅ KONSTATERA ATT UPPROPEN OM MERA BIDRAG TILL TIDNINGEN BöRJAR BÄRA FRUKT.
INTE MINST 68000-FOLKET BöRJAR NU KOMMA IGÅNG OCH PRESTERAR PROGRAM OCH HÅRDVARUUTVECKLING.
TVÅ AV KLUBBENS 68000-GURUS, PER WESTERBERG OCH BENGT PETTERSSON HAR INTRESSANTA BIDRAG ANG. FLOPPY OCH WINCHESTERANPASSNING.

DETTA BETYDER DOCK INTE ATT 6809-FOLKET BLIR LOTTLöSA, UTAN MERPARTEN AV TIDNINGEN ÄR FORTFARANDE TILLÄGNAD 8-BITARNA.

68000-BUG :

VISSA 68000-KORT HAR PROBLEM MED INSTABILITET. DETTA HAR SVEN-ÅKE ERIKSSON KIKAT NÄRMARE PÅ OCH FUNNIT ATT DET KAN BOTAS ENLIGT FöLJANDE :

TAG BORT DEN 555-KRETS SOM FINNS PÅ KORTET.
KOPPLA IN EN 4024-KRETS MELLAN KRETS U4 OCH 555:AN ENLIGT FöLJANDE :

FöRBIND PIN 6 PÅ U4 MED PIN 1 PÅ 4024:AN.
FöRBIND PIN 6 PÅ 4024:AN MED PIN 3 PÅ 555:ANS SOCKEL.

TIPS FöR PLACERING AV 4024:AN :

* KLIPP AV ALLA TUNNA BEN PÅ 4024:AN SÅ ATT BARA DE HORISONTELLA TJOCKARE DELARNA AV BENEN STICKER UT (LÄMNA PIN 14 ORöRD).
* LIMMA KRETSEN PÅ U4 SOM EN PIGGY-BACK KRETS (ANVÄND EV. EN BIT DUBBELHÄFTANDE TAPE).
* LöD IHOP PIN 14 PÅ KRETSARNA. (VCC)
* LöD PIN 7 PÅ 4024:AN TILL PIN 2 PÅ U4 (JORD).
* LöD PIN 6 PÅ U4 TILL PIN 1 PÅ 4024:AN.
* LöD EN VIRTRÅD TILL PIN 6 PÅ 4024:AN. FöRBIND DENNA TRÅD MED PIN 3 PÅ 555:ANS SOCKEL.
* TUTA OCH KöR.

EN RIKTIGT GOD JUL OCH ETT GOTT NYTT
ÅR PÅ ER ALLA öNSKAR

GUNNAR EJEMOH

******* 6 8 0 0 0 - TIPS *******

Ni som nu har fått systemet att fungera och har eller tänker skaffa winchester får här lite tips om uppstarten av en ny winchester.

Först måste mrea/udc.c anpassas till den kontroller och winchester man har (listan på den nya udc.c finns på annan plats i tidningen).
Där ska "VOID initudcdata" ändras så att "DISK_SIZE" sätts till det antal 512 bytes sektorer winchestern har (OBS IDRIS kan bara hantera 32 megabyte per filsystem) och det som börjar på "winch" ska anpassas efter de kommandon som kontrollern har och det som börjar med "drv-Ro" anpassas till den winchester man har.
Även filen mrea/abs.s (i den gamla IDRISen mrea/init.c) måste ändras för att tala om vart swapp-arean ska starta.

För att starta upp winchestern från floppyn så laddar man IDRIS och gör en patch på adress 1154 (66 till 60) som gör att man laddar data till winchester kontrollern men startar från floppyn.
Sätt root-disketten i floppy drive 0
Starta IDRIS: G 101E.
Logga på som root.
Gör ett device som kan formatera winchestern :

```
mkdev -b2 -u8 /dev/udc8
```

Gör själva formateringen:

```
echo h > /dev/udc8
```

/dev/udc8 betyder att IDRIS skall formatera winchestern.

Gör ett filsystem

```
mkfs -sxxxxx /dev/udc0
```

xxxxx står för antal 512 bytes block som finns tillgängliga.
Totalt antal 512 bytes block - 2000 (swapp size) = xxxxx
Swapp arean är det ställe där datorn lägger data när minnet är fullt.

Montera winchestern under /y :

```
mount /dev/udc0 /y
```

Om inte directory /y finns så görs detta med: mkdir /y

Montera originaldisk 1 i floppy 1 under /x

```
mount /dev/fdc1 /x
```

Sen kopieras floppyn till winchestern med :

```
cp -rd /x /y
```

Montera bort floppy 1 :

```
mount -u /dev/fdc1
```

Byt originaldisk 1 till originaldisk 2 och motera den:

```
mount /dev/fdc1 /x
```

Nu börjar det krångliga, för från disk 2 skall filerna spridas ut på olika directorys. Detta görs med:

```
mkdir /y/lib/hdrs
cp -r /x/librest/hdrs /y/lib/hdrs
cp /x/librest/libp.68k /y/lib/libp.68k
mkdir /y/etc/bin
cp /x/* /y/etc/bin
mv /y/etc/bin/p1 /y/odd/p1
mv /y/etc/bin/p2.68k /y/odd/p2.68k
```

Denna kopiering kan säkert göras både snabbare och enklare om man är bra på UNIX.

Montera bort originaldisk 2 och montera originaldisk 3 enligt ovan och kopiera den:

```
cp -rd /x/stage /y/stage
```

Montera bort originaldisk 3

Gör kommandot "sync" flera gånger och tryck sedan på reset-knappen. Ladda IDRIS och starta från 101E och förhoppningsvis så startar systemet upp från winchestern.

Nu finns det en sak kvar att göra, det är att ändra /etc/bin/c.proto.
På rad 2 skall /etc/bin/librest/hdrs ändras till /lib/hdrs.
På rad 3 skall /etc/bin/p1 ändras till /odd/p1
På rad 4 skall /etc/bin/p2.68k ändras till /odd/p2.68k

Filen /lib/help.ctrx ska byta namn till /lib/X.ctrx för att THIEF skall hitta hjälp-filen (man kan även ändra i THIEF-filen help.c).

Nu kan ni sätta igång och skriva program som vi alla kan ha nytta av.

Hälsningar

Per Westerberg

********** 6 8 0 0 0 - TIPS ********

För er som vill ha de nya IDRIS-filerna där en del förbättringar av felhanteringen är införda och några buggar rättade kan skicka in en 80 spårs disk till klubben (OBS sänd även med returporto). Även den tidigare saknade filen till THIEF och källkoden till KERMIT sänds med.

```
/*  VME-68000 SASI io handler
 *  NorSoft Consultants
 *  Monoboard computer
 *  \yvind Moe
 *  19/1-85
 *  Altered for Xebec 1410 and Rodime Ro352
 *  Bengt Pettersson 8/9-85
 */
#include <std.h>
#include <res.h>
#include <bio.h>
#include <usr.h>


#define PIA  struct pia
PIA  {
   unsigned char  gen_conreg,  space1,
                  service_reg, space2,
                  A_dirreg,    space3,
                  B_dirreg,    space4,
                  C_dirreg,    space5,
                  irq_reg,     space6,
                  A_conreg,    space7,
                  B_conreg,    space8,
                  A_datareg,   space9,
                  B_datareg,   space10;
      };
PIA  *pia_adr  = {0};

#define  S_REQ      0x01
#define  S_IO       0x02
#define  S_CD       0x04
#define  S_MSG      0x08
#define  S_BUSY     0x10
#define  S_ACK      0xdf
#define  S_RST      0xbf
#define  S_SEL      0x7f
#define  S_HIGH     0xff
#define  S_ON       (S_IO|S_CD|S_MSG|S_BUSY)
#define  S_FREE     (S_IO|S_CD|S_MSG|S_BUSY|S_REQ)

#define  PIA_DATA   0x04
#define  PIA_DIR    0x00
#define  PIA_IN     0x00
#define  PIA_OUT    0xff
unsigned char *x_adr = 0;  /* transfer address */

#define COMND  struct wi_cmd
COMND {
   unsigned char  code;
   unsigned char  lun;
   unsigned short log_adr;
   unsigned char  nmrofblocks;
   unsigned char  control;
   };
```

```
COMND  winch[8] = {0};

#define SS_READY  0
#define SS_TR00   1
#define SS_SENSE  2
#define SS_FORMAT 3
#define SS_READ   4
#define SS_WRITE  5
#define SS_ALT    6
#define SS_PAR    7

unsigned char drv_Ro352[8] = {0};

#define SENSE struct sense_bytes
SENSE {
   unsigned char  sense;
   unsigned char  lun;
   unsigned short log_adr;
   };
SENSE  drive_sense = {0};

/*  inline code :  */

#define   Select                              \
   pia->B_dirreg  = PIA_OUT;                  \
   pia->B_datareg = 0xfe;                     \
   pia->A_datareg = S_SEL;                    \
   while (pia->A_datareg & S_BUSY) ;          \
   pia->A_datareg = S_HIGH;                   \
   pia->B_dirreg  = PIA_IN;                   \
   while ((pia->A_datareg & S_ON) != (S_IO|S_MSG))

#define   put_byte(c)                         \
   if (!(pia->A_datareg & S_IO)) {            \
      putfmt("Unexpected IO state(1)\n");\
      return(1);                              \
      }                                       \
   pia->B_dirreg  = PIA_OUT;                  \
   pia->B_datareg = ~(*c);                    \
   while (pia->A_datareg & S_REQ);            \
   pia->A_datareg = S_ACK;                    \
   while (!(pia->A_datareg & S_REQ)) ;        \
   pia->A_datareg = S_HIGH;                   \
   pia->B_dirreg  = PIA_IN

#define   ready                               \
   while ((pia->A_datareg & (S_FREE)) != (S_FREE))

BUF rudcbuf   = {0};
DEVTAB udctab = {0};

#define NUNIT      1
#define DISK_SIZE  20808
#define PIA_ADR    0xf879c1
```

```
unsigned long disk_size = 0;

/* initialize the block device data
 */

VOID initudcdata()
{
    pia_adr = PIA_ADR;
    /* Test drive ready     */
    winch[0].code          = 0x00;
/* Recalib,clr err,tr00*/
winch[1].code          = 0x01;
/* Request sense          */
winch[2].code          = 0x03;
/* Format drive           */
winch[3].code          = 0x04;
winch[3].nmrofblocks = 7;
/* Read sector            */
winch[4].code          = 0x08;
winch[4].control       = 7;
winch[4].nmrofblocks = 1;
/* Write sector           */
winch[5].code          = 0x0a;
winch[5].control       = 7;
winch[5].nmrofblocks = 1;
/* Assign alt track       */
winch[6].code          = 0x0e;
    /* Assign disk param.   */
    winch[7].code          = 0x0c;
    drv_Ro352[0]   = 0x01;
    drv_Ro352[1]   = 0x32;
    drv_Ro352[2]   = 4;
    drv_Ro352[3]   = 0;
    drv_Ro352[4]   = 0;
    drv_Ro352[5]   = 0;
    drv_Ro352[6]   = 0;
    drv_Ro352[7]   = 11;
    disk_size      = DISK_SIZE;
}

VOID short_delay()
{
    int i;
    for(i=0; i<1000; i++)
        ;
}

/* initialize the sasi hostadapter and controller
 */

udcinit()
{
   extern    PIA *pia_adr;
   register  PIA *pia = pia_adr;
```

```
    init_pia();
    /* reset */
    pia->A_datareg = S_RST;
    short_delay();
    pia->A_datareg = S_HIGH;
    short_delay();
    if(!(pia->A_datareg & S_BUSY)) return(1);  /* if no winch */
    ready;
    if (assign_disk_par(0) != 0 ) return(2);
    return(0);
}

init_pia()
{
    extern   PIA *pia_adr;
    register PIA *pia = pia_adr;

    pia->gen_conreg  = 0x00;
    pia->service_reg = 0x00;
    pia->A_conreg  = 0x80;
    pia->B_conreg  = 0x80;
    pia->B_dirreg  = PIA_IN;
    pia->A_dirreg  = 0xe0;
    pia->A_datareg = S_HIGH;
    return(0);
}

assign_disk_par(lun)
unsigned char lun;
{
    extern unsigned char drv_Ro352[];
    extern unsigned char *x_adr;

    x_adr            = drv_Ro352;
    if (send_cmd(SS_PAR,lun,0,0) != 0) return(1);
    if (send_cmd(SS_READY,lun,0,0) != 0) return(3);
    if (send_cmd(SS_TR00,lun,0,0) != 0)  return(5);
    return(0);
}
/*  unstructured read
 */
VOID udcread(dev)
    DEV dev;
    {
    IMPORT BUF rudcbuf;
    IMPORT VOID udcstrat();

    physio(&udcstrat, &rudcbuf, dev, B_READ);
    }

/*  queue udc i/o
 */
VOID udcstrat(pb)
    FAST BUF *pb;
    {
```

```
    IMPORT DEVTAB udctab;
    extern unsigned long disk_size;
    UTINY devm = dminor(pb->b_dev);
    char confirm;

    if ((NUNIT <= devm) || (disk_size <=
        pb->b_blkno + ((pb->b_count + pb->b_resid) >> 9)))
        {
        if (devm >= 8) {
            devm -= 8;
            putfmt("Formatting udc%i........\n",devm);
            send_cmd(SS_FORMAT,(devm) << 5,0,0);
        }
        pb->b_flag =| B_ERROR;
        iodone(pb);
        }
    else
        {
        spl3();
        enq(&udctab, pb);
        udcstart();
        spl0();
        }
    }


/*  start i/o on UDC
 */
VOID udcstart()
{
    IMPORT DEVTAB udctab;
    FAST BUF *pb;
    FAST COUNT devm;
    BITS     ps = spl7();
    extern unsigned char *x_adr;

    while (pb = udctab.d_next)
        {
        udctab.d_stat = YES;
        devm = dminor(pb->b_dev);
        x_adr = pb->b_addr;
        if (send_cmd((pb->b_flag & B_READ) ? SS_READ : SS_WRITE
            (devm)<<5,
            pb->b_blkno,
            (pb->b_count>>9)) != 0) {
            ioerror(pb, pb->b_flag, 0);
            pb->b_flag =| B_ERROR;
        }
        udctab.d_stat = NO;
        udctab.d_nerr = 0;
        pb->b_resid = 0;
        spl(ps);
        iodone(deq(&udctab));
    }
}
```

```
/*   unstructured write
 */
VOID udcwrite(dev)
    DEV dev;
    {
    IMPORT BUF rudcbuf;
    IMPORT VOID udcstrat();

    physio(&udcstrat, &rudcbuf, dev, 0);
    }

get_sense(lun)
unsigned char lun;
{
   extern   SENSE  drive_sense;
   extern unsigned char *x_adr;

   x_adr           = &drive_sense;
   if (send_cmd(SS_SENSE,lun,0,0) != 0) {
      putfmt("Can´t get drive sense\n");
      return(1);
      }
   putfmt("Drive sense parametres :\n");
   putfmt("Error no: %h\n",drive_sense.sense);
   putfmt("Unit  no: %h\n",drive_sense.lun);
   putfmt("Sector  : %h\n",drive_sense.log_adr);
   return(0);
}

send_cmd(cmd,lun,log_sec,blkcnt)
unsigned char cmd,lun;
unsigned short log_sec;
unsigned long blkcnt;
{
   extern   PIA *pia_adr;
   register PIA *pia = pia_adr;
   unsigned short n;
   extern   COMND winch[];
   register char  *p;
   char status, message;

   status = 0;
   spl0();
if (!(pia->A_datareg & S_REQ)) {
   putfmt("Controller is busy, retry\n");
   udcinit();
   if (!(pia->A_datareg & S_REQ)) {
      putfmt("Controller is still busy\n");
      return(2);
      }
   }
Select;
while ((pia->A_datareg & S_ON) != (S_IO|S_MSG)) ;
switch (cmd) {
```

```
        case SS_READ :
        case SS_WRITE: winch[cmd].log_adr = log_sec;
                       winch[cmd].nmrofblocks = blkcnt;
                       break;
        case SS_ALT  : break;
        default      : break;
        }
    winch[cmd].lun = lun;
    for (n=0, p = &winch[cmd]; n <= 5 ; n++)  { put_byte(p++);}
    switch (cmd) {
       case SS_SENSE: while ((pia->A_datareg & S_ON) != (S_CD|S_MSG)) ;
                      if(swap_in(4))
                          status = 1;
                      break;
       case SS_READ : while ((pia->A_datareg & S_ON) != (S_CD|S_MSG)) ;
                      if(swap_in(512*blkcnt))
                          status = 1;
                      break;
       case SS_WRITE: while ((pia->A_datareg & S_ON) != (S_IO|S_CD|S_MSG)) ;
                      if(swap_out(512*blkcnt))
                          status = 1;
                      break;
       case SS_PAR  : while ((pia->A_datareg & S_ON) != (S_IO|S_CD|S_MSG)) ;
                      if(swap_out(8))
                          status = 1;
                      break;
       default      : while ((pia->A_datareg & S_ON) != S_MSG) ;
                      break;
       }
    /* status : */
    while (pia->A_datareg & (S_BUSY|S_CD|S_IO|S_REQ)) ;
    status          =| ~(pia->B_datareg);
    pia->A_datareg = S_ACK;
    while (!(pia->A_datareg & S_REQ));
    pia->A_datareg = S_HIGH;
    /* message : */
    while (pia->A_datareg & (S_ON|S_REQ)) ;
    while(pia->A_datareg & S_REQ) ;
    message         = ~(pia->B_datareg);
    pia->A_datareg = S_ACK;
    while (!(pia->A_datareg & S_REQ));
    pia->A_datareg = S_HIGH;
    ready;
    if (status && (cmd != SS_SENSE)) get_sense(lun);
    if (status) return(1);
    return(0);
}

swap_in(n)
register unsigned int n;
{
    extern   PIA *pia_adr;
    register PIA *pia = pia_adr;
    register char *adr;
    extern unsigned char *x_adr;
```

```
    adr = x_adr;
    for ( ; adr < (x_adr + n) ; ) {
       if (!(pia->A_datareg & S_CD))
          return(1);
       while (pia->A_datareg & S_REQ);
       *adr++             = ~(pia->B_datareg);
       pia->A_datareg  = S_ACK;
       while (!(pia->A_datareg & S_REQ)) ;
       pia->A_datareg  = S_HIGH;
       }
       return(0);
}
swap_out(n)
register unsigned int n;
{
    extern   PIA *pia_adr;
    register PIA *pia = pia_adr;
    register char *adr;
    extern unsigned char *x_adr;

    adr = x_adr;
    for ( ; adr < (x_adr + n) ; ) {
       if (!(pia->A_datareg & S_CD))
          return(1);
       pia->B_dirreg  = PIA_OUT;
       pia->B_datareg = ~(*adr++);
       while (pia->A_datareg & S_REQ);
       pia->A_datareg = S_ACK;
       while (!(pia->A_datareg & S_REQ)) ;
       pia->A_datareg = S_HIGH;
       pia->B_dirreg  = PIA_IN;
       }
       return(0);
}
```

```
/*  VME-68000 floppy disk driver
 *  NorSoft Consultants
 *  Monoboard computer
 *  \yvind Moe
 *  19/1-85
 *  Modified by BePe 851013, 851014
 */
#include <std.h>
#include <res.h>
#include <bio.h>
#include <usr.h>

/*  the timeout daemon
 */
LOCAL BOOL  daemon   = {0};
LOCAL COUNT timeleft = {0};

#define MAXBLOCKS  1600
#define LATCH_ADDR 0xf878c0
#define LATCH_DATA 0x0c00
#define MOTOR_BIT  0x0040
#define NRETRY     5
#define NFDC       2

BUF rfdcbuf   = {0};
DEVTAB fdctab = {0};

/*  unstructured read
 */
VOID fdcread(dev)
DEV dev;
{
    IMPORT BUF rfdcbuf;
    IMPORT VOID fdcstrat();

    physio(&fdcstrat, &rfdcbuf, dev, B_READ);
}

/*  queue fdc i/o
 */
VOID fdcstrat(pb)
FAST BUF *pb;
{
    IMPORT DEVTAB fdctab;
UTINY devm = dminor(pb->b_dev);

if (devm >= NFDC || MAXBLOCKS <= pb->b_blkno) {
    if (devm >= NFDC)
        putfmt("Only two floppy drives available\n");
    pb->b_flag |= B_ERROR;
    iodone(pb);
    }
else
    {
    spl3();
```

```
        enq(&fdctab, pb);
        fdcstart();
        spl0();
        }
}

/*  turn drive motor off after timeout
 */
VOID motor()
{
IMPORT BOOL daemon;
IMPORT COUNT timeleft;
IMPORT DEVTAB fdctab;
unsigned short *p;

daemon = NO;
if (--timeleft > 0 ) {
    daemon = YES;
    if (timeleft == 3) {
        p  = LATCH_ADDR;
        *p = LATCH_DATA | MOTOR_BIT;
    }
    timeout(&motor, 0, 60);
}
else {
    p  = LATCH_ADDR;
    *p = LATCH_DATA;
    }
}

/*  start i/o on UDC
 */
VOID fdcstart()
{
    IMPORT COUNT timeleft;
    IMPORT BOOL daemon;
    IMPORT DEVTAB fdctab;
    FAST BUF *pb;
    BITS     ps = spl7();
    FAST short s;
    short rd5sec(), wt5sec();

    while (pb = fdctab.d_next)
    {
fdctab.d_stat = YES;
if (pb->b_flag & B_READ)
  s = rd5sec(pb->b_addr,pb->b_blkno,dminor(pb->b_dev));
else
  s = wt5sec(pb->b_addr,pb->b_blkno,dminor(pb->b_dev));
if(s)
{
    pb->b_flag |= B_ERROR;
    ioerror(pb, pb->b_flag, s);
}
spl7();
```

```
        timeleft = 5;
        if (!daemon) {
            timeout(&motor, 0, 60);
            daemon = YES;
        }
        fdctab.d_stat = NO;
        fdctab.d_nerr = 0;
        if(pb->b_flag & B_ERROR)
            pb->b_resid = pb->b_count;
        else
            pb->b_resid = 0;
        iodone(deq(&fdctab));
    }
}

/*  unstructured write
 */
VOID fdcwrite(dev)
DEV dev;
{
    IMPORT BUF rfdcbuf;
    IMPORT VOID fdcstrat();

    physio(&fdcstrat, &rfdcbuf, dev, 0);
}

short rd5sec(p,sec,dev)
char  *p   ;
short sec  ;
short dev  ;
{
   char  i = 0;
   short s ;

   do  {
      s = _rd5sec(p,sec,dev) & 0xbf;
      if ( s )  {
         if ( s & 0x80 )  {
            i = 5 ;
            }
         else  {
            _init5fl(dev);
            }
         }
   i++ ;
   }  while ( ( i<5 ) && s ) ;
if ( s )  {
   putfmt("\nrd5sec :\n") ;
   if ( s & 0x80 ) putfmt("Drive not ready\n") ;
   if ( s & 0x20 ) putfmt("Deleted data mark\n") ;
   if ( s & 0x10 ) putfmt("Record not found\n") ;
   if ( s & 0x8  ) putfmt("CRC error\n") ;
   if ( s & 0x4  ) putfmt("Lost data\n") ;
   if ( s & 0x2  ) putfmt("Address out of range\n") ;
   if ( s & 0x1  ) putfmt("Seek error\n") ;
```

```
        putfmt("Cannot read the floppy\n");
        }
    return(s) ;
}


short wt5sec(p,sec,dev)
char  *p   ;
short sec  ;
short dev  ;
{
    char  i = 0;
    short s ;

    do  {
        s = _wt5sec(p,sec,dev) & 0xff;
        if ( s )  {
            if ( s & 0x80 )  {
                i = 5 ;
                }
            else  if ( s & 0x40 ) {
                i = 5;
                }
            else  {
                _init5fl(dev);
                }
            }
        i++ ;
        }  while ( ( i<5 ) && s ) ;
    if ( s )  {
        putfmt("\nwt5sec :\n") ;
        if ( s & 0x80 ) putfmt("Drive not ready\n") ;
        if ( s & 0x40 ) putfmt("Write protected\n") ;
        if ( s & 0x20 ) putfmt("Write fault\n") ;
        if ( s & 0x10 ) putfmt("Record not found\n") ;
        if ( s & 0x8  ) putfmt("CRC error\n") ;
        if ( s & 0x4  ) putfmt("Lost data\n") ;
        if ( s & 0x2  ) putfmt("Address out of range\n") ;
        if ( s & 0x1  ) putfmt("Seek error\n") ;
        putfmt("Cannot write to the floppy\n");
        }
    return(s) ;
}
```

```
*   VME-68000 Floppy assembly driver routines
*   WD-2797 Controller
*   Richard Seim
*   Modified for monoboard computer
*   15/10-84 \.Moe
*   Modified by BePe 851013, 851014
*
        .text
*
fladdr    = 0xf87901
ladisp    = -65
s8inch      = 0x0c80          * select 8 inch drive or 2Mhz to floppy
                                                          controller
motor       = 0x0c40
motwait     = 400000
hloaddel    = 22223           * gives 50 ms delay for head load.
*
*****************************************************************
*
*    init5fld()   - initialize the floppy data
*
*****************************************************************
*
    .globl  _init5fld
_init5fld:
    clr.b   drvno
    clr.b   latch
    clr.l   drvtbl
    clr.l   drvtbl+4
    rts
*
*****************************************************************
*
*   _init5fl(dn)  - library routine, initiate floppy
*
*     parameter :  dn - device number (two bytes)
*                    dn = 0x0  - drive 0
*                    dn = 0x1  - drive 1
*
*****************************************************************
*
        .globl   __init5fl
__init5fl:
    movem.l d0/d6/a0-a1,-(a7)
    move.w  22(a7),d7         *  device number
    and.w   #0x0f,d7
    move.b  d7,drvno
    lea     drvtbl,a1
    move.b  #0xff,0(a1,d7.w)     * signal restore on this drive
    movem.l (a7)+,d0/d6/a0-a1
    rts
*
*
*
*****************************************************************
*
```

```
*   _rd5sec(str,sn,dn)  - library routine, read one sector from diskette
*
*                         parameters :
*                         str - pointer to string buffer   (4 bytes)
*                         sn  - sector number              (2 bytes)
*                         dn  - device number              (2 bytes)
*
*************************************************************************
*
          .globl    __rd5sec
__rd5sec:
          movem.l d0-d6/a0-a3,-(a7)
          move.l  48(a7),a3     *  transfer address
          move.l  52(a7),d1     *  sector number
          move.l  56(a7),d7     *  device number
         and.l    #0x7ff,d1
         and.l    #0x0f,d7
          bsr      enterfl
          tst.w    d7
          bne.s    c.rdsend
          bsr      rdsec
c.rdsend:
          bsr      exitfl
          movem.l (a7)+,d0-d6/a0-a3
          rts
*
*
*************************************************************************
*
*    _wt5sec(str,sn,dn)  - library routine, write one sector to diskette
*
*                          parameters :
*                          str - pointer to string buffer   (4 bytes)
*                          sn  - sector number              (2 bytes)
*                          dn  - device number              (2 bytes)
*
*************************************************************************
*
          .globl    __wt5sec
__wt5sec:
         movem.l d0-d6/a0-a3,-(a7)
         move.l  48(a7),a3     *  transfer address
         move.l  52(a7),d1     *  sector number
         move.l  56(a7),d7     *  device number
        and.l    #0x7ff,d1
        and.l    #0x0f,d7
             bsr.s    enterfl
             tst.w    d7
             bne.s    c.wtsend
             bsr      wtsec
   c.wtsend:
       bsr exitfl
             movem.l (a7)+,d0-d6/a0-a3
             rts
   *
```

```
*
*
*
*****************************************************************
*
*    subroutine enterfl
*
*    input  : d7.b = drive number
*    output : a0.l = floppy device address
*             d7.w = status
*    local  : d6.l = parameter
*
*****************************************************************
enterfl:
     lea      fladdr,a0
     move.b   latch,d0
     and.w    #0xf0,d0
     moveq.l  #1,d5
     lsl.w    d7,d5
     or.b     d5,d0
     or.b     #motor,d0
     move.b   d0,latch
     or.w     #0x0c00,d0
     move.w   d0,ladisp(a0)
     move.l   #motwait,d6
     btst     #7,(a0)
     sne d0              * set flag if motor startup needed
                         * this will include head load time.
1:
     btst     #7,(a0)       * drive ready ?
     beq.s    1f
     subq.l   #1,d6
     bne 1b
     move.w   #0x8080,d7
     bra.s    9f
1:
     lea drvtbl,a1
     add.l    d7,a1
     cmp.b    drvno,d7          * drive change ?
     beq.s    1f
     move.b   d7,drvno          * update current drive
     move.b   4(a1),2(a0)       * update track register
     tst.b    d0                * check for needed head load time
     bne.s    1f
     move.l #hloaddel,d6
     bsr delaycy
1:   tst.b    (a1)
     beq.s    1f             * drive allready initialized
     clr.b (a1)
     clr.b 4(a1)
     bsr restore
     tst.w    d7
     beq.s    1f
     move.b   #0xff,(a1)        * mark not initialized
     move.w   #0x8001,d7        * seek error
```

```
    bra.s   9f
1:  clr.l   d7
9:  rts

*
*****************************************************************
*
*   subroutine exitfloppy
*
*   input  : a0.l = floppy device address
*   local  : d0.l = scratch
*
*****************************************************************
*
*
exitfl:
    movem.l d7/a1,-(sp)
    lea     drvtbl,a1
    clr.l   d7
    move.b  drvno,d7
    move.b  2(a0),4(a1,d7.w)     * save current track position
    movem.l (sp)+,d7/a1
    rts
*
*
*****************************************************************
*
*   subroutine restore (seek track 0)
*
*   input  :  a0.l = floppy device address
*   local  :  a1.l = timer scratch
*             d6.w = timer parameter
*   output :  d7.w = status
*
*****************************************************************
*
restore:
    movem.l d0/d6,-(a7)
    moveq   #0x08,d0          * restore command
    bra     seekx
*
*
*****************************************************************
*
*   subroutine  read sector
*
*   input  : a0.l = floppy device address
*            a3.l = start ram address
*            d1.l = sector   = track*numsec+sector-1
*   output : d7.w = status
*
*****************************************************************
*
rdsec:
        movem.l d0-d6/a4,-(a7)
```

```
        bsr     calpar
        tst.w   d7
    beq.s   1f
    move.w  #0x8002,d7
        bra.s   rdsec9      *  lsn out of range
1:
        bsr     seek
        tst.w   d7
    beq.s   1f
    move.w  #0x8001,d7
        bra.s   rdsec9      *  seek error
1:
    move.b  #0x88,d5    *  read command
        lsl.b   #1,d2
        or.b    d2,d5       *  update side info
        move.l  #0,d1
        move.l  #1,d2
        move.l  a0,a4
        add.l   #6,a4       *  pointer to FDC data reg
        move.b  d3,4(a0)    *  set sector number
    jsr _spl7       *  disable interupt
        move.b  d5,(a0)     *  write FDC command
        moveq   #8,d6
        bsr     delaycy     *  let FDC reg settle
        bra.s   rdloop
rdloop1:
        move.b  (a4),(a3)+  *  read one byte
rdloop:
        move.b  (a0),d0     *  read FDC status
        btst    d1,d0       *  still busy ?
        beq.s   rdsec1      *  completed
        btst    d2,d0       *  data request ?
        bne     rdloop1
        bra     rdloop
rdsec1:
    jsr _spl0       * enable interupt
        clr.l   d7
        move.b  (a0),d7     *  update return status
rdsec9:
        movem.l (a7)+,d0-d6/a4
        rts
*
*
*****************************************************************
*
*   subroutine  write sector
*
*   input   : a0.l = floppy device address
*             a3.l = start ram address
*             d1.l = sector   = track*numsec+sector-1
*   output : d7.w = status
*
*****************************************************************
*
wtsec:
```

```
        movem.l d0-d6/a4,-(a7)
        bsr     calpar
        tst.w   d7
    beq.s   1f
    move.w  #0x8002,d7
        bra.s   wtsec9          *  lsn out of range
1:
        bsr     seek
        tst.w   d7
    beq.s   1f
    move.w  #0x8001,d7
        bra.s   wtsec9          *  seek error
1:
    move.b  #0xa8,d5        *  write command
        lsl.b   #1,d2
        or.b    d2,d5           *  update side info
        move.l  #0,d1
        move.l  #1,d2
        move.l  a0,a4
        add.l   #6,a4           *  pointer to FDC data reg
        move.b  d3,4(a0)        *  set sector number
    jsr     _spl7           *  disable interupt
        move.b  d5,(a0)         *  write FDC command
        moveq   #8,d6
        bsr     delaycy         *  let FDC reg settle
        bra.s   wtloop
wtloop1:
        move.b  (a3)+,(a4)      *  write one byte
wtloop:
        move.b  (a0),d0         *  read FDC status
        btst    d1,d0           *  still busy ?
        beq.s   wtsec1          *  completed
        btst    d2,d0           *  data request ?
        bne     wtloop1
        bra     wtloop
wtsec1:
    jsr _spl0           *  enable interupt
        clr.l   d7
        move.b  (a0),d7         *  update return status
wtsec9:
        movem.l (a7)+,d0-d6/a4
        rts

**************************************
*
*       Seek to track
*
*       Entry: a0.l => floppy base address
*              d1.b =  track number
*       Exit : d7.w =  status
*
seek:
        movem.l d0/d6,-(a7)
        move.b  2(a0),d0        *  read current track
        cmp.b   d0,d1           *  same ?
```

```
        bne.s   seek1
        clr.l   d7
        bra.s   seek9
seek1:
        move.b  d1,6(a0)        *  update track info
    moveq   #0x10,d0        * seek command

seekx:                              * entry point for restore routine
    move.l  #500,d6         * delay 1 ms
    bsr delaycy
    move.b  latch,d6
    or.w    #s8inch,d6      * increase to 2Mhz for stepping
    move.w  d6,ladisp(a0)
    move.b  d0,(a0)         * write command to FDC (seek or restore)
        moveq   #8,d6
        bsr     delaycy       *  wait for FDC to settle
seek2:
        move.b  (a0),d0       *  read FDC status
        btst    #0,d0         *  busy ?
        bne     seek2         *  wait till completed
    move.b  latch,d6
    or.w    #0x0c00,d6
    move.w  d6,ladisp(a0)   * reduce to 1Mhz
        clr.l   d7
        move.b  d0,d7
        and.b   #0x98,d7      *  mask some bits
        beq.s   seek9
        move.w  #0x8001,d7    *  mark seek error
seek9:
        movem.l (a7)+,d0/d6
        rts
*
*
*****************************************
*
*         Calculate parameters
*
*         Entry: d1.l =  logical sector number
*
*         Exit : d1.b =  track number
*                d2.b =  head number
*                d3.b =  sector number
*                d7.w =  status
*
calpar:
        clr.l   d3
        move.w  #1600,d3
        cmp.l   d3,d1
        blt.s   calpar1
        move.l  #-1,d7        *  lsn out of range
        rts
calpar1:
        move.l  d1,d3         *  d3 = lsn
        moveq   #10,d7
        divu    d7,d3         *  d3 = cyl = lsn / sct pr trk
```

```
        move.w  d3,d1           *  d1 = cyl
        swap    d3
        add.w   #1,d3           *  d3 = sector
        and.l   #0xff,d3
        and.l   #0xff,d1
        clr.l   d2              *  d2 = head
        lsr.w   #1,d1           *  d1 = track = cyl / 2
        bcc.s   calpar2
        add.l   #1,d2           *  update head
calpar2:
        clr.l   d7
        rts
*
*
*****************************************
*
*    subroutine  delaycyclesequences
*       (no cycles = 9*d1+9)
*
*    input   : d6.l = delay
*
*****************************************
*
delaycy:
        subq.l  #1,d6
        bne.s   delaycy
        rts
*
*
        .even
        .data
drvno:  .byte   0
latch:  .byte   0
drvtbl: .long   0
    .long   0
```

A U T O L O G

av Per-Otto Olsson

När man skall kommunicera med en dator via telefon och modem så behövs det nästa alltid någon form av påloggning för att tala om att man är behörig. Dessa påloggningssekvenser är olika för olika datorer. Om man har kommunikation med många olika datorer så ör det en del saker man måste hålla i minnet.

AUTOLOG är en subrutin som kan hjälpa till att göra en automatisk påloggning. Subrutinen läggs in det program som normalt sköter uppringning och kommunikation. AUTOLOG skall ligga mellan uppringningen och kommunikationen. Den händige kan säkert utveckla sökrutiner så att man bara behöver ange t.ex "PD68" och så ringer datorn upp och loggar på automatiskt.

AUTOLOG tar en textsträng och utför de order som finns inlagda. Strängen består av valfri text, autopåloggningssekvens och valfri text. Den valfria texten kan användas för att göra sökningar eller innehålla de telefonnummer som man skall ringa. Autopåloggningssekvensen finns inom "<" och ">".

Kommandobeskrivning

<	$3C	Inleder påloggningssekvens. Sänds inte.
/hh	$2F	Initiera modemACIA (MC6850) med hexvärde hh.
=n	$3D	Vänta n sekunder och initiera sedan modemACIA:n med defaultvärde eller värde som getts i kommandot /hh.
?x	$3F	Vänta på tecknet "x" från datorn.
Uc	$5E	Sänd tecknet "c" som styrtecken.
!x	$21	Sänd tecknet "x". Används om man skall sända tecknen för kommandokod.
text		Sänd texten.
>	$3E	Avslutar påloggningssekvensen. Sänds inte.
CR	$0D	Avslutar påloggningssekvens. Sänds inte.

Exempel

PD68</01=1UM?:=1PER-OTTOUM?:=1MINKODUM>klubbdatorn

Förklaring

/01 initierar ACIA:n med $01.

=1 väntar 1 sekund.

UM sänder "Control-M" d.v.s. vagnretur, $0D.

?: väntar på kolon från datorn.

=1 väntar 1 sekund.

PER-OTTO Texten som sänds till datorn.

Resten av kommandona i strängen har redan presenterats.

I väntkommandot "=" så görs en reset och initiering av ACIA:n. Varför då? När man skall ange vilket tecken som man skall vänta på så bör detta tecken vara unikt. Går inte det att uppfylla så får man vänta på ett annat tecken och sedan räkna ut hur lång tid det tar innan datorn har "pratat" klart. En del datorer tycker inte om när man "pratar i mun" på dom, speciellt inte vid påloggning. De tecken som tagits emot under tiden är ointressanta men har förorsakat indikeringar om fel i ACIA:ns statusregister. Dessa indikeringar tas bort genom en reset.

Exempel (något krystat)

<ÜM?V=3HEJSANÜM>

Du "väcker" datorn med vagnretur och den svarar

VÄLKOMMEN TILL TJUSIGA DATORN.
ANGE LÖSEN

Du väntar sedan på "V" i VÄLKOMMEN och väntar sedan ytterligare
3 sekunder så att resten av texten hinner tas emot innan du sänder lösen "HEJSAN" och en vagnretur.

```
* A U T O T E S T
* Testa AUTOLOG

* AUTOTEST är ett kommando
* med följande utseende:

* AUTOLOG < autologonsträng >
* Ex. AUTOLOG </15ÜM??PER-OTTOÜM??OLSSONÜM>

* Innan man använder AUTOTEST måste
* man ladda in AUTOLOG.BIN med
* GET AUTOLOG.

              CD03   WARMS    EQU     $CD03
              C080   LINBUF   EQU     $C080

  C100                        ORG     $C100
  C100 8E    C080             LDX     #LINBUF
  C103 9D    00               JSR     0
  C105 7E    CD03             JMP     WARMS

                              END     $C100

0 ERROR(S) DETECTED
```

```
                * A U T O L O G
                * Subroutine to make auto logon
                * to a remote computer
                * V 1.0, 1985-10-25
                * by Per-Otto Olsson

E008    MDMADR  EQU     $E008       Computer dependent
E008    MDMCOM  EQU     MDMADR
E009    MDMDAT  EQU     MDMADR+1

2710    T100MS  EQU     10000       Delay 100 ms at 1MHz
000D    CR      EQU     $D
0003    RESET   EQU     3
0015    DFAULT  EQU     $15

                * INPUT:  X=address to auto logon string
                * OUTPUT: X  destroyed
                *         CC destroyed
                *         other registeres saved

0000                    ORG     0           Or some other place!

                0000    IN      EQU     *
0000 BF 00E9            STX     TEMP        Save X-register
0003 34 02              PSHA
0005 34 04              PSHB

                0007    L0      EQU     *
0007 8D 2E              BSR     GET         Get char from string
0009 81 3C              CMPA    #'<         START OF AUTOLOGON
000B 26 FA              BNE     L0          NOT YET
000D 8D 33              BSR     INZMDM

                000F    L1      EQU     *
000F 8D 26              BSR     GET         Get char from string
0011 81 2F              CMPA    #'/
0013 27 59              BEQ     INIT
0015 81 3F              CMPA    #'?
0017 27 3C              BEQ     WAIT        WAIT FOR CHARACTER
0019 81 5E              CMPA    #'U
001B 27 48              BEQ     CONTRL      SEND CONTROL CHARACTER
001D 81 21              CMPA    #'!
001F 27 2D              BEQ     SND         SEND NEXT CHARACTER
0021 81 3D              CMPA    #'=
0023 27 54              BEQ     DELAY       DELAY n seconds
0025 81 3E              CMPA    #'>
0027 27 09              BEQ     LOGEND      END OF LOGON
0029 81 0D              CMPA    #CR
002B 27 05              BEQ     LOGEND      END OF LOGON

                002D    SEND    EQU     *
002D BD 00DD            JSR     OUTMDM      SEND CHAR TO MODEM
0030 20 DD              BRA     L1

                0032    LOGEND  EQU     *
0032 35 04              PULB
0034 35 02              PULA
0036 39                 RTS

                * Get character from string
                0037    GET     EQU     *
0037 BE 00E9            LDX     TEMP
```

```
003A A6    84              LDAA    0,X
003C 30    01              INX
003E BF    00E9            STX     TEMP
0041 39                    RTS

                   * Reset and initialize ACIA
              0042 INZMDM  EQU     *
0042 C6    03              LDAB    #RESET
0044 F7    E008            STAB    MDMCOM
0047 F6    00E8            LDAB    MCTRL
004A F7    E008            STAB    MDMCOM
004D 39                    RTS

                   * Send next character to Modem
              004E SND     EQU     *
 004E 8D   E7              BSR     GET         Get char from string
 0050 BD   00DD            JSR     OUTMDM
 0053 20   BA              BRA     L1

                   * Wait for specified character
              0055 WAIT    EQU     *
 0055 8D   E0              BSR     GET         Get char from string
 0057 1F   894D            TAB
              005A WAIT1   EQU     *
>005A BD   00D2            JSR     INMDM       GET CHAR FROM MODEM
 005D 34   04 A1E0         CBA                 CHECK FOR MATCH
 0061 26   F7              BNE     WAIT1       NOT THE RIGHT ONE
 0063 20   AA              BRA     L1          THIS ONE IS OK

                   * Send next character as control character
              0065 CONTRL  EQU     *
 0065 8D   D0              BSR     GET         Get char from string
 0067 84   1F              ANDA    #$1F        MAKE CONTROL CHARACTER
>0069 BD   00DD            JSR     OUTMDM      SEND TO MODEM
 006C 20   A1              BRA     L1

                   * Initialize modemACIA
              006E INIT    EQU     *
 006E 8D   43              BSR     HEXGET      Get value
 0070 25   03              BCS     INIT1       Not hex, use old value
 0072 B7   00E8            STAA    MCTRL
 0075 8D   CB    INIT1     BSR     INZMDM
 0077 20   96              BRA     L1

                   * Delay 1-9 seconds
              0079 DELAY   EQU     *
 0079 8D   BC              BSR     GET         Get char from string
 007B 81   30              CMPA    #'0
 007D 23   1A              BLS     DEL1
 007F 81   39              CMPA    #'9
 0081 22   16              BHI     DEL1
 0083 84   0F              ANDA    #$0F

                   * Delay 1 second (at 1MHz)
              0085 DELAY1  EQU     *
0085 C6    0A              LDAB    #10         10*100 ms

                   * Delay 100 ms (at 1MHz)
              0087 DELAY2  EQU     *
0087 8E    2710            LDX     #T100MS

                   * Delay loop
              008A DELAY3  EQU     *
```

```
008A 12                      NOP
008B 30   1F               DEX
008D 26   FB               BNE     DELAY3

008F 5A                    DECB
0090 26   F5               BNE     DELAY2

0092 4A                    DECA
0093 26   F0               BNE     DELAY1     decrement seconds
                    * End of time out
0095 8D   AB               BSR     INZMDM     Initialize modemACIA
0097 0E   0F               JMP     L1

               0099 DEL1    EQU     *
0099 86   01               LDAA    #1
009B 20   E8               BRA     DELAY1

                    * Check if char is hex
               009D HEXCHK  EQU     *
009D 80   47               SUBA    #$47
009F 2A   0F               BPL     HEXNOK
00A1 8B   06               ADDA    #6
00A3 2A   04               BPL     HEXCH1
00A5 8B   07               ADDA    #7
00A7 2A   07               BPL     HEXNOK
00A9 8B   0A        HEXCH1 ADDA    #10
00AB 2B   03               BMI     HEXNOK
00AD 1C   FE               CLC                Char was hex
00AF 39                    RTS
00B0 1A   01        HEXNOK SEC                 Char was not hex
00B2 39                    RTS

                    * Get hex char from string
               00B3 HEXGET  EQU     *
00B3 8D   82               BSR     GET        Get char from string
00B5 8D   E6               BSR     HEXCHK
00B7 25   14               BCS     HEXGE8
00B9 48                    ASLA
00BA 48                    ASLA
00BB 48                    ASLA
00BC 48                    ASLA
00BD 1F   894D             TAB
00C0 9D   37               JSR     GET        Get char from string
00C2 8D   D9               BSR     HEXCHK
00C4 25   09               BCS     HEXGE9
00C6 34   04 ABE0          ABA
00CA 1C   FE               CLC
00CC 39                    RTS
00CD 9D   37        HEXGE8 JSR     GET        Skip next char
00CF 1A   01        HEXGE9 SEC
00D1 39                    RTS

                    * Input char from modem
                  00D2 INMDM    EQU     *
00D2 B6   E008             LDAA    MDMCOM
00D5 85   01               BITA    #1
00D7 27   F9               BEQ     INMDM
00D9 B6   E009             LDAA    MDMDAT
00DC 39                    RTS

                    * Output char to modem
                  00DD OUTMDM   EQU     *
00DD F6   E008             LDAB    MDMCOM
00E0 C5   02               BITB    #2
```

```
00E2 27   F9              BEQ    OUTMDM
00E4 B7   E009            STAA   MDMDAT
00E7 39                   RTS


00E8 15           MCTRL   FCB    DFAULT
00E9 0000         TEMP    FDB    0

                          END

0 ERROR(S) DETECTED
```

TILL SALU:

SWTPC-dator-MULTIUSER
Dubbla 8 tums drivar - dubbelsidiga - dubbel packning
TEC-70 terminal
Centronics printer
Winchester SWTPC felfri
Winchester SWTPC defekt

Ovanstående utrustning säljes komplett eller i delar - med fulltsändig uppsättning av manualer och är alldeles för kraftfull för ett förtag som varvar ner...

Ring Ulf Knöppel tel 08-610119 dagtid eller 08-619001 - kvällstid.

SÄLJES

1 MP-R eprom-prog	75:-
1 MP-N med Pilot på disk (5,8)	200:-
1 TC-3 kassettinterface JPC	75:-
1 AD-16 A/D-omv. JPC	75:-
2 tomma 64 K minneskort för 6116 (el 2716) med beskrivning per st.	200:-

KÖPES elBYTES

68B09E, 6883, WD2123 och μPD7220A

Pehr Tham, 031- 11 13 41 under dagtid.

```
00001                                   NAM   VT100 filter
00002
00003          * The following program is a filter wich converts VT100 codes to you
00004          * terminals codes. You have to adapt it to fit your terminal.
00005          *
00006          * Use like: OS9:<your terminalprogram> ! vt100
00007          *
00008          * Carl Nordin, Lidköping
00009          *
00010
00011   0001             stdout   EQU   $01
00012
00013   0006             F$Exit   EQU   $06
00014   0089             I$Read   EQU   $89
00015   008A             I$Write  EQU   $8A
00016
00017   00D3             E$Eof    EQU   211
00018
00019   0010             Prog     EQU   $10
00020   0001             Bin      EQU   $01
00021   0080             Shar     EQU   $80
00022   0001             Rev      EQU   $01
00023
00024   001B             esc      EQU   $1B
00025
00026   0000 87CD0146            MOD   Size,Name,Prog+Bin,Shar+Rev,Exec,Storage
00027
00028   000D 56543130    Name     FCS   "VT100"
00029
00030          * HW1 strings
00031
00032   0012 1B          inverse  FCB   esc         Set inverse video
00033   0013 2B3530              FCC   "+50"
00034   0004             invlen   EQU   *-inverse
00035
00036   0016 1B          normal   FCB   esc         Set normal video
00037   0017 2D3530              FCC   "-50"
00038   0004             norlen   EQU   *-normal
00039
00040   001A 1B          curpos   FCB   esc         Start of curpos sequence
00041   001B 3D                  FCC   "="
00042   0002             curplen  EQU   *-curpos
00043
00044   001C 1B          eraline  FCB   esc         Erase to end of line
00045   001D 4B                  FCC   "K"
00046   0002             erallen  EQU   *-eraline
00047
00048   001E 1B          erascr   FCB   esc         Erase to end of screen
00049   001F 4A                  FCC   "J"
00050   0002             eraslen  EQU   *-erascr
00051
00052   0020 1B          eraall   FCB   esc         Erase entire screen
00053   0021 45                  FCC   "E"
00054   0002             eraalen  EQU   *-eraall
00055
00056 D 0000             escflag  RMB   1           Got an escape flag
00057 D 0001             escseqfl RMB   1           Esc sequence (<ESC>A)
00058 D 0002             buffer   RMB   10          Buffer for esc sequence
00059 D 000C             buffp    RMB   2           Pointer into buffer
00060 D 000E             count    RMB   1           Number of characters in buffer
```

```
00061
00062   00C8            Storage EQU   200
00063
00064   0022 6FC4       Exec    clr   escflag,U
00065   0024 6F41               clr   escseqfl,U
00066   0026 6F4E               clr   count,U
00067   0028 3042               leax  buffer,U
00068   002A AF4C               stx   buffp,U
00069
00070
00071   002C 1700F9     loop    lbsr  getchar
00072   002F 0D00               tst   escflag    If an esc has been recieved th
00073   0031 2615               bne   escseq
00074
00075   0033 811B               cmpa  #esc       If not esc then print it
00076   0035 260C               bne   passch
00077
00078   0037 6CC4               inc   escflag,U  Set esc flag
00079   0039 AE4C               ldx   buffp,U
00080   003B A780               sta   ,X+
00081   003D AF4C               stx   buffp,U
00082   003F 6C4E               inc   count,U
00083   0041 20E9               bra   loop       Wait for the sequence
00084
00085   0043 1700D3     passch  lbsr  putchar    Pass character on
00086   0046 20E4               bra   loop
00087
00088   0048 AE4C       escseq  ldx   buffp,U    Save sequence
00089   004A A780               sta   ,X+
00090   004C AF4C               stx   buffp,U
00091   004E 6C4E               inc   count,U
00092
00093   0050 6D41               tst   escseqfl,U
00094   0052 2606               bne   seqend
00095
00096   0054 8130               cmpa  #'0        Check for termination of seque
00097   0056 25D4               blo   loop       0 <= x <= ~
00098   0058 2004               bra   escseq1
00099
00100   005A 8140       seqend  cmpa  #'@        Check for termination
00101   005C 25CE               blo   loop       @ <= x <= ~
00102
00103   005E 817F       escseq1 cmpa  #$7F
00104   0060 22CA               bhi   loop
00105
00106   0062 C602               ldb   #2         If two characters in buffer th
00107   0064 E14E               cmpb  count,U    for "Ä" CSI
00108   0066 2608               bne   term
00109
00110   0068 815B               cmpa  #'Ä
00111   006A 2604               bne   term
00112
00113   006C 6C41               inc   escseqfl,U Set flag for CSI
00114   006E 20BC               bra   loop
00115
00116        * Check terminator to determine what kind of action to take
00117
00118   0070 6D41       term    tst   escseqfl,U Check for CSI
00119   0072 2602               bne   csi
00120   0074 2014               bra   done
```

```
00121
00122                * CSI sequences
00123
00124  0076 3042      csi     leax  buffer,U   Prepare for routines
00125  0078 3002              leax  2,X         Point to parameter
00126
00127  007A 816D              cmpa  #'m         Attributes
00128  007C 270E              beq   attrib
00129  007E 8148              cmpa  #'H         Cursor position
00130  0080 2724              beq   cursorp
00131  0082 814B              cmpa  #'K         Erase line
00132  0084 2744              beq   eline
00133  0086 814A              cmpa  #'J         Erase screen
00134  0088 274C              beq   escr
00135
00136  008A 2096      done    bra   Exec
00137
00138           * Interpret CSI seqences
00139
00140  008C 8D62      attrib  bsr   getnum      Get number
00141  008E 270A              beq   noatt       Set no attributes
00142
00143  0090 308DFF7E          leax  inverse,PCR
00144  0094 108E0004          ldy   #invlen
00145  0098 2008              bra   att1
00146
00147  009A 308DFF78  noatt    leax  normal,PCR
00148  009E 108E0004          ldy   #norlen
00149
00150  00A2 8D6F      att1    bsr   putstr
00151  00A4 20E4              bra   done
00152
00153  00A6 3410      cursorp pshs  X
00154  00A8 308DFF6E          leax  curpos,PCR Start of sequence
00155  00AC 108E0002          ldy   #curplen
00156  00B0 8D61              bsr   putstr
00157  00B2 3510              puls  X
00158  00B4 8D3A              bsr   getnum      Get Y
00159  00B6 2702              beq   cur1        No parameter
00160  00B8 8001              suba  #1          1 - 24 -> 0 - 23
00161  00BA 8B20      cur1    adda  #$20
00162  00BC 8D5B              bsr   putchar
00163
00164  00BE 8D30              bsr   getnum
00165  00C0 2702              beq   cur2
00166  00C2 8001              suba  #1
00167  00C4 8B20      cur2    adda  #$20
00168  00C6 8D51              bsr   putchar
00169  00C8 20C0              bra   done
00170
00171  00CA 308DFF4E  eline    leax  eraline,PCR
00172  00CE 108E0002          ldy   #erallen
00173  00D2 8D3F              bsr   putstr
00174  00D4 20B4              bra   done
00175
00176  00D6 8D18      escr    bsr   getnum
00177  00D8 260A              bne   escr1
00178
00179  00DA 308DFF40          leax  erascr,PCR
00180  00DE 108E0002          ldy   #eraslen
```

```
00181   00E2 2008                bra   escr2
00182
00183   00E4 308DFF38  escr1     leax  eraall,PCR
00184   00E8 108E0002            ldy   #eraalen
00185
00186   00EC 8D25      escr2     bsr   putstr
00187   00EE 209A                bra   done
00188
00189        * Get a number, if no number exists return 0
00190
00191   00F0 6FE2      getnum    clr   ,-S          Acc
00192
00193   00F2 A680      getnum1   lda   ,X+          Get ASCII
00194   00F4 8130                cmpa  #'0
00195   00F6 2517                blo   getnumx
00196   00F8 8139                cmpa  #'9
00197   00FA 2213                bhi   getnumx
00198   00FC 8030                suba  #'0          To binary
00199   00FE E6E4                ldb   ,S           Multiply old value by 10
00200   0100 58                  aslb
00201   0101 3404                pshs  B
00202   0103 58                  aslb
00203   0104 58                  aslb
00204   0105 EBE0                addb  ,S+
00205   0107 3404                pshs  B
00206   0109 ABE0                adda  ,S+
00207   010B A7E4                sta   ,S
00208   010D 20E3                bra   getnum1
00209
00210   010F 6DE4      getnumx   tst   ,S
00211   0111 3582                puls  A,PC
00212
00213
00214   0113 8601      putstr    lda   #stdout      Print a string
00215   0115 103F8A              OS9   I$Write
00216   0118 39                  rts
00217
00218   0119 3412      putchar   pshs  A,X          Print one character
00219   011B 1F41                tfr   S,X
00220   011D 108E0001            ldy   #1
00221   0121 8601                lda   #stdout
00222   0123 103F8A              OS9   I$Write
00223   0126 3592                puls  A,X,PC
00224
00225   0128 327F      getchar   leas  -1,S         Local storage
00226   012A 1F41                tfr   S,X          Read one character
00227   012C 108E0001            ldy   #1
00228   0130 4F                  clra
00229   0131 103F89              OS9   I$Read
00230   0134 2404                bcc   getch1
00231
00232   0136 C1D3                cmpb  #E$Eof       Exit when done
00233   0138 2705                beq   exit
00234
00235   013A 3502      getch1    puls  A
00236   013C 847F                anda  #$7F         Strip parity
00237   013E 39                  rts
00238
00239   013F 5F        exit      clrb
00240   0140 103F06              OS9   F$Exit
```

```
Microware OS-9 Assembler 2.1   10/30/85 16:38:31                Page 005
VT100 filter -

00241
00242   0143 5D99CC                EMOD
00243
00244   0146               Size    EQU   *
00245

00000 error(s)
00000 warning(s)
$0146 00326 program bytes generated
$000F 00015 data bytes allocated
$038B 00907 bytes used for symbols
```

OS9/Flex system säljes:

Dator: baserad på Primaldatas PD2000
omkopplingsbar mellan FLEX och OS9
2 serie portar
1 parallell port (Centronics)

Skivminne: 1 st. DS DD 40
1 st. ES DD 40

Terminal: ADDS Viewpoint

Printer: Epson MX80 Mk.III

Viss programvara ingår.

Pris: 6.000:-

SINCLAIR QL säljes:

Dator: QL (MC68008)

två mikro-drivrar och
RS232 kabel ingår.
En del programvara ingår:
Quill (ord-behandling)
Abacus (kalkyl)
Archive (databas)
Easel (grafik, statistiska kurvor)
Assembler development kit
BCPL development kit

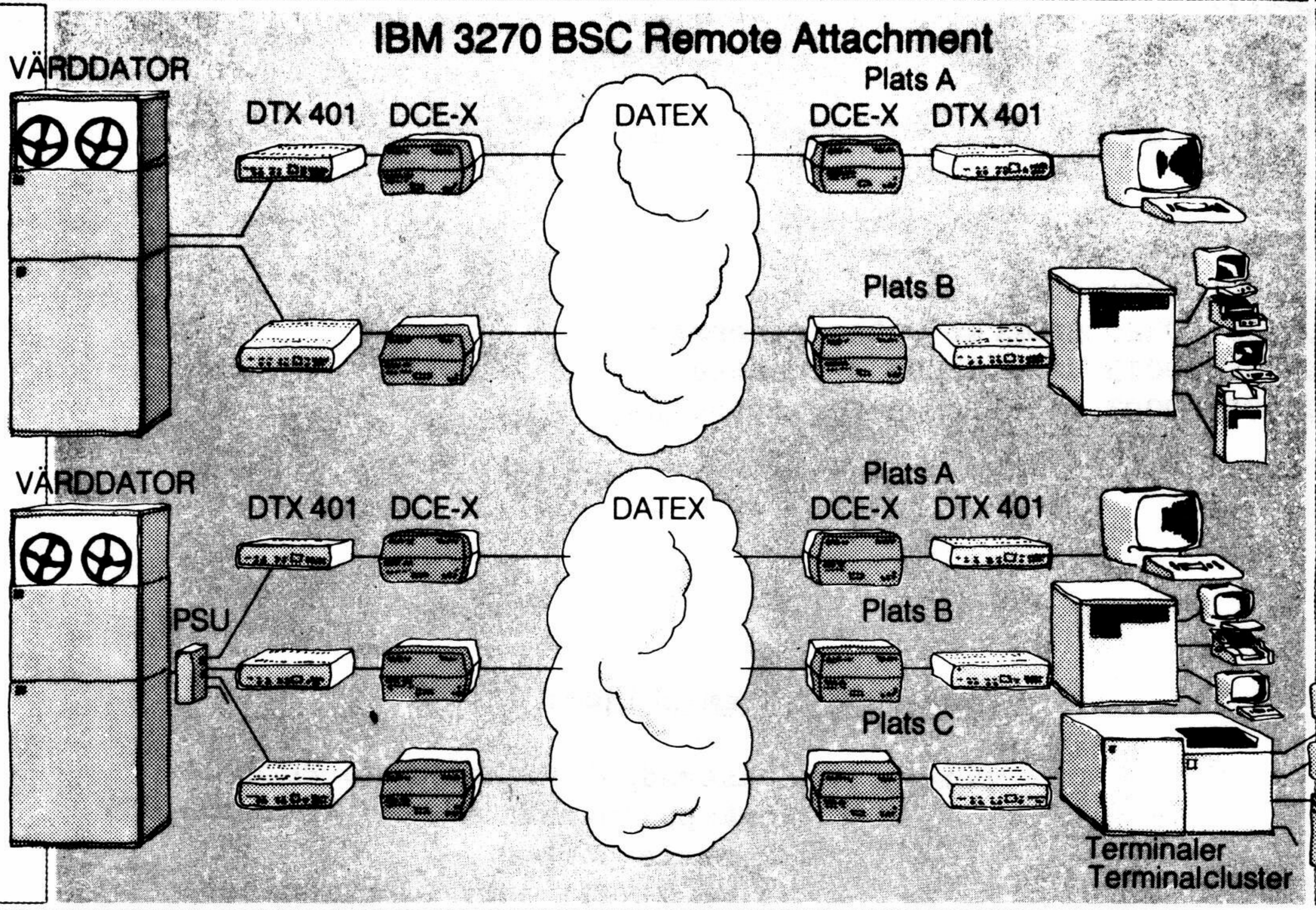
plus en del inknappat.

div. litteratur medföljer.

Pris: 3.000:-

Thomas Bergstam (1031) Tel. (08) 99 75 68.

DATEXPORT



Tekniska data för DTX 401

Protokoll
IBM 3270 BSC, 2780/3780, EBCDIC eller ASCII-kod.
Selektiv adressering (multidrop).
Full eller halv duplex,simulering.
Datahastighet: 600, 2400, 4800 eller 9600 b/s.

Front
Tryckknappar för test och loop-funktioner.
Lysdioder för REDO, DATAFAS och NÄTSPÄNNING.
Sifferindikering för diagnos av Datexnätet och uppkopplare.
CALL PROGRESS-signaler från Datexnätet visas.

Internt
Inställningar för Datexnummer (el direktanrop), protokollstyp, dator/terminal mode, clusteradress, pollfrekvens, tidsövervakning, nerkoppling, antal anropsförsök och tid mellan anrop.
Full/halv duplex, bittakt från DCE, DTE eller internt.
Siffervärden programmeras med vridomkopplare och lagras i batteriuppbackat minne.
På sifferindikator kan inställda värden kontrolleras.

Bakstycke
Anslutning mot Datex DCE (X.21):
Balanserat snitt enligt X.27 (V.11). Använda kretsar: G, T, R, C, I och S.
Kontaktdon 15-pol, ISO4903. Kabel till DCE-X medföljer.

Anslutning mot DTE, dator eller terminal:
Funktion enligt V.24/V.28 (RS232-modemsnitt).
Använda kretsar: 102, 103, 104, 105, 106, 107, 108/2, 109, 113/114, och 115.
Kontaktdon 25-pol, ISO 2110.

Nätanslutning: 230V, 50 Hz, 13W
Storlek: 260x54x280 mm (bxhxd)
Vikt: 1,5 kg

Tillbehör:
PSU 400, linjedelare för multidrop, 4 linjer.

Dextraferm startades för 14 år sedan och har idag marknadens bredaste produktprogram för datakommunikation och är marknadsledande som leverantör av anpassningar för Datex-nätet.

DEXTRAFERM

Box 17011 104 62 STOCKHOLM Tel: 08-44 03 50

** Ron Cain's small-C V1.1 **

Flex version 2.1, 13 Aug 1982

By S. Stepanoff

Som synes har klubben en C-kompilator för oss som fortfarande åker 8-bitarsmaskiner. Denna C gör skäl för namnet tiny C då den saknar t ex long integer och stdio.h. Flyttalsoperationer finns inte heller. Trots dessa brister ger compilatorn oss en möjlighet att få prova detta nu så populära programmeringsspråk.
Som ni kan se av artikelrubriken är kompilatorn skriven för 6800 och Flex 2.
Misströsta inte. Tack vare den eminente medlemmen Lars-Erik Lundström så finns det även en version för 6809 och Flex 9. Han har helt enkelt skrivit om den del av compilatorn som är skriven i assembler + en del andra ändringar.
Lika bra att erkänna att jag inte kan programera. Därav följer att jag endast har kunnat prova de demo-program som följer med. Dessa fungerar alldeles utmärkt utom ett program som heter Life. Detta program är tydligen skrivet för en mer avancerad C-version.
För att få en uppfattning om hur pseudo-koden fungerar kan man lämpligen titta på 'ccint.txt'.

Följande program finns på systemdisketten i Flex 9-versionen:
CC.CMD
CCC.H
FSIZE.CMD
FLEXPTRS.TXT
MAP.CMD
PRTLIB.ASM
PRTLIB.BIN

I versionen för Flex 2 används naturigtvis hela disken 'COMPILER'.

Bruksanvisning för att skapa en körklar fil med extension '.CMD'.

1. Sätt din systemdisk i drive 0.
   Disken med programmet i drive 1.
   Skriv nu 'cc' och kompileringen startar med att du får en del frågor. Det är bara att svara på dessa. När 'UTFIL' efterfrågas savara du med >filnamn.asm<. Nästa fråga lyder 'INFIL'. Om du då lämpligen kör ett av demo-programmen så svara t ex 'ECHO.C'. Efter en stund frågas efter 'INFIL' igen. Detta för att du efter 10 års idogt tränande skall kunna länka in fler program. Eftersom du inte vet ett enda dugg vad som skall länkas in föreslår jag att du helt enkelt trycker CR.

2. Om och när allt fungerat tillfredsställande vid kompileringen har du nu begåvats med en fil som har extension '.ASM'. Denna fil skall nu assembleras.

3. Efter assembleringen gör du 'FSIZE ECHO' och noterar högsta laddadress.

4. Gör nu 'GET ECHO' och 'GET CCINT'.

5. Om du inte redan har givit upp fullföljd med:
   'SAVE ECHO.CMD 0 högsta laddadress 0'.

Äntligen är du vid det hägrande målet. Du är innehavare av en fil med extension '.CMD'. Detta är den körklara filen. LYCKA TILL.

Beställning av disketter enligt de vanliga rutinerna. ULF HOLM

```
                * Här kommer nu till slut programlistningen
                * till terminal CÅ-790.
                * OBS !! Den som vill göra ändringar i programmet
                * gör detta på egen risk. Skohornsteknik har används
                * för att få in programmet i 2k Eprom.
                *
                * Videoterminalprogram för
                * Intelligent Videoterminal CÅ-790A
                * 24 rader * 40 tecken
                *
                * COPYRIGHT CÅ-ELEKTRONIK AB
                *          BOX 2010
                *          135 02 TYRESÖ
                *
                * 1980-10-09 16:30
                *
                * Vid start-up  visas  CÅ-790.x
                * x= baudrate
                * 0=    50 baud  4=   150 baud  8=  1800 baud  <=  4800 baud
                * 1=    75 baud  5=   300 baud  9=  2000 baud  ==  7200 baud
                * 2=   110 baud  6=   600 baud  :=  2400 baud  >=  9600 baud
                * 3=   135 baud  7=  1200 baud  ;=  3600 baud  ?= 19200 baud
                *
                * Följande kontrolltecken behandlas direkt i ASCII-mode
                * från serieingången RS-232-C
                * De måste föregås av ESCAPE (¤1B) i TEXT-mode
                *
0007            CTLG    EQU     7           BELL-KOD
0008            CTLH    EQU     8           BACKSPACE
000A            LF      EQU     ¤A          RADMATNING MED EV. SCROLLNING
000D            CR      EQU     ¤D          VAGNRETUR
0011            DC1     EQU     ¤11         READER ON Starta bandspelare, avspelning
0012            DC2     EQU     ¤12         PUNCH ON Starta bandspelare, inspelning
0013            DC3     EQU     ¤13         READER OFF
0014            DC4     EQU     ¤14         PUNCH OFF
001B            ESC     EQU     ¤1B         SPECIAL Se nedan
                *
                * DC1-DC4 Förutsätter att TAPE ON lysdioden är tänd
                * F1+Å ger TAPE ON, F1+Ö ger TAPE OFF från tangentbordet
                *
                * Följande kontrollkoder skall föregås av ESC (¤1B)
                * i både ASCII och TEXT-mode från RS-232-C ingången.
                *
0001            CTLA    EQU     1           PAGE MODE
0002            CTLB    EQU     2           TÄND CURSORN
0003            CTLC    EQU     3           CLEAR END OF BILD
0005            CTLE    EQU     5           RADERA HELA BILDEN + CURSOR HOME
0009            CTLI    EQU     9           CURSOR FRAMÅT
000B            CTLK    EQU     ¤B          CURSOR UPP
000C            CTLL    EQU     ¤C          CURSOR HOME
000E            CTLN    EQU     ¤E          PRINTER OFF
000F            CTLO    EQU     ¤F          SKIFTA TILL TELETEXTMODE
0010            CTLP    EQU     ¤10         PRINTER ON
0015            CTLU    EQU     ¤15         SLÄCK CURSORN
0016            CTLV    EQU     ¤16         RESTORE POINTER
0017            CTLW    EQU     ¤17         CURSOR NER
0019            CTLY    EQU     ¤19         SAVE POINTER
001A            CTLZ    EQU     ¤1A         RADERA FRÅN CURSOR TILL RADENS SLUT
                *
                * Koderna ¤20-37 anger från vilken rad scrollning skall ske
                * Ex. vis ESC (¤1B) + ¤23 = scrollning börjar på rad 3 (fjärde uppifrån)
```

```
62              *
63              * För cursorpositionering används två tecken efter ESC-kod.
64              * Det första tecknet är radnr och det andra position på raden.
65              * Radnr anges ¤00-¤17 med bit 6=1, dvs ¤40-¤57
66              * Posnr anges ¤00-¤27 med bit 6=1, dvs ¤40-¤67
67              *
68              * Kontrolltecknens funktion vid TEXT-mode :
69              *
70              * CTL A (01) RÖd alfanumerisk                CTL Q (¤11) Röd grafik
71              * CTL B (02) Grön alfanumerisk text          CTL R (¤12) Grön grafik
72              * CTL C (03) Gul alfanumerisk text           CTL S (¤13) Gul grafik
73              * CTL D (04) Blå alfanumerisk text          CTL T (¤14) Blå grafik
74              * CTL E (05) Violett alfanumerisk text      CTL U (¤15) Violett grafik
75              * CTL F (06) Cyan alfanumerisk text         CTL V (¤16) Cyan grafik
76              * CTL G (07) Vit alfanumerisk text          CTL W (¤17) Vit grafik
77              * CTL H (08) Start av blinkande tecken      CTL Y (¤19) Kontinuerlig grafik
78              * CTL I (09) Slut på blinkande tecken       CTL Z (¤1A) Separerad grafik
79              * CTL K -ENDAST FRÅN TANGENTBORDET- Dubbel teckenhöjd
80              * CTL L (¤C) Normal teckenhöjd                  CTL Ö (¤1C) Svart bakgrund
81              * CTL M (¤D) -ENDAST RS-232-INGÅNG- Dubbel teckenhöjd
82              * CTL N (¤E) Återgå till ASCII-mode             CTL Å (¤1D) Ny bakgrundsfärg
83              *                                              CTL Ü (¤1E) Bibehållen grafik
84              *                                              CTL+SHIFT O (¤1F) Skifta grafik
85              *
86              * Rubout-kod erhålles från Sverigebordet med CTRL < (¤7F)
87              *
88              * Funktionskoder från tangentbordet. Erhålles genom att
89              * trycka på F1-tangenten och den angivna tangenten
90              * eller för vissa funktioner F1 och CTRL och den angivna tangenten
91              *
92              * Tangent Motsv. hexkod. Funktion
93              *    A         C1          On line med full duplex
94              * CTRL A       81          On line med halv duplex
95              *    B         C2          Lokal utskrift
96              *    C         C3          Radera från cursor till radens slut
97              *    D         C4          Ta bort tabulatorstopp
98              *    E         C5          Radera hela bilden
99              *    F         C6          Flytta cursorn till nästa tabstopp
100             *    G         C7          Sänd den rad på vilken cursorn står,
101             *                           från vänsterkanten till cursorn.
102             *    H         C8          Cursor vänster
103             * CTRL H       88          Cursor vänster
104             *    I         C9          Cursor höger
105             * CTRL I       89          Cursor höger
106             *    J         CA          Sätt in ett tecken på cursorns plats,
107             *                           alla tecken till höger flyttas ett steg.
108             *    K         CB          Cursor upp
109             *    +         AB          Cursor upp
110             *    L         CC          Cursor home
111             *    M         CD          Ta bort den rad som cursorn står på
112             *    N         CE          Printer från
113             *    O         CF          Ta bort det tecken som cursorn står på
114             *    P         D0          Printer till
115             *    Q         D1          Sänd hela sidan till datorn
116             * CTRL+Q       91          Sänd hela sidan till KCS-modemet.
117             *    R         D2          Page mode
118             * CTRL+R       92          Fyll bilden med tecken från
119             *                           KCS-modemet.
120             *    S         D3          Sätt in en tabstopp
121             *    T         D4          Tvåradsscroll
122             *    U         D5          Normal fullsidesscroll
123             *    V         D6          Sätt in en ny rad där cursorn står
```

```
124             *    W       D7          Cursor ner
125             *    X       D8          Återställ normalparametrar (restart)
126             *    Y       D9          Halvsidesscroll
127             *    Å       DD          KCS-modem inkopplat
128             *    Ä       DB          Ta bort alla tabstoppar
129             *    Ö       DC          KCS-modem urkopplat
130             *    !       A1          Ingen paritet
131             *    "       A2          Jämn paritet
132             *    #       A3          Udda paritet
133             *    -       AD          Cursor ner
134             *    0       B0          50 baud (ej i version 4)
135             *    1       B1          75 baud (ej i version 4)
136             *    2       B2          110baud
137             *    3       B3          135 baud
138             *    4       B4          150 baud
139             *    5       B5          300 baud
140             *    6       B6          600 baud
141             *    7       B7          1200 baud
142             *    8       B8          1800 baud
143             *    9       B9          2000 baud
144             *    :       BA          2400 baud
145             *    ;       BB          3600 baud
146             *    <       BC          4800 baud
147             *    =       BD          7200 baud
148             *    >       BE          9600 baud
149             *    ?       BF          19200 baud
150             *  CTRL D    84          sändn 75 baud, mottagn. 1200 baud
151             *    ¤       A4          cursorn tänd
152             *    %       A5          cursorn släckt
153             *    É       C0          sänd från bildens början till cursorn
154             *
155             *
156  000F       SI      EQU     CTLO
157  000E       SO      EQU     ¤0E         SHIFT OUT: ASCII MODE
158  0020       SP      EQU     ¤20
159             *
160             * EQU-SATSER FÖR ADRESSANPASSNING
161             *
162  C000       BILD    EQU     ¤C000
163  D000       TPGM    EQU     ¤D000       PROGRAMMINNE 2 K
164  0000       TRAM    EQU     ¤0000       RAM
165  00C0       TVBHI   EQU     ¤C0         HÖGA HALVAN AV BILDADRESSEN
166             *
167             * PERIFERIKRETSAR
168             *
169  9000       PPIAA   EQU     ¤9000       PIA FÖR PRINTER
170  9001       PPIAAC  EQU     PPIAA+1
171  9002       PPIAB   EQU     PPIAA+2
172  9003       PPIABC  EQU     PPIAA+3
173             *
174  9000       PACIA   EQU     PPIAA       ACIA FÖR PRINTER
175  9001       PACIAD  EQU     PACIA+1     DITO DATAREG.
176             *
177  9100       PCIDR   EQU     ¤9100       DATAREGISTER 2651
178  9101       PCISR   EQU     PCIDR+1     STATUSREGISTER 2651
179  9102       PCIMR   EQU     PCIDR+2     MODE REGISTER 1 & 2
180  9103       PCICR   EQU     PCIDR+3     COMMAND REGISTER 2651
181             *
182  9200       PIATBD  EQU     ¤9200       TANGENTBORDSDATA
183  9201       PIATBC  EQU     PIATBD+1
184  9202       PIAKD   EQU     PIATBD+2    KONTROLLFUNKTIONER
185  9203       PIAKC   EQU     PIATBD+3    D:O CONTROLREG.
```

```
186                *
187  9300          KCIAC   EQU     ¤9300       KCS-MODEMETS ACIA KONTROLLREG
188  9301          KCIAD   EQU     KCIAC+1     D:O DATAREG
189                *
190  C000          TVB     EQU     BILD
191  00C3          TVBOHI  EQU     TVBHI+3
192  0098          TVBOLO  EQU     ¤98         LÅGA HALVAN AV TVR23
193  00C3          TVNDHI  EQU     TVBOHI
194  00C0          TVNDLO  EQU     ¤C0         LÅGA HALVAN AV TVBEND
195  C1E0          TVR12   EQU     TVB+480
196  C370          TVR22   EQU     TVB+880
197  C398          TVR23   EQU     TVB+¤398    FÖRSTA POS PÅ NEDERSTA RADEN
198  C3C0          TVBEND  EQU     TVB+¤3C0
199  0028          LENGD   EQU     ¤28
200  0085          SKIP1   EQU     ¤85
201  008C          SKIP2   EQU     ¤8C
202                *
203                *RAMCELLER
204                *FÖR DATALAGRING
205                *
206  0000                  ORG     TRAM
207  0000          PRBUF   RMB     40          PRINTERBUFFERT
208  0028          PRPOS   RMB     1
209  0029          PRPOSA  RMB     1
210  002A          PEKARE  RMB     2           PEKARE FÖR CURSORPOSITION
211  002C          XHI     RMB     1
212  002D          XLO     RMB     1
213  002E          RPCIM1  RMB     1           RAM-CELL MODEREG 1
214  002F          RPCIM2  RMB     1           RAM-CELL MODEREG 2
215  0030          INBUF   RMB     4           FYRA CELLER FÖR DATA UNDER IRQ
216  0034          ROLADR  RMB     2           STARTADRESS VID SCROLLNING
217  0036          HPOS    RMB     1           HORISONTALPOSITION (0-39) PÅ RADEN
218  0037          HPOS2   RMB     1           CELL VID INSERT/DELETE
219  0038          RADNR   RMB     1           NUMMER PÅ RAD FÖR SPEC. FUNKTION
220  0039          ADRESS  RMB     2
221  003B          ERRRAM  RMB     2           CELLER FÖR DELAY AV ERRORLED
222  003D          INPEK   RMB     2
223  003F          OUTPEK  RMB     2
224  0041          LOFLAG  RMB     1           0=ON LINE, >1=LOCAL
225  0042          TXTFLG  RMB     1           0=ASCII-TERMINALMODE, FF=TELETEXTMODE
226  0043          PAGFLG  RMB     1           0=SCROLL, >1=PAGE MODE
227  0044          SNKCFL  RMB     1           0=NORM, >1 = TX BILDEN TILL KCS
228  0045          CURFLG  RMB     1           0=CURSOR ON, >1=CURSOR OFF
229  0046          DUPFLG  RMB     1           0=FULL DUPLEX, >1=HALV DUPLEX
230  0047          TAPFLG  RMB     1           0=EJ DC1-4 RESPONSE, >0 KCS-AUTOMATIK
231  0048          KCDATA  RMB     1           SISTA DATABYTEN TILL KCS
232  0049          ESCFLG  RMB     1           FLAGGA FÖR ESCAPE-TECKEN
233  004A          PARFLG  RMB     1           FLAGGA FÖR PARITETSBIT
234  004B          PRTFLG  RMB     1           PRINTFLAGGA
235  004C          KOXFLG  RMB     1           0=X-KOORDINAT, >0=Y-KOORDINAT
236  004D          KOFLAG  RMB     1           FLAGGA FÖR XY-MODE
237  004E          KCRFLG  RMB     1           FLAGGA FÖR KCS PLAY (=RECEIVE)
238  004F          KCTFLG  RMB     1           FLAGGA FÖR KCS RECORD (=TRANSMIT)
239  0050          TAB01   RMB     39          TABULATORTABELL
240  0077          TAB39   EQU     *           SLUTET PÅ DITO
241  0077          HIADR   RMB     2           ADRESSCELL FÖR HEREIS-RUTIN
242  0079          TXPAUS  RMB     2           CELL FÖR POSITIONSRÄKNING-PAUS
243  C3FF          STACK   EQU     ¤C3FF
244                *
245                *PROGRAMSTART
246                *
247  D000                  ORG     TPGM
```

```
248                     *
249  D000 3E            PMR2    FCB     ¤3E         9600 BAUD INT CLK, DEFAULT
250                     *
251  D001 43            TYPNR   FCC     'CÅ-790-4'
252                     *
253  D009 FF FF                 FDB     ¤FFFF       PLATS FÖR ID
254                     *
255  D00B 8E C3FF       START   LDS     #STACK
256  D00E CE 0000               LDX     #0
257  D011 8D 7B                 BSR     CLRTA2      NOLLA RAM
258  D013 BD D284               JSR     ERASE
259                     *
260                     * SÄTT UPP PCI 2651
261                     *
262  D016 CE 9100               LDX     #¤9100
263  D019 A6 03                 LDA A   3,X         PCICR NOLLA PEKAREN FÖR MR1
264  D01B 86 4E                 LDA A   #¤4E        8+1 STOPPBIT INGEN PARITET
265  D01D A7 02                 STA A   2,X         PCIMR REG 1
266  D01F 97 2E                 STA A   RPCIM1      SPARA ÄVEN I RAM
267  D021 B6 D000               LDA A   PMR2
268  D024 A7 02                 STA A   2,X         PCIMR REG 2
269  D026 97 2F                 STA A   RPCIM2
270  D028 86 05                 LDA A   #¤5
271  D02A A7 03                 STA A   3,X         PCICR COMMAND REG
272                     *
273                     * SÄTT UPP PIAN
274                     *
275  D02C CE 9200               LDX     #¤9200
276  D02F 6F 01                 CLR     1,X         PIATBC NOLLA KONTROLLREG
277  D031 6F 00                 CLR     0,X         PIATBD PA0-7 INGÅNGAR
278  D033 6F 03                 CLR     3,X         PIAKC
279  D035 86 3D                 LDA A   #¤3D        CA1 STROBE,CA2 BELL NEG PULS
280  D037 A7 01                 STA A   1,X         PIATBC
281  D039 86 FF                 LDA A   #¤FF        PB0-7 UTGÅNGAR
282  D03B A7 02                 STA A   2,X         PIAKD
283  D03D C6 04                 LDA B   #4
284  D03F E7 03                 STA B   3,X         PIAKC
285  D041 A7 02                 STA A   2,X         PIAKD
286                     *
287                     * RESET KCS-ACIAN
288                     *
289  D043 5A                    DEC B
290  D044 F7 9300               STA B   KCIAC       RESET
291                     *
292                     * SÄTT UPP PRINTERPIAN/ACIAN
293                     *
294  D047 F7 9000               STA B   PACIA
295  D04A C6 11                 LDA B   #¤11        2 STOPPBITAR
296  D04C F7 9000               STA B   PACIA
297                     *
298  D04F CE C001       RESTA0  LDX     #TVB+1      BILDPOS
299  D052 8E D000               LDS     #TYPNR-1
300  D055 C6 08                 LDA B   #8
301  D057 32            RESTA2  PUL A
302  D058 A7 00                 STA A   0,X         LÄGG UT "CÅ-790.XX"
303  D05A 08                    INX
304  D05B 5A                    DEC B
305  D05C 26 F9                 BNE     RESTA2
306  D05E 96 2F         RESTA5  LDA A   RPC1M2      HÄMTA SPEEDVÄRDET
307  D060 A7 00                 STA A   0,X         AKTUELLT BAUDRATEVÄRDE
308  D062 8E C3FF       RESTAR  LDS     #STACK
309  D065 BD D278               JSR     TEVEBE
```

```
310  D068 DF 77          STX    HIADR     SPARA GILTIGT VÄRDE FÖR RESTORE-FUNKTION
311  D06A DF 34          STX    ROLADR    FULL SCROLL
312  D06C 86 FC          LDA A  #¤FC      NOLLA LED & KCS-STYRNING
313  D06E B7 9202        STA A  PIAKD
314  D071 BD D693        JSR    ONLINE
315  D074 8D 0C          BSR    NIX
316  D076 20 1F          BRA    VLOOP
317                *
318  D078 BD D2F9 DFAULT JSR    NOCUR
319  D07B CE 0041        LDX    #LOFLAG   NOLLA FÖR NORMALVÄRDEN
320  D07E 8D 0E          BSR    CLRTA2
321  D080 20 CD          BRA    RESTA0
322                *
323  D082 CE 0030 NIX    LDX    #INBUF    ÅTERSTÄLLARE
324  D085 DF 3D          STX    INPEK
325  D087 DF 3F          STX    OUTPEK
326  D089 0E             CLI
327  D08A 39             RTS
328                *
329                *
330  D08B CE 0050 CLRTAB LDX    #TAB01    NOLLA ALLA TABULATORSTOPPAR
331  D08E 6F 00   CLRTA2 CLR    0,X
332  D090 08             INX
333  D091 8C 0078        CPX    #TAB39+1  SLUTET?
334  D094 26 F8          BNE    CLRTA2    NIX
335  D096 39             RTS
336                *
337                * HÄR ÄR INPUTLOOPEN
338                *
339  D097 8E C3FF VLOOP  LDS    #STACK
340  D09A BD D14D        JSR    DISCUR    PETA UT CURSORN
341  D09D 8D 56          BSR    ERRUT     BLINKA MED ERROR-LEDEN
342  D09F BD D63D        JSR    RESERR
343  D0A2 8D 5D          BSR    TSTDCD    KOLLA DCD-SNÖRET
344  D0A4 0F      LOOP3  SEI               SPÄRRA IRQ
345  D0A5 01             NOP
346  D0A6 DE 3F          LDX    OUTPEK
347  D0A8 9C 3D          CPX    INPEK     KOLLA OM DATA KVAR I BUFFERT
348  D0AA 26 13          BNE    LOOP5     JO TA HAND OM'ET
349  D0AC 8D D4          BSR    NIX       ÅTERSTÄLL PEKAREN
350  D0AE F6 9101        LDA B  PCISR     HÄMTA STATUS
351  D0B1 C5 02          BIT B  #2        KOLLA OM RX LEDIG
352  D0B3 26 0A          BNE    LOOP5
353  D0B5 7D 0041        TST    LOFLAG
354  D0B8 26 65          BNE    LOOP30
355  D0BA BD D649        JSR    DTRRDY    LEDIGMARKERA !
356  D0BD 20 60          BRA    LOOP30    TESTA BORDET
357                *
358  D0BF 0E      LOOP5  CLI
359  D0C0 DE 3F          LDX    OUTPEK    HÄMTA DATA
360  D0C2 A6 00          LDA A  0,X
361  D0C4 08             INX
362  D0C5 8C 0034        CPX    #INBUF+4
363  D0C8 27 AE          BEQ    DFAULT
364  D0CA DF 3F   LOOP8  STX    OUTPEK
365  D0CC BD D65F        JSR    DTRNO     SPÄRRA INTECKEN
366  D0CF BD D1AC        JSR    INPUT
367  D0D2 7D 0041        TST    LOFLAG
368  D0D5 26 CD          BNE    LOOP3
369                *
370  D0D7 BD D14D LOOP17 JSR    DISCUR    PETA UT CURSORN
371  D0DA 7D 004F        TST    KCTFLG    KOLLA OM KCS INSP.
```

```
372  D0DD 27 40          BEQ    LOOP30    NIX
373  D0DF BD D65F        JSR    DTRNO
374  D0E2 96 48          LDA A  KCDATA    HÄMTA DATA
375  D0E4 8D 02          BSR    KCOUT
376  D0E6 20 37          BRA    LOOP30
377                 *
378  D0E8 8D 03  KCOUT   BSR    KCOUT5
379  D0EA B7 9301        STA A  KCIAD
380  D0ED F6 9300 KCOUT5 LDA B  KCIAC     VÄNTA TILLS TECKNET HAR GÅTT IVÄG
381  D0F0 C5 02          BIT B  #2
382  D0F2 27 F9          BEQ    KCOUT5
383  D0F4 39             RTS
384                 *
385  D0F5 DE 3B  ERRUT   LDX    ERRRAM
386  D0F7 27 25          BEQ    TSTRTS
387  D0F9 09             DEX
388  D0FA DF 3B          STX    ERRRAM    LÄGG TILLBAKS
389  D0FC 26 20          BNE    TSTRTS
390  D0FE 7E D573        JMP    ERLDOF    SLÄCK ERRORLEDEN
391                 *
392  D101 F6 9101 TSTDCD LDA B  PCISR     HÄMTA STATUS
393  D104 37             PSH B          SPARA
394  D105 C5 40          BIT B  #¤40      FINNS DCD?
395  D107 27 05          BEQ    TSTDC5
396
397  D109 BD D591        JSR    DCDLED    TÄND DCD
398  D10C 20 03          BRA    TSTDC8
399                 *
400  D10E BD D567 TSTDC5 JSR    DCDOFF    SLÄCK DCD
401  D111 33      TSTDC8 PUL B          ÅTERTAG
402  D112 C4 30          AND B  #¤30      FRAMING EL OVERRUN?
403  D114 27 08          BEQ    TSTRTS    NIX
404  D116 BD D59D        JSR    ERRLED    TÄND LAMPAN
405  D119 CE 0FFF        LDX    #¤0FFF    BLINKCELL
406  D11C DF 3B          STX    ERRRAM
407  D11E 39      TSTRTS RTS
408                 *
409  D11F F6 9201 LOOP30 LDA B  PIATBC    KOLLA OM KNAPP TRYCKT
410  D122 2A 07          BPL    LOOP50    NIX
411  D124 BD D65F        JSR    DTRNO
412  D127 8D 34          BSR    KBDIN
413  D129 8D 22          BSR    DISCUR
414  D12B 7D 004E LOOP50 TST    KCRFLG
415  D12E 27 10          BEQ    VLOOP7
416  D130 4F             CLR A
417  D131 8D 10          BSR    KCIN
418  D133 4D             TST A
419  D134 27 0A          BEQ    VLOOP7
420  D136 81 14          CMP A  #¤14      SKIP DC1
421  D138 22 04          BHI    LOOP70
422  D13A 81 11          CMP A  #¤11      SKIP DC2-4
423  D13C 24 02          BHS    VLOOP7
424  D13E 8D 54  LOOP70  BSR    OUT1
425  D140 7E D097 VLOOP7 JMP    VLOOP
426                 *
427  D143 F6 9300 KCIN   LDA B  KCIAC     HÄMTA STATUS
428  D146 57             ASR B
429  D147 24 03          BCC    KCIRTS
430  D149 B6 9301        LDA A  KCIAD
431  D14C 39      KCIRTS RTS
432                 *
433  D14D DE 2A  DISCUR  LDX    PEKARE    HÄMTA BILDADRESSEN
```

```
434  D14F E6 00          LDA B  0,X       HÄMTA TECKEN
435  D151 C4 7F          AND B  #¤7F      MASKA BORT EV GAMMAL CURSOR
436  D153 7D 0045        TST    CURFLG    SKALL DEN UT ?
437  D156 26 02          BNE    CUR90     NIX
438  D158 CB 80          ADD B  #¤80      LÄGG PÅ NY CURSOR
439  D15A E7 00   CUR90  STA B  0,X       LÄGG TILLBAKS
440  D15C 39             RTS
441                 *
442  D15D B6 9200 KBDIN  LDA A  PIATBD
443  D160 2A 0C          BPL    SCAN90    OM >7F SÅ ÄR DET STYRKODER
444  D162 CE D732        LDX    #TAB2     TANGENTBORDSTABELL
445  D165 BD D227 SCAN20 JSR    SCAN      KOLLA IGENOM TABELLEN
446  D168 8C D7D1        CPX    #TAB2ND
447  D16B 26 F8          BNE    SCAN20    KNALLA RUNT TILLS SLUTET
448  D16D 39             RTS
449                 *
450  D16E 7D 0041 SCAN90 TST    LOFLAG
451  D171 27 21          BEQ    OUT1
452  D173 81 0F          CMP A  #SI       ÄR DET TELETEXTSHIFT ?
453  D175 26 05          BNE    SCAN93    NIX
454  D177 97 42   TXTMOD STA A  TXTFLG    JODÅ
455  D179 7E D597        JMP    TXTON     TÄND LAMPA
456                 *
457  D17C 7D 0042 SCAN93 TST    TXTFLG    KOLLA OM TELETEXTMODE
458  D17F 26 03          BNE    SCAN94
459  D181 7E D216        JMP    SRCH0     ASCII GOTO NORMAL
460                 *
461  D184 81 0B   SCAN94 CMP A  #¤0B      OM 0B BYT MOT 0D
462  D186 26 05          BNE    SCAN95    NIX
463  D188 86 0D          LDA A  #¤0D      =DUBBEL HÖJD
464  D18A 7E D235        JMP    VISUAB
465                 *
466  D18D 81 0D   SCAN95 CMP A  #CR
467  D18F 26 3C          BNE    INPUT2    TEXT GOTO CHARACTER TEST
468  D191 7E D2E2        JMP    CARRR     GO CR+LF
469                 *
470  D194 7D 0046 OUT1   TST    DUPFLG    KOLLA OM HALV DUPLEX
471  D197 27 04          BEQ    OUT       NIX
472  D199 36      OUT10  PSH A             SPARA TECKNET ** BUG I VERSION 1&2
473  D19A 8D E0          BSR    SCAN93    JO PETA UT PÅ SKÄRMEN FÖRST
474  D19C 32             PUL A
475  D19D F6 9101 OUT    LDA B  PCISR     HÄMTA PCI STATUS
476  D1A0 C5 40          BIT B  #¤40      KOLLA OM DCD
477  D1A2 27 07          BEQ    RETI      OM EJ DCD GÅ RTS
478  D1A4 C5 01          BIT B  #1        LEDIG?
479  D1A6 27 F5          BEQ    OUT       VÄNTA
480  D1A8 B7 9100        STA A  PCIDR
481  D1AB 39      RETI   RTS
482                 *
483  D1AC 97 48   INPUT  STA A  KCDATA
484  D1AE 84 7F          AND A  #¤7F      MASKA PARITETEN
485  D1B0 7D 004A        TST    PARFLG    FINNS PARITETSBIT ?
486  D1B3 27 0E          BEQ    INPUT0    NIX
487  D1B5 F6 9101        LDA B  PCISR     HÄMTA STATUS
488  D1B8 C4 38          AND B  #¤38      KOLLA OM FEL I ÖVERFÖRINGEN
489  D1BA 27 07          BEQ    INPUT0    NIX
490  D1BC BD D63D        JSR    RESERR    NOLLA PCI
491  D1BF 86 7F          LDA A  #¤7F      ERSÄTT PARITETSFEL MED RUBOUT-TECKEN
492  D1C1 20 53          BRA    SRCH0     GÅ OCH PETA UT RUBOUT'ET
493                 *
494  D1C3 4D      INPUT0 TST A
495  D1C4 27 E5          BEQ    RET1
```

```
496                     *
497  D1C6 81 1B    INPUT1  CMP A  #ESC
498  D1C8 26 03            BNE    INPUT2
499  D1CA 97 49            STA A  ESCFLG
500  D1CC 39               RTS
501                     *
502  D1CD 7D 0049  INPUT2  TST    ESCFLG    KOLLA FLAGGAN
503  D1D0 27 31            BEQ    INPUT3
504  D1D2 85 40    INESC   BIT A  #¤40      KOLLA OM CURSORPOS
505  D1D4 27 03            BEQ    INESC2    NIX
506  D1D6 7E D444          JMP    KOORD1
507                     *
508  D1D9 81 20    INESC2  CMP A  #¤20      RAD FRÅN VILKEN SCROLL SKALL BÖRJA
509  D1DB 25 13            BLO    INESC3
510  D1DD 81 37            CMP A  #¤37
511  D1DF 22 0F            BHI    INESC3
512  D1E1 16               TAB
513  D1E2 C0 20            SUB B  #¤20
514  D1E4 BD D45B          JSR    BEGVI2    GÅ OCH FIXA ABSOLUT ADRESS
515  D1E7 97 34            STA A  ROLADR
516  D1E9 D7 35            STA B  ROLADR+1
517  D1EB 7F 0043          CLR    PAGFLG
518  D1EE 20 04            BRA    CLRESC    NOLLA FLAGGAN
519                     *
520  D1F0 8B C0    INESC3  ADD A  #¤C0      FIXA FÖR GEMENSAM TABELL
521  D1F2 8D 04            BSR    ESCRUT
522  D1F4 7F 0049  CLRESC  CLR    ESCFLG    NOLLA ESCFLAGGAN
523  D1F7 39               RTS
524                     *
525  D1F8 CE D6FF  ESCRUT  LDX    #TAB3     TABELL FÖR ESCAPE-KODER
526  D1FB 8D 2A    ESCRU2  BSR    SCAN
527  D1FD 8C D744          CPX    #TAB3ND
528  D200 26 F9            BNE    ESCRU2
529  D202 39               RTS
530                     *
531  D203 7D 0042  INPUT3  TST    TXTFLG
532  D206 27 0A            BEQ    INPUT6
533                     *
534  D208 81 0E    INPUT4  CMP A  #SO       SHIFT TO ASCII?
535  D20A 26 29            BNE    VISUAB    NIX
536  D20C 7F 0042  ASCMOD  CLR    TXTFLG
537  D20F 7E D56D          JMP    TXTOFF
538                     *
539  D212 81 7F    INPUT6  CMP A  #¤7F      OM RUBOUT GÅ RTS
540  D214 27 10            BEQ    INPRTS
541  D216 81 1F    SRCH0   CMP A  #¤1F      TESTA OM SYNLIGT TECKEN
542  D218 22 1B            BHI    VISUAB
543  D21A 8B C0            ADD A  #¤C0      FIXA FÖR GEMENSAMTABELL
544  D21C CE D6FF  SRCH1   LDX    #TAB1
545  D21F 8D 06    SRCH2   BSR    SCAN
546  D221 8C D717          CPX    #TAB1ND   SLUTET PÅ ACIATABELLEN ?
547  D224 26 F9            BNE    SRCH2
548  D226 39       INPRTS  RTS
549                     *
550  D227 A1 00    SCAN    CMP A  0,X
551  D229 26 06            BNE    SCAN2
552  D22B 31               INS
553  D22C 31               INS
554  D22D EE 01            LDX    1,X
555  D22F 6E 00            JMP    0,X
556                     *
557  D231 08       SCAN2   INX
```

```
558  D232 08              INX
559  D233 08              INX
560  D234 39      RETSC   RTS
561               *
562  D235 36      VISUAB  PSH A             SPARA A
563  D236 8D 04           BSR    VISU10
564  D238 32              PUL A             ÅTERTAG
565  D239 7E D7D1         JMP    PRINT
566               *
567  D23C DE 2A   VISU10  LDX    PEKARE
568  D23E A7 00           STA A  0,X
569  D240 08              INX
570  D241 7C 0036         INC    HPOS
571  D244 DF 2A           STX    PEKARE
572  D246 8C C3C0 VISU30  CPX    #TVBEND
573  D249 26 20           BNE    INBILD
574  D24B 7D 0043         TST    PAGFLG    KOLLA OM SCROLL
575  D24E 27 08           BEQ    VISU80    0=SCROLL, FF=PAGE
576  D250 8D 26           BSR    TEVEBE
577  D252 7D 0042         TST    TXTFLG    OM TELETEXT EJ RADERING
578  D255 27 0A           BEQ    SUDDA2
579  D257 39              RTS
580               *
581  D258 8D 2E   VISU80  BSR    SCROLL    GÅ OCH FLYTTA ÜÜÜÜ
582  D25A CE C398         LDX    #TVR23
583  D25D 20 1C           BRA    TEVEB2    +RTS
584               *
585  D25F DE 2C   SUDDA   LDX    XHI
586  D261 C6 28   SUDDA2  LDA B  #LENGD
587  D263 4F      SUDDA4  CLR A             FYLL MED NOLLOR
588  D264 A7 00   SUDDA3  STA A  0,X
589  D266 08              INX
590  D267 5A              DEC B
591  D268 26 FA           BNE    SUDDA3
592  D26A 39      RTS10   RTS
593               *
594  D26B 96 36   INBILD  LDA A  HPOS
595  D26D 81 28           CMP A  #LENGD
596  D26F 26 F9           BNE    RTS10
597  D271 8D 0A           BSR    HPNOLL
598  D273 20 EC           BRA    SUDDA2
599               *
600  D275 BD D2F9 CTLROU  JSR    NOCUR
601               *
602  D278 CE C000 TEVEBE  LDX    #TVB      ÅTERSTÄLL BILDADRESSEN
603  D27B DF 2A   TEVEB2  STX    PEKARE    I PEKAREN
604  D27D 7F 0036 HPNOLL  CLR    HPOS
605  D280 39              RTS
606               *
607  D281 DE 2A   CTZROU  LDX    PEKARE    RADERA TILL BILDENS SLUT
608  D283 8C              FCB    SKIP2     SPARA EN BYTE
609               *
610  D284 8D F2   ERASE   BSR    TEVEBE    SE OVAN
611  D286 20 18           BRA    ERASB     RADERA SKÄRMEN
612               *
613               * PROGRAMSTYRD SCROLL MED INDEXMANIPULERING
614               *
615  D288 BD D65F SCROLL  JSR    DTRNO     SPÄRRA INTECKEN UNDER SCROLLNING
616  D28B DE 34           LDX    ROLADR    HÄMTA ADRESS FÖR SCROLLNING
617  D28D 8C C398         CPX    #TVR23    ÄR DET ENRADSSKRIFT ?
618  D290 27 0C           BEQ    SCROL9    JOVISST
619  D292 A6 28   SCROL5  LDA A  LENGD,X   HÄMTA TECKNET PÅ RADEN UNDER
```

```
620   D294 84 7F          AND A  #¤7F
621   D296 A7 00          STA A  0,X        LÄGG PÅ X-PLATSEN
622   D298 08             INX
623   D299 8C C398 SCROL8 CPX    #TVR23     LÄNGST NER ?
624   D29C 26 F4          BNE    SCROL5
625   D29E DF 2A   SCROL9 STX    PEKARE     SPARA AKTUELL POS
626   D2A0 4F      ERASB  CLR A             FYLL MED NOLLOR
627   D2A1 A7 00   ERAS   STA A  0,X
628   D2A3 08             INX               NÄSTA POS PÅ RADEN
629   D2A4 8C C3C0        CPX    #TVBEND
630   D2A7 26 F8          BNE    ERAS
631   D2A9 39             RTS
632                 *
633   D2AA BD D7D1 LFDR   JSR    PRINT
634   D2AD 8D 4A          BSR    NOCUR      ÅTER MED PEKARE I X-REG
635   D2AF D0 36          SUB B  HPOS       GÅ TILL VÄNSTERKANTEN
636   D2B1 82 00          SBC A  #0         TA MED EV. CARRY
637   D2B3 CB 28   LF10   ADD B  #LENGD     EN RAD NERÅT
638   D2B5 89 00          ADC A  #0         TA MED CARRYN
639   D2B7 81 C3          CMP A  #TVNDHI    UTANFÖR ?
640   D2B9 26 1A          BNE    LF40
641   D2BB C1 BF          CMP B  #TVNDLO-1
642   D2BD 23 16          BLS    LF40
643   D2BF 7D 0043        TST    PAGFLG
644   D2C2 26 06          BNE    LF30
645   D2C4 8D C2          BSR    SCROLL     RADMATA
646   D2C6 8D 39          BSR    PEKAB      ÅTERSTÄLL POS PÅ RADEN
647   D2C8 20 15          BRA    LF95
648                 *
649   D2CA CE C000 LF30   LDX    #TVB
650   D2CD BD D261        JSR    SUDDA2
651   D2D0 86 C0          LDA A  #TVBHI
652   D2D2 5F             CLR B
653   D2D3 20 0A          BRA    LF95
654                 *
655   D2D5 BD D43D LF40   JSR    STORAX     SUDDA UNDERLIGGANDE RAD FÖRE UTSKRIFT
656   D2D8 BD D25F        JSR    SUDDA
657   D2DB 96 2C          LDA A  XHI
658   D2DD D6 2D          LDA B  XLO
659   D2DF 7E D400 LF95   JMP    CTW95      ÅTERSTÄLL POS.
660                 *
661   D2E2 BD D7D1 CARRR  JSR    PRINT
662   D2E5 8D 12          BSR    NOCUR
663   D2E7 D0 36          SUB B  HPOS
664   D2E9 82 00          SBC A  #0
665   D2EB 8D 90          BSR    HPNOLL
666   D2ED 7D 0042 CAR20  TST    TXTFLG
667   D2F0 26 C1          BNE    LF10
668                 *
669   D2F2 97 2A   STOPEK STA A  PEKARE
670   D2F4 D7 2B          STA B  PEKARE+1
671   D2F6 DE 2A   GETPEK LDX    PEKARE
672   D2F8 39             RTS
673                 *
674   D2F9 DE 2A   NOCUR  LDX    PEKARE
675   D2FB A6 00          LDA A  0,X
676   D2FD 84 7F          AND A  #¤7F       TA BORT CURSORN
677   D2FF A7 00          STA A  0,X
678   D301 96 2A   PEKAB  LDA A  PEKARE
679   D303 D6 2B          LDA B  PEKARE+1
680   D305 39      RETURN RTS
681                 *
```

```
682  D306 36       INCPOS  PSH A
683  D307 7C 0036          INC     HPOS
684  D30A 96 36            LDA A   HPOS
685  D30C 81 28            CMP A   #LENGD
686  D30E 26 03            BNE     INCSLU
687  D310 7F 0036          CLR     HPOS
688  D313 32       INCSLU  PUL A
689  D314 39       INCRTS  RTS
690                  *
691                  * RUTINER FÖR ATT SPARA RESP ÅTERTA PEKAREN
692                  *
693  D315 8D E2    SAVEPO  BSR     NOCUR       TA BORT CURSORN
694  D317 DF 77            STX     HIADR       SPARA
695  D319 96 36            LDA A   HPOS
696  D31B 97 37            STA A   HPOS2
697  D31D 39               RTS
698                  *
699  D31E 8D D9    RESTOR  BSR     NOCUR
700  D320 DE 77            LDX     HIADR       ÅTERTAG VÄRDET
701  D322 DF 2A            STX     PEKARE
702  D324 96 37            LDA A   HPOS2
703  D326 97 36            STA A   HPOS
704  D328 39               RTS
705                  *
706                  * SÄND EN RAD, AVSLUTA MED CR
707                  * MAX 40 TECKEN
708                  *
709  D329 8D 1B    SENDRA  BSR     SEND10
710  D32B 8D C9            BSR     GETPEK      HÄMTA POSITIONEN I X-REG
711  D32D D6 36            LDA B   HPOS        HÄMTA ANTALET
712  D32F 27 74            BEQ     SEND9       OM NOLL SÅ ÄR DET FEL
713  D331 5C               INC B               PEKA PÅ EN FÖR MYCKET
714  D332 08               INX
715  D333 09       S100    DEX                 BACKA PEKAREN
716  D334 8C BFFF          CPX     #TVB-1
717  D337 27 04            BEQ     S400
718  D339 5A               DEC B               MINSKA ANTALSRÄKNAREN
719  D33A 26 F7            BNE     S100        GÅ TILL RADENS BÖRJAN
720  D33C 85               FCB     SKIP1
721                  *
722  D33D 08       S400    INX
723  D33E 9C 2A            CPX     PEKARE      CURSORNS POSITION
724  D340 27 63            BEQ     SEND9       KLART
725  D342 8D 08            BSR     SEND        SKICKA IVÄG'ET
726  D344 20 F7            BRA     S400
727                  *
728  D346 BD D693  SEND10  JSR     ONLINE
729  D349 7E D659          JMP     RXFRON      +RTS
730                  *
731  D34C A6 00    SEND    LDA A   0,X         HÄMTA TECKEN
732  D34E 84 7F            AND A   #¤7F        TA BORT CURSORN
733  D350 36               PSH A
734  D351 8B 80            ADD A   #¤80        CURSORMARKERA PÅ SKÄRMEN
735  D353 A7 00            STA A   0,X
736  D355 DF 77            STX     HIADR       SPARA PEKAREN
737  D357 32               PUL A
738  D358 7D 0042          TST     TXTFLG      KOLLA OM TEXTMODE
739  D35B 26 25            BNE     SEND4       ISÅFALL INGEN CR ELLER PAUS
740  D35D 7D 0079          TST     TXPAUS      VAR DET CR SIST ?
741  D360 27 08            BEQ     SEND6       NIX KOLLA OM NOLL
742  D362 4D               TST A               ÄR DET NOLL ?
743  D363 27 16            BEQ     SEND8       NOLLOR EFTER CR IGNORERAS
```

```
744  D365 7F 0079          CLR    TXPAUS    NOLLA CR-FLAGGAN
745  D368 20 1D            BRA    SEND5
746                  *
747  D36A 4D       SEND6   TST A            ÄR DET EN NOLLA EFTER CR ?
748  D36B 26 1A            BNE    SEND5     NIX
749  D36D 86 0D            LDA A  #¤0D      SÄND CR
750  D36F 97 79            STA A  TXPAUS    FLAGGA
751  D371 BD D19D          JSR    OUT
752  D374 86 01            LDA A  #1
753  D376 BD D5D2          JSR    TDEL1     VÄNTLOOP
754  D379 DE 77            LDX    HIADR     HÄMTA PEKAREN IGEN
755  D37B A6 00    SEND8   LDA A  0,X
756  D37D 84 7F            AND A  #¤7F      TA BORT CURSORN
757  D37F A7 00    SEND85  STA A  0,X
758  D381 39               RTS
759                  *
760  D382 4D       SEND4   TST A            KOLLA OM NOLL
761  D383 26 02            BNE    SEND5     NIX
762  D385 86 20            LDA A  #SP       ERSÄTT MED SPACE
763                  *
764                  *
765  D387 7D 0044  SEND5   TST    SNKCFL    KOLLA OM KCS-TX
766  D38A 27 05            BEQ    SEND55    NIX
767  D38C BD D0E8          JSR    KCOUT
768  D38F 20 EE            BRA    SEND85
769                  *
770  D391 BD D19D  SEND55  JSR    OUT       SÄND TILL DATORN
771  D394 20 E9            BRA    SEND85
772                  *
773                  * SÄND HELA SIDAN MED CR EFTER VARJE RAD
774                  *
775  D396 8D AE    SENDPA  BSR    SEND10
776  D398 CE BFFF          LDX    #TVB-1    BÖRJA FRÅN BÖRJAN
777  D39B 08       SENDP2  INX
778  D39C 8C C3C0          CPX    #TVBEND
779  D39F 27 04            BEQ    SEND9     KLART ?
780  D3A1 8D A9            BSR    SEND
781  D3A3 20 F6            BRA    SENDP2
782                  *
783  D3A5 BD D643  SEND9   JSR    RXTILL
784  D3A8 7E D097  SEND11  JMP    VLOOP
785                  *
786                  * SÄND FRÅN BILDENS BÖRJAN TILL CURSORNS POSITION
787                  *
788  D3AB 8D 99    SENDCU  BSR    SEND10
789  D3AD CE BFFF          LDX    #TVB-1
790  D3B0 20 8B            BRA    S400
791                  *
792                  * SÄND HELA SIDAN TILL KCS-MODEMET
793                  *
794  D3B2 97 44    SNDKCS  STA A  SNKCFL
795  D3B4 BD D5BF          JSR    PNCH20    STARTA BANDET
796  D3B7 CE BFFF          LDX    #TVB-1    BÖRJA FRÅN ÖVRE VÄNSTRA HÖRNET
797  D3BA 08       SNDKC2  INX
798  D3BB 8C C3C0          CPX    #TVBEND
799  D3BE 27 04            BEQ    SNDKC9    SLUT
800  D3C0 8D 8A            BSR    SEND
801  D3C2 20 F6            BRA    SNDKC2
802                  *
803  D3C4 86 EE    SNDKC9  LDA A  #¤EE      EOF MARKERING
804  D3C6 BD D0E8          JSR    KCOUT
805  D3C9 7F 0044          CLR    SNKCFL
```

```
806  D3CC BD D588          JSR    TAPEOF
807  D3CF 20 D7            BRA    SEND11
808                  *
809                  *
810                  * CURSOR UPP UTAN RADERING
811                  *
812  D3D1 BD D67E CUPP     JSR    OFFLIN
813  D3D4 BD D2F9 CUPP2    JSR    NOCUR
814  D3D7 C0 28            SUB B  #LENGD
815  D3D9 82 00            SBC A  #0
816  D3DB 81 C0            CMP A  #TVBHI
817  D3DD 24 25            BCC    CTW99
818  D3DF 86 C3            LDA A  #TVBOHI
819  D3E1 C6 98            LDA B  #TVBOLO
820  D3E3 20 1B            BRA    CTW95
821                  *
822                  * CURSOR NER UTAN RADERING
823                  *
824  D3E5 BD D67E CNER     JSR    OFFLIN
825  D3E8 BD D2F9 CNER2    JSR    NOCUR
826  D3EB D0 36            SUB B  HPOS
827  D3ED 82 00            SBC A  #0
828  D3EF CB 28            ADD B  #LENGD
829  D3F1 89 00            ADC A  #0
830  D3F3 81 C3            CMP A  #TVNDHI
831  D3F5 26 07            BNE    CTW90
832  D3F7 C1 C0            CMP B  #TVNDLO
833  D3F9 26 03            BNE    CTW90
834  D3FB 86 C0            LDA A  #TVBHI
835  D3FD 5F               CLR B
836  D3FE 8D 3D    CTW90   BSR    STORAX
837  D400 DB 36    CTW95   ADD B  HPOS          ÅTERSTÄLL POSITIONEN
838  D402 89 00            ADC A  #0
839  D404 7E D2F2  CTW99   JMP    STOPEK
840                  *
841                  *
842                  * CURSOR HÖGER UTAN RADERING
843                  *
844  D407 BD D67E CRAJT    JSR    OFFLIN
845  D40A BD D2F9 CRAJT2   JSR    NOCUR
846  D40D BD D306          JSR    INCPOS
847  D410 08               INX
848  D411 8C C3C0          CPX    #TVBEND
849  D414 26 03            BNE    CTI80
850  D416 CE C000          LDX    #TVB
851  D419 DF 2A    CTI80   STX    PEKARE
852  D41B 39               RTS
853                  *
854                  *
855                  * CURSOR VÄNSTER UTAN RADERING
856                  *
857  D41C BD D7D1 CLEFT1   JSR    PRINT     GO OCH BACKA PRINTBUFFERTEN
858  D41F 20 03            BRA    CLEFT2
859                  *
860  D421 BD D67E CLEFT    JSR    OFFLIN
861  D424 BD D2F9 CLEFT2   JSR    NOCUR
862  D427 96 36            LDA A  HPOS       HÄMTA IN VÄRDET PÅ HPOS
863  D429 4A               DEC A             MINSKA MED ETT
864  D42A 81 FF            CMP A  #¤FF       FÖR LÅNGT ?
865  D42C 26 02            BNE    CTH30
866  D42E 86 27            LDA A  #¤27
867  D430 97 36    CTH30   STA A  HPOS       LÄGG TILLBAKS
```

```
868  D432 09              DEX
869  D433 8C BFFF         CPX    #TVB-1      KOLLA OM UTANFÖR ÖVERKANTEN
870  D436 26 E1           BNE    CTI80       NIX
871  D438 CE C3BF         LDX    #TVBEND-1   SISTA POS PÅ BILDEN
872  D43B 20 DC   CTH50   BRA    CTI80
873                       *
874  D43D 97 2C   STORAX  STA A  XHI
875  D43F D7 2D   STORAB  STA B  XLO
876  D441 DE 2C           LDX    XHI
877  D443 39              RTS
878                       *
879  D444 36      KOORD1  PSH A
880  D445 BD D2F9         JSR    NOCUR
881  D448 7D 004C         TST    KOXFLG      0=X
882  D44B 26 26           BNE    KOORDY
883  D44D 33              PUL B               A FINNS NU I B-ACK
884  D44E C0 40           SUB B  #¤40
885  D450 8D 09           BSR    BEGVI2
886  D452 BD D2F2 BEGVI7  JSR    STOPEK
887  D455 7C 004C         INC    KOXFLG      FLAGGA FÖR Y
888  D458 7E D27D         JMP    HPNOLL
889                  *
890                  * RUTIN FÖR ATT GE X-REG ABSOLUTA ADRESSEN
891                  * TILL BÖRJAN PÅ DEN RAD VARS NUMMER ÅTERFINNS I B-ACK.
892                  *
893  D45B 4F      BEGVI2  CLR A
894  D45C 5D              TST B               KOLLA OM RAD NOLL
895  D45D 27 10           BEQ    BEGVI9
896  D45F C1 17           CMP B  #23         KOLLA OM LAGOM
897  D461 22 0C           BHI    BEGVI9      FÖR STOR
898  D463 8D D8           BSR    STORAX      ÖVERFÖR A+B TILL X
899  D465 8D 08           BSR    BEGVI9
900  D467 CB 28   BEGVI4  ADD B  #LENGD
901  D469 89 00           ADC A  #0
902  D46B 09              DEX
903  D46C 26 F9           BNE    BEGVI4
904  D46E 39              RTS
905                  *
906  D46F 86 C0   BEGVI9  LDA A  #TVBHI
907  D471 5F              CLR B               A+B HAR NU ADRESSEN TILL FÖRSTA BILDPOS
908  D472 39              RTS
909                  *
910  D473 32      KOORDY  PUL A               ÅTERTAG
911  D474 80 40           SUB A  #¤40
912  D476 27 0B           BEQ    KORDY9
913  D478 81 27           CMP A  #LENGD-1     KOLLA ATT INOM GRÄNSERNA
914  D47A 22 07           BHI    KORDY9       UJ FÖR STORT
915  D47C 08      KORDY5  INX                 KLAMPA PÅ
916  D47D BD D306         JSR    INCPOS
917  D480 4A              DEC A               KLART?
918  D481 26 F9           BNE    KORDY5
919  D483 DF 2A   KORDY9  STX    PEKARE
920  D485 4F              CLR A
921  D486 97 4C           STA A  KOXFLG      NOLLA FÖR NORMALLÄGE
922  D488 7E D1F4         JMP    CLRESC
923                  *
924  D48B DE 2A   ERLINE  LDX    PEKARE      ERASE END OF LINE
925  D48D C6 28   ERLIN2  LDA B  #LENGD      40 TECKENS LÄNGD
926  D48F D0 36           SUB B  HPOS        MINSKA MED AKTUELL POSITION
927  D491 7E D263         JMP    SUDDA4
928                  *
929                  * DELETE LINE
```

```
930                     *
931  D494 DE 2A   ROLUP   LDX     PEKARE
932  D496 DF 39           STX     ADRESS     SPARA CURSORNS POSITION
933  D498 BD D301         JSR     PEKAB      HÄMTA A+B
934  D49B D0 36           SUB B   HPOS       GÅ TILL VÄNSTERKANTEN..
935  D49D 82 00           SBC A   #0
936  D49F 8D 9C           BSR     STORAX
937  D4A1 BD D292         JSR     SCROL5     FLYTTA RUBBET
938  D4A4 DE 39           LDX     ADRESS
939  D4A6 DF 2A           STX     PEKARE
940  D4A8 39              RTS
941                     *
942                     * INSERT LINE
943                     *
944  D4A9 BD D301 ROLNER  JSR     PEKAB
945  D4AC D0 36           SUB B   HPOS
946  D4AE 82 00           SBC A   #0
947  D4B0 BD D43D         JSR     STORAX
948  D4B3 CE C3C0         LDX     #TVBEND    BÖRJA SCROLLA I NEDERKANTEN
949  D4B6 09      ROLNE5  DEX                OCH GÅ UPPÅT
950  D4B7 A6 00           LDA A   0,X        HÄMTA TECKEN
951  D4B9 84 7F           AND A   #¤7F       TA BORT CURSORN
952  D4BB A7 28           STA A   LENGD,X    OCH LÄGG PÅ RADEN UNDER
953  D4BD 9C 2C           CPX     XHI        KLART ?
954  D4BF 26 F5           BNE     ROLNE5     NIX
955  D4C1 7E D261         JMP     SUDDA2     I DENNA RUTIN + RTS
956                     *
957                     * DELETE CHARACTER
958                     *
959  D4C4 8D 11   ROLEFT  BSR     LERAJ      FIXA HÖGERKANTENS ADRESS
960  D4C6 D6 37           LDA B   HPOS2      ANTAL POSITIONER
961  D4C8 DE 2A           LDX     PEKARE
962  D4CA A6 01   ROLE30  LDA A   1,X        HÄMTA..
963  D4CC A7 00           STA A   0,X        ..OCH LÄGG UT
964  D4CE 08              INX
965  D4CF 5A              DEC B              MINSKA ANTALET
966  D4D0 26 F8           BNE     ROLE30     KLART?
967  D4D2 86 20   DELCA8  LDA A   #SP        LÄGG EN BLANK..
968  D4D4 A7 00           STA A   0,X        I HÖGRA KANTEN
969  D4D6 39              RTS
970                     *
971  D4D7 86 27   LERAJ   LDA A   #LENGD-1   TOTALA H-ANTALET
972  D4D9 90 36           SUB A   HPOS       MINSKA MED AKTUELL POS.
973  D4DB 26 03           BNE     LERA5
974  D4DD 31              INS
975  D4DE 31              INS
976  D4DF 39              RTS
977                     *
978  D4E0 97 37   LERA5   STA A   HPOS2      SPARA TILLS VIDARE
979  D4E2 BD D301         JSR     PEKAB
980  D4E5 DB 37           ADD B   HPOS2
981  D4E7 89 00           ADC A   #0
982  D4E9 7E D43D         JMP     STORAX     XHI HAR NU HÖGRA KANTENS ADRESS
983                     *
984                     * INSERT CHARACTER
985                     *
986  D4EC 8D E9   ROLRAJ  BSR     LERAJ
987  D4EE 09      ROLRA5  DEX                MINSKA
988  D4EF A6 00           LDA A   0,X        HÄMTA TECKEN
989  D4F1 84 7F           AND A   #¤7F
990  D4F3 A7 01           STA A   1,X
991  D4F5 9C 2A           CPX     PEKARE     ÄR VI FRAMME?
```

```
 992  D4F7 26 F5          BNE     ROLRA5
 993  D4F9 20 D7          BRA     DELCA8
 994                *
 995                *
 996  D4FB F6 9201 BELLR    LDA B   PIATBC     GE PIIIIIP !!!!
 997  D4FE 86 35            LDA A   #¤35
 998  D500 B7 9201          STA A   PIATBC
 999  D503 F7 9201          STA B   PIATBC
1000  D506 39       BELL9   RTS
1001                *
1002                *
1003  D507 8D 03    SETTAB  BSR     TABRX       FIXA PEKAREN
1004  D509 6C 00            INC     0,X         FLAGGA FÖR TABSTOPP
1005  D50B 39               RTS
1006                *
1007  D50C D6 36    TABRX   LDA B   HPOS        DÄR VI ÄR
1008  D50E 5C               INC B
1009  D50F CE 004F          LDX     #TAB01-1    FÖRSTA TABSTOPP
1010  D512 08       TABRX2  INX
1011  D513 8C 0077          CPX     #TAB39
1012  D516 27 03            BEQ     TABRX9
1013  D518 5A               DEC B
1014  D519 26 F7            BNE     TABRX2
1015  D51B 39       TABRX9  RTS
1016                *
1017  D51C 8D EE    DELTAB  BSR     TABRX
1018  D51E 6F 00            CLR     0,X         TA BORT TABSTOPP
1019  D520 39               RTS
1020                *
1021  D521 BD D2F9 SKIP     JSR     NOCUR
1022  D524 8D E6            BSR     TABRX
1023  D526 8C 0077          CPX     #TAB39
1024  D529 27 F0            BEQ     TABRX9
1025  D52B C6 20            LDA B   #SP
1026  D52D DF 39    SKIP3   STX     ADRESS
1027  D52F DE 2A            LDX     PEKARE
1028  D531 E7 00            STA B   0,X         FYLL MED SPACE
1029  D533 08               INX
1030  D534 DF 2A            STX     PEKARE
1031  D536 96 36            LDA A   HPOS
1032  D538 81 27            CMP A   #LENGD-1    ÄR DET SISTA POS ?
1033  D53A 27 0F            BEQ     SKIP9
1034  D53C 7C 0036          INC     HPOS
1035  D53F DE 39            LDX     ADRESS
1036  D541 08               INX
1037  D542 8C 0077          CPX     #TAB39
1038  D545 27 D4            BEQ     TABRX9
1039  D547 6D 00            TST     0,X
1040  D549 27 E2            BEQ     SKIP3       KNALLA PÅ TILLS STOPP HITTAS
1041  D54B 7D 0041 SKIP9    TST     LOFLAG
1042  D54E 26 CB            BNE     TABRX9      OM LOCAL GO RTS
1043  D550 86 20            LDA A   #SP         SÄND SPACE
1044  D552 7E D19D          JMP     OUT
1045                *
1046                *
1047  D555 97 4B    PRTON   STA A   PRTFLG
1048  D557 20 41            BRA     LPRTON      TÄND PRINTLAMPA
1049                *
1050  D559 7F 004B PRTOFF   CLR     PRTFLG
1051  D55C 20 12            BRA     LPRTOF
1052                *
1053  D55E C6 01    TAPEON  LDA B   #1          KASSETTSPELARE TILL
```

```
1054  D560 8C          FCB   SKIP2
1055  D561 C6 03 RECORD LDA B #3       INSPELNINGSOMKOPPLING
1056  D563 8C          FCB   SKIP2
1057  D564 C6 04 LOFLIN LDA B #¤04     ON LINE LED SLÄCKT
1058  D566 8C          FCB   SKIP2
1059  D567 C6 08 DCDOFF LDA B #¤8      LED FÖR DCD LÅG
1060  D569 8C          FCB   SKIP2
1061  D56A C6 10 TACOFF LDA B #¤10     LED FÖR TAPEKONTROLL
1062  D56C 8C          FCB   SKIP2
1063  D56D C6 20 TXTOFF LDA B #¤20     LED FÖR TELETEXTMODE
1064  D56F 8C          FCB   SKIP2
1065  D570 C6 40 LPRTOF LDA B #¤40     PRINTER LED FRÅN
1066  D572 8C          FCB   SKIP2
1067  D573 C6 80 ERLDOF LDA B #¤80
1068  D575 FA 9202     ORA B PIAKD
1069  D578 20 28       BRA   PIAK99
1070               *
1071  D57A 7F 0047 TAPOFF CLR   TAPFLG
1072  D57D 20 EB       BRA   TACOFF     SLÄCK LED
1073               *
1074  D57F 97 47 TAPON  STA A TAPFLG
1075  D581 20 11       BRA   TACON      TÄND LED
1076               *
1077  D583 5F    READOF CLR B
1078  D584 D7 4E       STA B KCRFLG
1079  D586 D7 4F       STA B KCTFLG
1080               *
1081  D588 C6 FC TAPEOF LDA B #¤FC     KASSETTSPELARE FRÅN
1082  D58A 8C          FCB   SKIP2
1083  D58B C6 FD RECOFF LDA B #¤FD     AVSPELNING FRÅN
1084  D58D 8C          FCB   SKIP2
1085  D58E C6 FB LONLIN LDA B #¤FB     TERM ONLINE
1086  D590 8C          FCB   SKIP2
1087  D591 C6 F7 DCDLED LDA B #¤F7     DCD LÅG
1088  D593 8C          FCB   SKIP2
1089  D594 C6 EF TACON  LDA B #¤EF
1090  D596 8C          FCB   SKIP2
1091  D597 C6 DF TXTON  LDA B #¤DF
1092  D599 8C          FCB   SKIP2
1093  D59A C6 BF LPRTON LDA B #¤BF     PRINTER LED
1094  D59C 8C          FCB   SKIP2
1095  D59D C6 7F ERRLED LDA B #¤7F     ERROR LED TÄND
1096  D59F F4 9202     AND B PIAKD
1097  D5A2 F7 9202 PIAK99 STA B PIAKD
1098  D5A5 39          RTS
1099               *
1100  D5A6 7D 0047 READON TST   TAPFLG
1101  D5A9 27 0E       BEQ   TAPRTS
1102  D5AB 8D B1       BSR   TAPEON     STARTA BANDET
1103  D5AD 86 15       LDA A #¤15       8+1 STOPPBIT
1104  D5AF B7 9300     STA A KCIAC
1105  D5B2 8D D7       BSR   RECOFF     AVSPELNING
1106  D5B4 97 4E       STA A KCRFLG
1107  D5B6 4F          CLR A
1108  D5B7 97 4F       STA A KCTFLG
1109  D5B9 39    TAPRTS RTS
1110               *
1111  D5BA 7D 0047 PNCHON TST   TAPFLG
1112  D5BD 27 FA       BEQ   TAPRTS
1113  D5BF BD D65F PNCH20 JSR   DTRNO
1114  D5C2 8D 9D       BSR   RECORD
1115  D5C4 86 11       LDA A #¤11       8+2 STOPPBIT
```

```
1116  D5C6 B7 9300          STA A  KCIAC
1117  D5C9 97 4F            STA A  KCTFLG
1118  D5CB 4F               CLR A
1119  D5CC 97 4E            STA A  KCRFLG
1120  D5CE 97 48            STA A  KCDATA
1121                  *
1122  D5D0 86 04    TDEL    LDA A  #¤04
1123  D5D2 CE FFFF  TDEL1   LDX    #¤FFFF
1124  D5D5 09       TDEL2   DEX                  VÄNTLOOP
1125  D5D6 26 FD            BNE    TDEL2
1126  D5D8 4A               DEC A                I TVÅ SLINGOR
1127  D5D9 26 F7            BNE    TDEL1
1128  D5DB 39               RTS
1129                  *
1130                  * RUTIN FÖR BAUDRATEOMKOPPLING
1131                  *
1132  D5DC 36       SPEED   PSH A
1133  D5DD BD D2F9          JSR    NOCUR
1134  D5E0 32               PUL A
1135  D5E1 F6 9103          LDA B  PCICR        NOLLA FÖR MR1
1136  D5E4 84 0F            AND A  #¤0F
1137  D5E6 81 04            CMP A  #4           KOLLA OM 1 EL 2 STOPPBITAR
1138  D5E8 22 06            BHI    ENSTOP
1139  D5EA D6 2E    TOSTOP  LDA B  RPCIM1       HÄMTA I RAM
1140  D5EC CB 80            ADD B  #¤80         TVÅSTOPP
1141  D5EE 20 04            BRA    SPEED5
1142                  *
1143  D5F0 D6 2E    ENSTOP  LDA B  RPCIM1
1144  D5F2 C4 7F            AND B  #¤7F
1145  D5F4 D7 2E    SPEED5  STA B  RPCIM1
1146  D5F6 D6 2F            LDA B  RPCIM2       HÄMTA RAM-STATUS
1147  D5F8 C4 30            AND B  #¤30         MASKA INT/EXT KLOCKA
1148  D5FA 1B               ABA
1149  D5FB D6 2E            LDA B  RPCIM1       HÄMTA RAM FÖR MODEREG 1
1150  D5FD F7 9102  SPEED8  STA B  PCIMR        LÄGG UT TILL REG 1
1151  D600 B7 9102          STA A  PCIMR        LÄGG UT TILL REG 2
1152  D603 97 2F            STA A  RPCIM2       SPARA I RAM
1153  D605 7E D04F  SPEED9  JMP    RESTA0
1154                  *
1155  D608 7F 004A  NOPAR   CLR    PARFLG       INGEN PARITETSBIT
1156  D60B D6 2E            LDA B  RPCIM1
1157  D60D C4 C0            AND B  #¤C0         SPARA STOPPBITSANTALET
1158  D60F 86 0E            LDA A  #¤0E         8 BITAR, INGEN PARITET
1159  D611 1B       NOPAR3  ABA
1160  D612 97 2E            STA A  RPCIM1
1161  D614 B7 9102          STA A  PCIMR
1162  D617 96 2F            LDA A  RPCIM2
1163  D619 B7 9102          STA A  PCIMR
1164  D61C 20 E7            BRA    SPEED9
1165                  *
1166  D61E D6 2E    EVPAR   LDA B  RPCIM1       JÄMN PARITET
1167  D620 C4 C0            AND B  #¤C0
1168  D622 86 3A            LDA A  #¤3A         7 + JÄMN PAR.
1169  D624 97 4A            STA A  PARFLG
1170  D626 20 E9            BRA    NOPAR3
1171                  *
1172  D628 D6 2E    ODDPAR  LDA B  RPCIM1
1173  D62A C4 C0            AND B  #¤C0
1174  D62C 86 1A            LDA A  #¤1A         7 BITAR UDDA PARITET
1175  D62E 97 4A            STA A  PARFLG
1176  D630 20 DF            BRA    NOPAR3
1177                  *
```

```
1178  D632 C6 4E   DVBAUD  LDA B  #%01001110 75 BAUD TX, 1200 BAUD RX
1179  D634 86 17           LDA A  #%00010111 EXT TXCLK
1180  D636 D7 2E           STA B  RPCIM1
1181  D638 20 C3           BRA    SPEED8
1182                       *
1183                       * RUTIN FÖR STYRNING AV PCI 2651
1184                       *
1185  D63A C6 20   RTSLO    LDA B  #¤20       REQUEST TO SEND LOW
1186  D63C 8C              FCB    SKIP2
1187  D63D C6 10   RESERR   LDA B  #¤10       RESET ERROR
1188  D63F 8C              FCB    SKIP2
1189  D640 C6 08   FBREAK   LDA B  #8         FORCE BREAK
1190  D642 8C              FCB    SKIP2
1191  D643 C6 04   RXTILL   LDA B  #¤04       RECEIVER TILL
1192  D645 8C              FCB    SKIP2
1193  D646 C6 07   RXEN     LDA B  #¤7        RECEIVER ENABLE
1194  D648 8C              FCB    SKIP2
1195  D649 C6 02   DTRRDY   LDA B  #¤2        DATA TERM READY LOW
1196  D64B 8C              FCB    SKIP2
1197  D64C C6 01   TXEN     LDA B  #1         TRANSMITTER ENABLE
1198  D64E FA 9103         ORA B  PCICR
1199  D651 20 14           BRA    PCI99
1200                       *
1201  D653 C6 DF   RTSHI    LDA B  #¤DF       REQUEST TO SEND HIGH
1202  D655 8C              FCB    SKIP2
1203  D656 C6 F7   NBREAK   LDA B  #¤F7       NORMAL
1204  D658 8C              FCB    SKIP2
1205  D659 C6 FB   RXFRON   LDA B  #¤FB       RECEIVER FRÅN
1206  D65B 8C              FCB    SKIP2
1207  D65C C6 F8   RXDIS    LDA B  #¤F8       RECEIVER DISABLE +DTRNO+TXDIS
1208  D65E 8C              FCB    SKIP2
1209  D65F C6 FD   DTRNO    LDA B  #¤FD       DATA TERMINAL NOT READY
1210  D661 8C              FCB    SKIP2
1211  D662 C6 FE   TXDIS    LDA B  #¤FE       TRANSMITTER DISABLE
1212  D664 F4 9103         AND B  PCICR
1213  D667 F7 9103 PCI99    STA B  PCICR
1214  D66A 39              RTS
1215                       *
1216  D66B 8D F2   IRQSEQ   BSR    DTRNO
1217  D66D B6 9100         LDA A  PCIDR      HÄMTA DATA CLEARA INTERRUPTET
1218  D670 DE 3D           LDX    INPEK
1219  D672 8C 0034         CPX    #INBUF+4   FULLT?
1220  D675 26 01           BNE    IRQ95
1221  D677 09              DEX
1222  D678 A7 00   IRQ95    STA A  0,X
1223  D67A 08              INX
1224  D67B DF 3D           STX    INPEK
1225  D67D 3B      RETINT   RTI
1226                       *
1227  D67E 7C 0041 OFFLIN   INC    LOFLAG
1228  D681 8D D0           BSR    RTSHI
1229  D683 8D DA           BSR    DTRNO
1230  D685 7E D564         JMP    LOFLIN     +RTS
1231                       *
1232  D688 8D F4   OFLI70   BSR    OFFLIN
1233  D68A 8E C3FF         LDS    #STACK
1234  D68D 7E D0A4         JMP    LOOP3
1235                       *
1236  D690 4F      ONL1FU   CLR A             0=FULL DUPLEX
1237  D691 97 46   ONLIHA   STA A  DUPFLG     HALV DUPLEX
1238  D693 8D B1   ONLINE   BSR    RXEN       TERMINAL READY
1239  D695 7F 0041         CLR    LOFLAG
```

```
1240  D698 8D A0           BSR    RTSLO
1241  D69A 7E D58E         JMP    LONLIN     TÄND ON-LINE LED
1242                *
1243  D69D CE C000 NOMRAD  LDX    #TVB       FULL SCROLL
1244  D6A0 20 08           BRA    RADX
1245                *
1246  D6A2 CE C370 TWORAD  LDX    #TVR22
1247  D6A5 20 03           BRA    RADX
1248                *
1249  D6A7 CE C1E0 HALVSC  LDX    #TVR12
1250  D6AA DF 34   RADX    STX    ROLADR
1251  D6AC 4F              CLR A
1252  D6AD 97 43   PAGE    STA A  PAGFLG
1253  D6AF 39              RTS
1254                *
1255                * RUTIN FÖR ATT SÄNDA BREAK
1256                *
1257  D6B0 8D 8E   LBREAK  BSR    FBREAK
1258  D6B2 86 04           LDA A  #4
1259  D6B4 BD D5D2         JSR    TDEL1
1260  D6B7 8D 9D           BSR    NBREAK
1261  D6B9 8C              FCB    SKIP2
1262  D6BA 8D 87   HIRTS   BSR    RXTILL
1263  D6BC 7E D097         JMP    VLOOP
1264                *
1265                *
1266  D6BF 4F      CURON   CLR A              CURSOR TILL
1267  D6C0 97 45   CUROFF  STA A  CURFLG      CURSOR FRÅN
1268  D6C2 39      PRTS    RTS
1269                *
1270                * HERE-IS FUNKTION
1271                *
1272  D6C3 8D CE   HEREIS  BSR    ONLINE
1273  D6C5 8D 92           BSR    RXFRON
1274  D6C7 CE D7E8         LDX    #TEXTH1    STRÄNGEN
1275  D6CA A6 00   HERE50  LDA A  0,X        HÄMTA TECKEN
1276  D6CC 27 EC           BEQ    HIRTS      OM NOLL = KLART
1277  D6CE DF 77           STX    HIADR
1278  D6D0 BD D199         JSR    OUT1O
1279  D6D3 DE 77           LDX    HIADR
1280  D6D5 08              INX
1281  D6D6 20 F2           BRA    HERE50
1282                *
1283  D6D8 BD D57F MOTKCS  JSR    TAPON
1284  D6DB 8D D0           BSR    PAGE
1285  D6DD BD D5A6         JSR    READON
1286  D6E0 BD D278         JSR    TEVEBE
1287  D6E3 BD D143 MOTKC3  JSR    KCIN       HÄMTA TECKEN
1288  D6E6 25 0E           BCS    MOTKC4
1289  D6E8 F6 9201         LDA B  PIATBC
1290  D6EB 2A F6           BPL    MOTKC3
1291  D6ED B6 9200         LDA A  PIATBD
1292  D6F0 BD D588 MOTKC9  JSR    TAPEOF
1293  D6F3 7E D097         JMP    VLOOP
1294                *
1295  D6F6 81 EE   MOTKC4  CMP A  #¤EE        EOF ?
1296  D6F8 27 F6           BEQ    MOTKC9
1297  D6FA BD D23C         JSR    VISU10
1298  D6FD 20 E4           BRA    MOTKC3
1299                *
1300  D6FF         TAB3    EQU    *
1301  D6FF CA      TAB1    FCB    ¤CA         LINE FEED
```

```
1302  D700 D2 AA          FDB     LFDR
1303  D702 CD             FCB     ¤CD         CARRIAGE RETURN
1304  D703 D2 E2          FDB     CARRR
1305  D705 C7             FCB     ¤C7         BELL-KOD
1306  D706 D4 FB          FDB     BELLR
1307  D708 D1             FCB     ¤D1         READER ON
1308  D709 D5 A6          FDB     READON
1309  D70B D2             FCB     ¤D2         PUNCH ON
1310  D70C D5 BA          FDB     PNCHON
1311  D70E D3             FCB     ¤D3         READER OFF
1312  D70F D5 83          FDB     READOF
1313  D711 D4             FCB     ¤D4         PUNCH OFF
1314  D712 D5 83          FDB     READOF
1315  D714 C8             FCB     ¤C8         CURSOR VÄNSTER
1316  D715 D4 1C          FDB     CLEFT1
1317                      *
1318  D717                TAB1ND  EQU     *   NEDANSTÅENDE RUTINER ENDAST EFTER ESCAPE
1319                      *
1320  D717 C9             FCB     ¤C9         CURSOR HÖGER
1321  D718 D4 0A          FDB     CRAJT2
1322  D71A CF             FCB     ¤CF
1323  D71B D1 77          FDB     TXTMOD
1324  D71D CB             FCB     ¤CB         CURSOR UPP
1325  D71E D3 D4          FDB     CUPP2
1326  D720 D7             FCB     ¤D7         CURSOR NER
1327  D721 D3 E8          FDB     CNER2
1328  D723 D6             FCB     ¤D6         RESTORE POINTER (ESC+CTL V)
1329  D724 D3 1E          FDB     RESTOR
1330  D726 D9             FCB     ¤D9         SAVE POINTER (ESC+CTLY)
1331  D727 D3 15          FDB     SAVEPO
1332  D729 C2             FCB     ¤C2         TÄND CURSORN
1333  D72A D6 BF          FDB     CURON
1334  D72C C1             FCB     ¤C1         PAGEMODE
1335  D72D D6 AD          FDB     PAGE
1336  D72F D5             FCB     ¤D5         SLÄCK CURSORN
1337  D730 D6 C0          FDB     CUROFF
1338                      *
1339  D732 CE     TAB2    FCB     ¤CE         PRINTER OFF  =N
1340  D733 D5 59          FDB     PRTOFF
1341  D735 D0             FCB     ¤D0         PRINTER ON =P
1342  D736 D5 55          FDB     PRTON
1343  D738 C5             FCB     ¤C5         RADERA SKÄRMEN = E
1344  D739 D2 84          FDB     ERASE
1345  D73B DA             FCB     ¤DA         ERASE END OF BILD =Z
1346  D73C D2 81          FDB     CTZROU
1347  D73E C3             FCB     ¤C3         ERASE END OF LINE =C
1348  D73F D4 8B          FDB     ERLINE
1349  D741 CC             FCB     ¤CC         CURSOR HOME =L
1350  D742 D2 75          FDB     CTLROU
1351                      *
1352  D744                TAB3ND  EQU     *
1353                      *
1354  D744 C1             FCB     ¤C1         ON LINE + FULL DUPLEX =A
1355  D745 D6 90          FDB     ONLIFU
1356  D747 81             FCB     ¤81         ON LINE + HALV DUPLEX =CTRL A
1357  D748 D6 91          FDB     ONLIHA
1358  D74A C2             FCB     ¤C2         OFF LINE =B
1359  D74B D6 88          FDB     OFLI70
1360  D74D D5             FCB     ¤D5         NORMALSCROLL =U
1361  D74E D6 9D          FDB     NOMRAD
1362  D750 D9             FCB     ¤D9         HALVSIDESSCROLL =Y
1363  D751 D6 A7          FDB     HALVSC
```

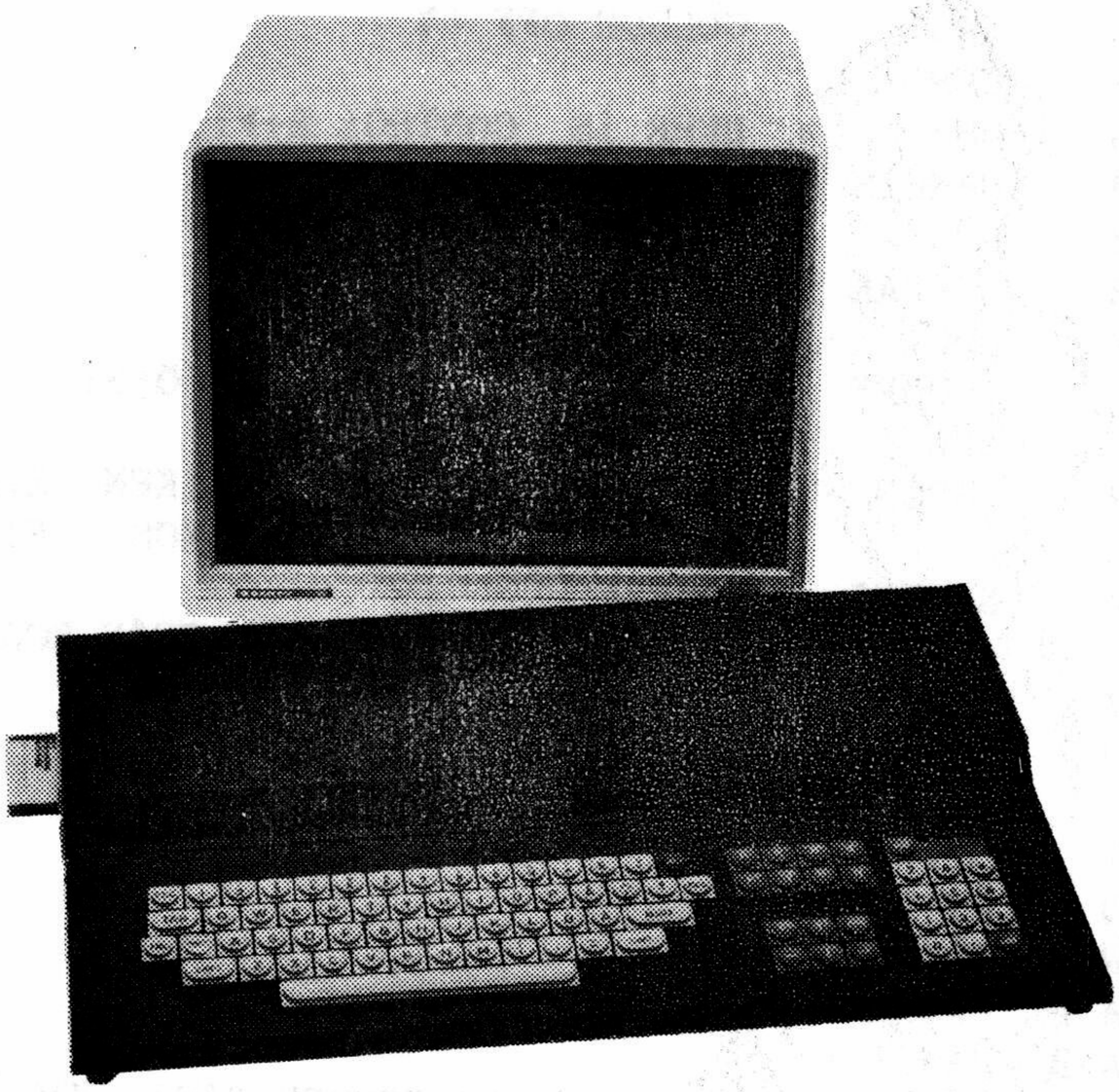
```
1364  D753 D4          FCB    ¤D4       TVÅRADSSCROLLMODE =T
1365  D754 D6 A2       FDB    TWORAD
1366  D756 D2          FCB    ¤D2       PAGE MODE =R
1367  D757 D6 AD       FDB    PAGE
1368  D759 D7          FCB    ¤D7       CURSOR NER + OFF LINE  =W
1369  D75A D3 E5       FDB    CNER
1370  D75C CB          FCB    ¤CB       CURSOR UPP + OFF LINE =K
1371  D75D D3 D1       FDB    CUPP
1372  D75F AB          FCB    ¤AB       CURSOR UPP + OFF LINE
1373  D760 D3 D1       FDB    CUPP
1374  D762 AD          FCB    ¤AD       CURSOR NER + OFF LINE
1375  D763 D3 E5       FDB    CNER
1376  D765 88          FCB    ¤88       CURSOR VÄNSTER + OFF LINE
1377  D766 D4 21       FDB    CLEFT
1378  D768 89          FCB    ¤89       CURSOR HÖGER + OFF LINE
1379  D769 D4 07       FDB    CRAJT
1380  D76B CD          FCB    ¤CD       DELETE LINE =M
1381  D76C D4 94       FDB    ROLUP
1382  D76E D6          FCB    ¤D6       INSERT LINE =V
1383  D76F D4 A9       FDB    ROLNER
1384  D771 C8          FCB    ¤C8       CURSOR VÄNSTER + OFF LINE =H
1385  D772 D4 21       FDB    CLEFT
1386  D774 C9          FCB    ¤C9       CURSOR HÖGER + OFF LINE =I
1387  D775 D4 07       FDB    CRAJT
1388  D777 CA          FCB    ¤CA       INSERT CHARACTER =J
1389  D778 D4 EC       FDB    ROLRAJ
1390  D77A CF          FCB    ¤CF       DELETE CHARACTER =O
1391  D77B D4 C4       FDB    ROLEFT
1392  D77D D1          FCB    ¤D1       SÄND SIDA  =Q
1393  D77E D3 96       FDB    SENDPA
1394  D780 B2          FCB    ¤B2       110 BAUD  =2
1395  D781 D5 DC       FDB    SPEED
1396                *FCB ¤B3 135 BAUD  =3
1397                *FDB SPEED
1398  D783 B4          FCB    ¤B4       150 BAUD  =4
1399  D784 D5 DC       FDB    SPEED
1400  D786 B5          FCB    ¤B5       300 BAUD  =5
1401  D787 D5 DC       FDB    SPEED
1402  D789 B6          FCB    ¤B6       600 BAUD  =6
1403  D78A D5 DC       FDB    SPEED
1404  D78C B7          FCB    ¤B7       1200 BAUD =7
1405  D78D D5 DC       FDB    SPEED
1406                *FCB ¤B8 1800 BAUD  =8
1407                *FDB SPEED
1408                * FCB ¤B9 2000 BAUD  =9
1409                * FDB SPEED
1410  D78F BA          FCB    ¤BA       2400 BAUD  =:
1411  D790 D5 DC       FDB    SPEED
1412                * FCB ¤BB 3600 BAUD  =;
1413                * FDB SPEED
1414  D792 BC          FCB    ¤BC       4800 BAUD  =<
1415  D793 D5 DC       FDB    SPEED
1416                * FCB ¤BD 7200 BAUD  ==
1417                * FDB SPEED
1418  D795 BE          FCB    ¤BE       9600 BAUD  =>
1419  D796 D5 DC       FDB    SPEED
1420                *FCB ¤BF 19200 BAUD  =?
1421                *FDB SPEED
1422  D798 A1          FCB    ¤A1       INGEN PARITET  =!
1423  D799 D6 08       FDB    NOPAR
1424  D79B A2          FCB    ¤A2       JÄMN PARITET   ="
1425  D79C D6 1E       FDB    EVPAR
```

```
1426  D79E A3          FCB   ¤A3       UDDA PARITET    =#
1427  D79F D6 28       FDB   ODDPAR
1428  D7A1 DB          FCB   ¤DB       NOLLA TABSTOPP =Ä
1429  D7A2 D0 8B       FDB   CLRTAB
1430  D7A4 D3          FCB   ¤D3       SET TABSTOPP =S
1431  D7A5 D5 07       FDB   SETTAB
1432  D7A7 C4          FCB   ¤C4       DELETE TABSTOPP =D
1433  D7A8 D5 1C       FDB   DELTAB
1434  D7AA C6          FCB   ¤C6       FLYTTA TILL TABSTOPP =F
1435  D7AB D5 21       FDB   SKIP
1436  D7AD C7          FCB   ¤C7       SÄND RAD =G
1437  D7AE D3 29       FDB   SENDRA
1438  D7B0 C0          FCB   ¤C0       SÄND FRÅN B1LDENS BÖRJAN TILL CURSORN = É
1439  D7B1 D3 AB       FDB   SENDCU
1440  D7B3 84          FCB   ¤84       DATAVISIONSSPEED
1441  D7B4 D6 32       FDB   DVBAUD
1442  D7B6 D8          FCB   ¤D8       ÅTERSTÄLL NORMALPARAMETRAR =X
1443  D7B7 D0 78       FDB   DFAULT
1444  D7B9 A4          FCB   ¤A4       CURSOR ON =¤
1445  D7BA D6 BF       FDB   CURON
1446  D7BC A5          FCB   ¤A5       CURSOR OFF =%
1447  D7BD D6 C0       FDB   CUROFF
1448  D7BF 82          FCB   ¤82       HERE IS  CONTROL B+F1
1449  D7C0 D6 C3       FDB   HEREIS
1450  D7C2 83          FCB   ¤83       BREAK  CONTROL C +F1
1451  D7C3 D6 B0       FDB   LBREAK
1452  D7C5 DC          FCB   ¤DC
1453  D7C6 D5 7A       FDB   TAPOFF    STÄNG AV TAPE-FUNKT.  Ö+F1
1454  D7C8 DD          FCB   ¤DD
1455  D7C9 D5 7F       FDB   TAPON     KOPPLA IN KCS-AUTOMATIKEN  Å+F1
1456  D7CB 91          FCB   ¤91       SÄND HELA SIDAN TILL KCS = F1+CTRL Q
1457  D7CC D3 B2       FDB   SNDKCS
1458  D7CE 92          FCB   ¤92       FYLL BILDEN MED DATA FRÅN KCS
1459  D7CF D6 D8       FDB   MOTKCS
1460  D7D1             TAB2ND EQU    *
1461                   *
1462  D7D1 7D 004B     PRINT  TST    PRTFLG
1463  D7D4 27 0C             BEQ    PRITS
1464  D7D6 8D 03       PRINT2 BSR    PRINT5
1465  D7D8 B7 9001           STA A  PACIA+1
1466  D7DB F6 9000     PRINT5 LDA B  PACIA
1467  D7DE C5 02             BIT B  #2        VÄNTA TILLS TECKEN GÅTT IVÄG
1468  D7E0 27 F9             BEQ    PRINT5
1469  D7E2 39          PRITS  RTS
1470                   *
1471  D7E8                   ORG    TPGM+¤7E8
1472                   *
1473                   * TEXTSTRÄNG FÖR HERE IS
1474                   *
1475  D7E8             TEXTHI EQU    *
1476                   *
1477  D7F7                   ORG    TPGM+¤7F7
1478                   *
1479  D7F7 00                FCB    0         SLUT PÅ TEXTSTRÄNGEN
1480                   *
1481  D7F8 D6 6B             FDB    IRQSEQ
1482  D7FA D6 7D             FDB    RETINT
1483  D7FC D6 7D             FDB    RETINT
1484  D7FE D0 0B             FDB    START
1485                   *
1486                         END
```

Candela

Candela är en 6809-baserad dator, som utvecklats i Linköping för i första hand styr- mät- och reglertillämpningar inom industri, utveckling och utbildning. Som operativsystem finns möjlighet att välja mellan OS-9 level 1 och FLEX9. Till båda operativsystemen finns programvara för grafik samt de inbyggda A/D- och D/A-omvandlarna.

Med hjälp av PROM-bank och s k RAM-floppy kan OS-9 samtidigt ha tillgång till ett stort antal moduler utan läsning från disk. Detta gör Candela-datorn snabb och lättanvänd. Hela skivan i den inbyggda 3" diskenheten kan användas för egna program, då alla systemmoduler finns i datorns RAM eller PROM.



Faktaruta

Processor: 6809
Operativsystem: OS9 och/eller FLEX
RAM: 128 kByte varav 64 kByte systemminne och 64 kByte bildminne
PROM: 8 kByte (kernel, monitor)
RAM-floppy: 128 kByte som standard, expanderbar till 1 MByte
PROM-bank: Plats för 256 kByte (varav 64 kByte i PROM-kassett) prommad programvara för inlösning via port. Inläsningshastighet: 650 kbit/s
Flexskive-enhet: 3" mikro-floppy, 160 kByte/sida formatterat. kontakt finns för anslutning av ytterligare max 3 enheter, 3" eller 5 1/4"
Högupplösningsgrafik: upp till 512 x 480 punkter, 4 färger av 256 möjliga.
Lågupplsöningsgrafik: upp till 256 x 480 punkter, 16 färger av 256
Tangentbord: Egen processor 6802, 2 k RAM, 4 k PROM, 1 PIA och 2 ACIOR. Alfanumeriska, funktions-, markör- och numeriska tangenter. Markör- och funktionstangenter programmerbara med upp till 16 ASCII-tecken vardera. Timer, egen printer-utgång, ställbar auto-repetitionsfrekvens, editeringsmöjligheter m.m. m.m.

I/O OCH EXPANSIONSKONTAKTER

ANALOG I/O: Analoga inkanaler: 8 st, analoga utkanaler 2 st + kontrollsignaler.
Snabb A/D-omvandlare: 8 bitar 2 µs omvandlingstid
Noggrann A/D-omvandlare: 12 bitar 100 ms
D/A-omvandlare: 2 st, 8 bitar
SERIAL I/O: RS232 2 st serieportar (en via tangentbordet)
DIGITAL I/O: 16 bitar + 4 kontrollsignaler (PIA)
AUX I/O: 4 kontrollsignaler. Kan användas till Esseltes MÄT-100-serie.
SAMT kontakter för buss-expansion, prom-kassett, extra skiv-enheter, RGB- och monokrom monitor.

STANDARDPROGRAMVARA OS9
På skiva: OS9 och grafik-mätpaket.
I PROM-bank: Basic 09, utilities, EDITOR-ASSEMBLER-DEBUG-paket, Bildskärms-editor ED9

STANDARDPROGRAMVARA FLEX
I PROM-bank: FLEX, utilities, XBASIC
På skiva: Grafik-paket

Tillbehör

Monitor (färg eller monokrom)
Printer
Plotter, 4 eller 6 färger
RAM-modul 64 kB
PROM-kasset med plats för två EPROM 8--32 kB
Interface för Databoard 4680 I/O-kort

PRISEXEMPEL

Art.nr.	Benämning	Pris
11 000	Candela, FD 160 kB, OS-9, Basic09, ASM, Grafik-mät-paket, Bildskärmseditor ED9 128 kB RAM-floppy	15.000:-
11 001	Candela, FD 160 kB, FLEX, XBasic, Expac, 128 kB RAM-floppy	16.000:-
11 010	Bildskärm, Sanyo 14" RGB-analog	5.350:-
11 015	Bildskärm, Sanyo 12" monokrom	1.550:-
11 100	RAM-modul 64 kB	500:-
11 110	Huvudkort	3.900:-
11 111	Videokort	450:-
11 112	PROM-A/D-kort	1.200:-
11 115	Separat tangentbord (monterat)	2.500:-

Till ovanstående priser tillkommer moms och frakt.
Reservation för prisjusteringar.

Candela Data AB

Postadress
Box 20
581 02 Linköping

Telefon
013/10 47 50

Besöksadress
Repslagaregatan 12
Linköping