INSTITUTTET FOR HUSBYGNING

Rapport nr. **185**

JOHS. F. MUNCH-PETERSEN

STORE, AFBILDNINGSTRO PRINTERKOPIER AF SKÆRMBILLEDER PÅ QL - DATAMATEN

Den polytekniske Læreanstalt, Danmarks tekniske Højskole

Lyngby 1988

INDHOLDSFORTEGNELSE

FORORD

Rapporten behandler de muligheder, der findes for at få rimelige "skærmkopier" ud fra skærmbilledet, når datamaten er en QL og printerne er billige 8 x 8 matrixprintere.

Rapporten kræver et godt kendskab såvel til printernes muligheder som til skærmbilledets opbygning.

IFH-rapport 177, Rumgitter på QL, som jeg udarbejdede i 1986, giver en del af forklaringen.

Nævnte rapports pag. 102, SKÆRMOPBYGNING, gengives på næste side.

Herudover kan henvises til printer-manualerne og til IFH-rapporterne 178 og 179.

20. april 1988

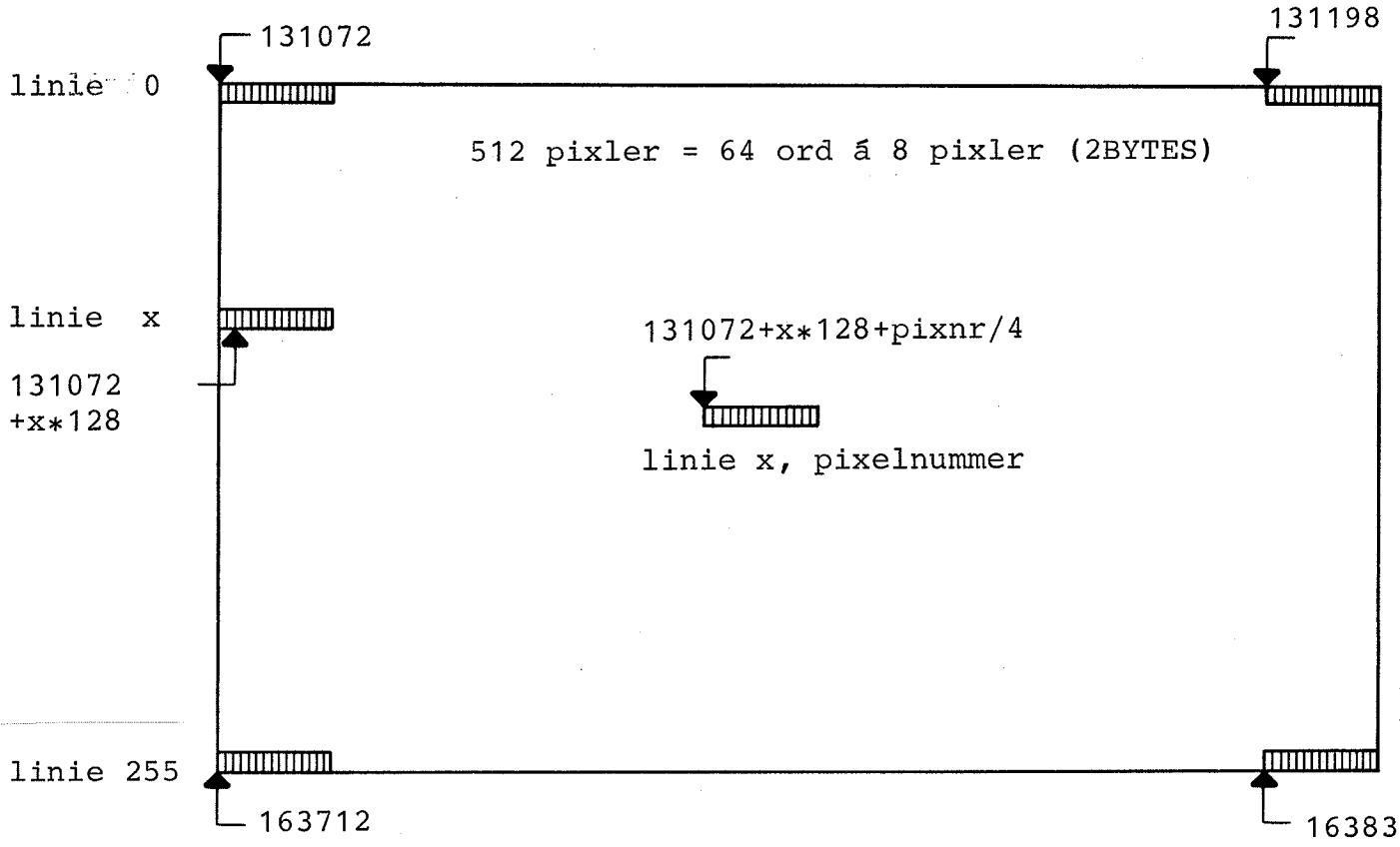
Johs. F. Munch-Petersen

De beskrevne programmer benytter:

- SINCLAIR QL 512 K, dansk udgave
- TOOLKIT 2, V2.05 1986
- En diskette-station
- CUB microvetic grafik-skærm
- SINCLAIR QL-printer eller
- SHINWA CPA80S-printer (begge til ser1)
- TURBO kombiler, Version 2, DIGITAL PRECISION

IFH-rapporter om QL-datamaten:

177 RUMGITTER På QL. En introduktion til Grafik PÅ QL.
178 BRUGERDEFINEREDE KARAKTERER På QL.
179 DOWNLOAD -KARAKTERER På QL/CPA80S - printeren.
180 LINEÆR REGRESSION PÅ QL. I, Vejledning.
181 LINEÆR REGRESSION PÅ QL. II, Programudskrift.
182 ENSIDET VARIANSANALYSE PÅ QL.
185 STORE, AFBILDNINGSTRO PRINTERKOPIER AF SKÆRMBILLEDER PÅ QL



SKÆRMOPBYGNING

(evt. POKE á 8 pixler)

G	G	G	G	G	G	G	G	n *256
R	R	R	R	R	R	R	R	y

128,64,32,16, 8, 4, 2, 1 binære

G = grøn R = rød
G+R= hvid 0+0= sort

Eksempel:
grøn-rød-sort-hvid + 4 sorte
POKEs med 144 * 256 + 80

(163840)

8 PIXLER

8 pixler 16 bit information (=2 BYTES)

```
SBYTES navn,adresse,2
POKE_W adresse, n*256+y
```

pixelnummeret skal være deleligt med 8. (pixnr/4 skal være et lige tal)

HELE SKÆRMEN

512*256 pixler, på 512*256 / 8 adresser (á 2 BYTES)

IALT 512*256 / 8 * 2 BYTES = 32768 BYTES

```
SBYTES navn,131072,32768
```

SKÆRMENS "pixler"

Skærmen har 512 pixler vandret, 256 pixler lodret.

En pixel er et rektangel med højde-/længde-forholdet 1.355.

Prøv fx LINE 0,0 TO 100,135.5. Resultatet bliver en ret linie, der tydeligt er dannet af en række pixler uden de spring eller "afrundingsfejl", de fleste skrå linier viser. Linier med hældningen tangens = 1.355/n er regelmæssige.

Bruges grafiske kommandoer, vil et kvadrat fx. vise sig at bestå af 271 pixler vandret og 200 pixler lodret.

Dette pixelmønster kan ikke uden videre overføres til printerens "dots", som i øvrigt heller ikke er anbragt i et kvadratisk mønster.

Skærmkopier, hvor enkelte "pixler" omdannes til enkelte "dots" bliver aldrig særligt pæne.

Dyre printere, med mange nåle, kan give acceptable resultater, men ikke gode, når udgangspunktet er QL's skærm med en begrænset opløsningsevne.

De bedste printere, fx. Laser-printerne, kan give god grafik, men den dannes som på en plotter - ved "grafiske" kommandoer til printeren, ikke ved skærmkopiering.

PRINTERENS "dots"

Printerens prikker - dots - dannes af en række nåle. På CPA80S og QL-printerne er der 8(9) nåle lodret over hinanden.

Bedre printere har flere nåle, fx. 16.

I det følgende tages udgangspunkt i CPA80S-printeren.

QL-printeren minder meget om den, har næsten de samme kommandoer, men kan udskrive i NLQ, Near Letter Quality. 4 gange langsommere. Til gengæld har den ikke download-faciliteter for brugerdefinerede karaktersæt.

Printeren udskriver, i vandrette linier
- enten almindelige karakterer.
- eller grafiske karakterer.

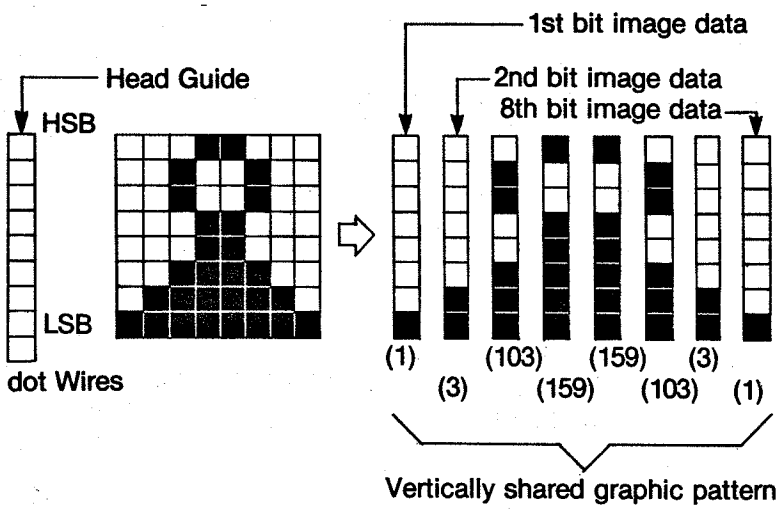
DE ALMINDELIGE KARAKTERER er opbygget over en 8 x 8 matrice (matrix). Der kan tilføjes en kode, så tegnet flytter en (nål/dot) ned. Dette gælder fx. g, j, y, m.fl., der benytter de nederste 8 nåle.

Brugeren kan også programmere sig frem til karakterer - på QL'en dog kun via grafiske karakterer, se fx. IFH-rapport 178 , Bruger-definerede Karakterer på QL.

På CPA-80S printeren kan karakterer downloades i printerens hukommelse og senere benyttes til udskrift, ved skift mellem "dansk" og "downloaded" karaktersæt. Se fx. IFH-rapport 179, Downloadkarakterer på QL, hvor jeg viser en række eksempler.

Hver af de 8 søjler, en karakter er opbygget af, er defineret ved et tal mellem 0 og 255, der angiver, hvilke af de 8 mulige nåle, der skal aktiveres: Værdi 1,2,4...64,128 nedefra og op. Summen af værdierne er søjlens "kommando" til printeren. "127" vil fx. ikke aktivere den øverste nål (128), men de syv nedenunder (1...64, sum 127).

Se nedenstående figur 15 fra CPA80S-manualen.



DE GRAFISKE KARAKTERER omfatter kun een søjle (8 dots i samme lodrette række).

Afhængigt af, hvor mange "dots per line" printeren sættes til, kan der opnås forskellige detaljeringsgrader. Se manualerne, hvor der også redegøres for, at man ikke kan bruge alle nåle i hver søjle, hvis der vælges mange "dots per line".

PRINTERENS ARBEJDSFELT er:

Vandret: Typisk 80 bogstaver à 8 dots, 640 dots på 203 mm (8").

Lodret: Med linieafstand 24/216 = 1/9" kan der printes kontinuerligt i lodret retning. Denne linieafstand skal altså vælges ved skærmkopiering.
Der fås 108 linier à 8 dots, 864 dots på 305 mm (12").

KOPI-MULIGHEDER

A. EN MINDRE KOPI På TVÆRS AF PAPIRET,
med mulighed for normalt printet figurtekst.

Tager vi som eksempel et kvadrat på skærmen fås:

Lodret: 200 pixler som giver:
200 x 305/864 = 71 mm.

Vandret: 271 pixler som giver:

a. CPA80S med 640 dots per line.
271 x 203/640 = 86 mm (+20% fejl).

b. QL med 640 dots per line (langsom)
271 x 203/960 = 57 mm (-20% fejl).

c. QL har også 576 og 720 dots per line,
der giver henholdsvis 33% og 8% fejl.

d. CPA80S med grafikkarakterer, som slås sammen 8 og 8 til download-karakterer (idet matrixen "drejes" 90°).
Udskrevet i ELITE fås da 96 karakterer à 8 pixler = 768 dots.
271 x 203/768 = 72 mm (1% fejl).
Et sådant program er omtalt senere, men er langsomt.

IFH's ALMINDELIGT BENYTTEDE MASKINKODEPROGRAMMER HAR:

"scopy" til CPA80S-printeren: 20% fejl som a. ovenfor.
"qcopy" til QL-printeren: 8% fejl som c. ovenfor.

B. EN STOR KOPI VENDT 90°,
udnyttende A4-formatet.

Som det vil ses, fungerer ideen udmærket på CPA80S-printeren, med 6 minutter printetid. På QL-printeren tager det desværre 11 minutter (TURBO-kompileret).

Skaleringsfejlen er ~ ½%, men der kan maximalt udnyttes 400(430) x 213 pixler, dog 512 x 213, hvis der printes over perforeringen.

C. EN STOR, RETVENDT KOPI,
forudsat, at der kun skal kopieres ca. 300 pixler vandret.

SMÅ RETVENDTE KOPIER

Eksemplet "experiment" viser, at den ovenfor under A.d. omtalte mulighed er gennemførlig (pag. 37). Brugen af gentagne download af et analyseret bogstav, efterfulgt af bogstavets udprintning giver en langsom PROCedure, godt 2 minutter for et stort skærmbillede, i TURBO-kompileret udgave. Denne tid vil blive (lidt) forøget, hvis der skal kunne kopieres vilkårlige dele af skærmbilledet.

"Experiment" er afprøvet uden de (op til fire) IF-sætninger, der kræves, hvis skærmbilledet ikke starter og slutter, lodret og vandret med hjørnekoordinater og længder, bredder, der er multipla af 8.

STORE KOPIER, DREJET 90°

Uanset hvordan skærmen skal kopieres, er det nemmest at anvende læsning af QL'ens eget skærmlager.

Som vist pag. 4, starter billedet i adresse 131072(2^7). Her kan læses en byte, d.v.s. 8 bits for de første 8 pixler, der angiver (1 eller 0) om pixlen er hvid-grøn eller rød-sort. Næste byte, i 131073, angiver tilsvarende (1 eller 0) om det hvid-grønne er hvidt eller grønt, hhv. om det rød-sorte er rødt eller sort.

D.v.s. at hvis der benyttes sort skærm (0-0), vil alle de ønskede stregfarver vise 1 i enten den første eller den anden byte. Hvid = 1-1, rød = 0-1, grøn = 1-0.

Læses disse adresser, begyndende i billedets nederste, venstre hjørne og opefter i kolonner à 8 pixler, kan informationen direkte benyttes til en skærmkopi, men en lille en, og tilmed drejet 90°.

Skal der opnås en stor kopi, må informationen gentages i udprintningen.

Princippet kan illustreres således, hvor de 8 bits, aflæst fra skærmen, tænkes at angive hvid-grøn-rød (1) eller sort (0), udledt af læsning af 2 bytes:

Skærm, aflæst	Printer information						
0110111	0		0		0		
	0		0		0		
8 pixler,	1		1		1		1. linie
vandret	1	+	1	+	1		med tre ens
	1		1		1		grafiske tegn
	1		1		1		
	0		0		0		
	0		0		0		
	0		0		0		
	0		0		0		
	1		1		1		2.linie
	1	+	1	+	1		med tre ens
	1		1		1		grafiske tegn
	1		1		1		
	1		1		1		
	1		1		1		

Antallet af pixler er blevet til det dobbelte antal dots (vandret på skærm, lodret på printer), og tilsvarende 3-dobling i (lodret/vandret retning).

Med kvadratet fra før som eksempel fås nu:

271 vandrette pixler = 542 lodrette dots d.v.s. 542 x 305/864 = 191 mm.

200 lodrette pixler = 600 vandrette dots d.v.s. 600 x 203/640 = 190 mm.

Dette giver en næsten afbildningstro, stor printerkopi.

Både CPA80S- og QL-printerne har 640 dots per line.

Printetid 5 minutter hhv. 11 minutter.

Se eksemplerne "cp80skærm", "qlskærm", "cp80kopi" og "kalkkopi".

GENERELT

Printetiden er en afgørende faktor for mange brugere. I mange tilfælde er det ligegyldigt om cirkler bliver elipser osv. I hvert fald synes "business-grafik" at være salgbart, selvom "lagkagen" ikke er cirkulær. Noget så mærkeligt er næppe acceptabelt for ingeniører -og slet ikke, hvis billedet viser en genstand, fx. et betonelement, hvor dimensionsforvrængning giver et forkert indtryk, fx. af de statiske muligheder.

De kommercielle skærmkopi-programmer til QL er hurtige, men giver forvrængning, og kopierne er små. 1 pixel giver een dot. Disse kopier anvendes som "huskeseddel", til orientering i mine programmer omtalt i IFH-rapporterne 177 og 180/181, hvor det endelige grafiske resultat er en plottertegning.

I denne rapport er der gjort forsøg på få grafisk rimelige "arbejdskopier" på printeren.

Det er ærgeligt, at dyre printere med høj opløsningsevne ikke kan kopiere en skærm uden speciel hokuspokus. I reglen kopieres - som vist i denne rapport - skærmpixlerne blot i "forstørret udgave", fx. som en 2x3 dot stor blok.

Til gengæld øges printetiden med en faktor ca. 4 i forhold til udskrift med PICA, når der bruges grafiktegn. På QL-printeren tager en side A4, med 1/9" linieafstand, 110 sekunder med PICA, 550 sekunder med grafiktegn - en faktor på 5.

Imidlertid: Hvis grafik blev defineret i et rastermønster, så detaljeret at det kunne udnyttes af de bedste printere, ville informationsmængden om "prikker" blive 10- til 20-doblet. Der ville ikke være problemer med at lave algoritmer, der "afrundede" til den almindelige, grove skærms raster, - men analyse, SAVE og LOAD etc. af den fulde information ville give lagerproblemer og tage tid.

Den fremtidige CAD må bygge på fuld kompatibilitet mellem en grov skærm og en nøjagtig plotter/dyr printer.

Det er måske den største hindring for udviklingen af et avanceret system. Lager- og tidsproblemerne vil udviklingen løse på få år.

På IFH's QL-computere må vi imidlertid se på de praktiske muligheder idag:
"scopy" og "qcopy", vor normale, indkøbte skærmkopi-software laver små, grimme, men læselige kopier på ca. 20 sekunder.

De af mig, i denne rapport, foreslåede programmer tager 2-6 minutter, selv om de er kompilerede. De er alle større og mere læselige, de fleste (næsten) afbildningstro.

Kan denne tidsforøgelse reduceres ?

Jeg har lavet et eksperiment med det senere viste program "kalkkopi". Som demonstreret tager det 353 sekunder, ca. 6 minutter. Det bygger på PRINT af grafikkarakter, opbygget ved 2- og 3-dobling af pixlerne lodret/vandret og har en afbildningsfejl (se cirklen) på ca. ½%. Udskriver jeg et tilsvarende printerareal med præ-definerede grafiktegn, er printetiden 345 sekunder. Regnetiden medfører altså kun 8 sekunders forsinkelse, kompileret programudgave (men måske 5 minutter i SUPERBASIC). Udskrift af normale bogstaver på det tilsvarende papirareal tager 91 sekunder! Tidsforbruget er altså afhængigt af printerens evne til at printe "grafiktegn" i stedet for "normaltegn", ikke om kompileren eller programmet er mere eller mindre fremragende.

Dette gælder CPA80S-printeren. QL-printeren bruger generelt den dobbelte tid på at printe grafiktegn! Her er programtiden uden betydning.

Imidlertid er der i dag mange, også billige, printere, der skriver væsentligt hurtigere.

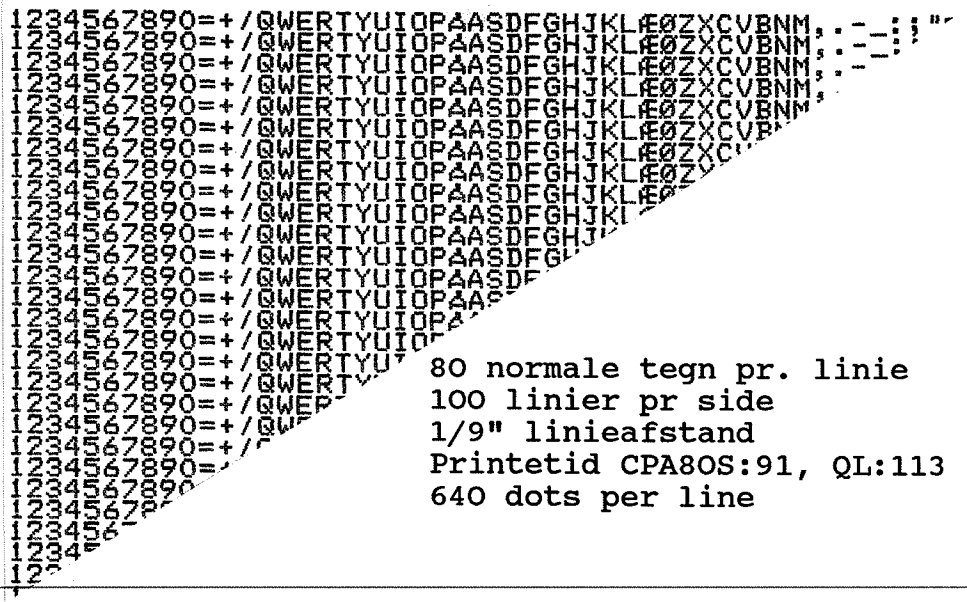
De her demonstrerede programmer er altså på længere sigt ikke gode nok. De må omarbejdes til maskinkode. Hvor meget dette vil hjælpe, ved jeg ikke. Det er trods alt store flader, der skal udprintes i de fleste af de viste eksempler.

Der er også vist et eksempel ("experiment" med "scrcopy"), der giver en lille kopi, af størrelse som de kommercielle (pag. 37).

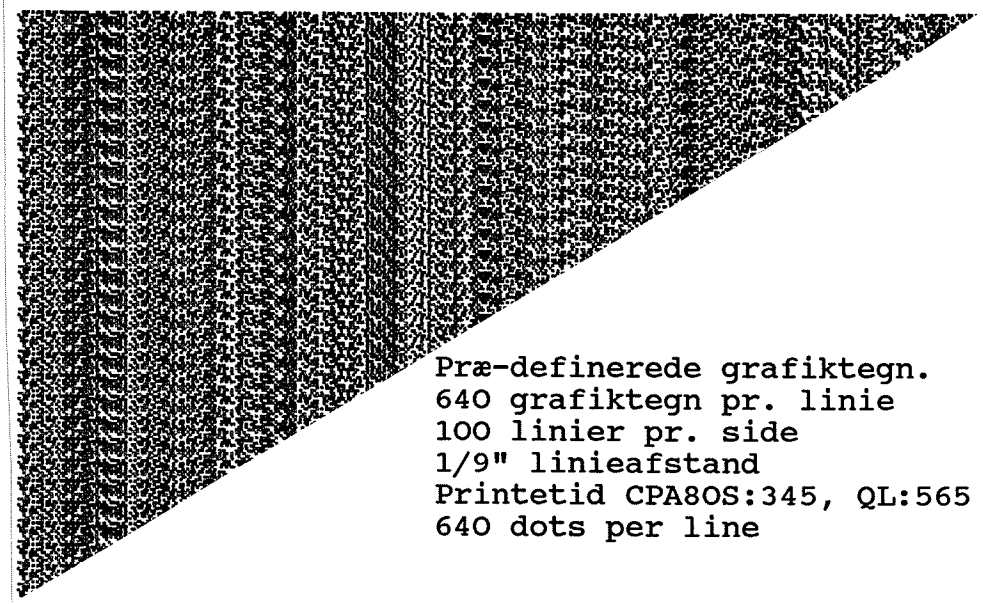
Printetiderne er:

"scopy" 65 sekunder (21% fejl, CPA80S-printer)
"qcopy" 112 sekunder (8% fejl, QL-printer)
"scrcopy" 139 sekunder (1% fejl, CPA80S-printer)

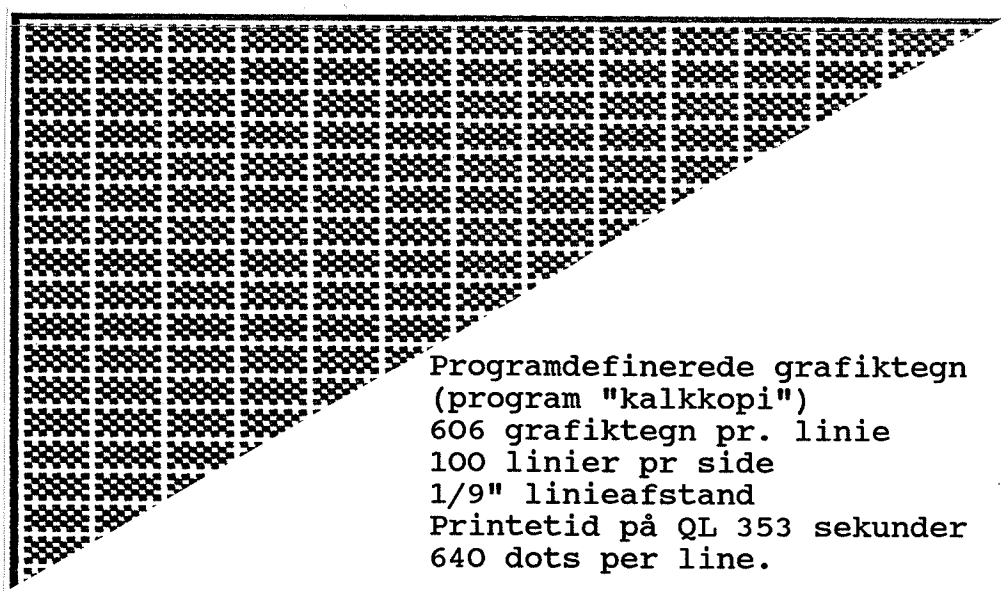
Denne fordobling af printetiden er vel acceptabel, hvis afbildningsfejlen skal reduceres.



80 normale tegn pr. linie
100 linier pr side
1/9" linieafstand
Printetid CPA80S:91, QL:113 sekunder
640 dots per line



Præ-definerede grafiktegn.
640 grafiktegn pr. linie
100 linier pr. side
1/9" linieafstand
Printetid CPA80S:345, QL:565 sekunder
640 dots per line



Programdefinerede grafiktegn
(program "kalkkopi")
606 grafiktegn pr. linie
100 linier pr side
1/9" linieafstand
Printetid på QL 353 sekunder
640 dots per line.

Program til præ-definerede grafiktegn, vist på modstående side i uddrag:

```
100 BAUD 4800: TRA 1 : OPEN#3,ser
110 DIM v$(606)
120 FOR n= 1 TO 606: v$ = v$ & CHR$(RND(1 TO 255))
130 BPUT#3;27,51,24
140 tid = DATE
150 FOR n=1 TO 100: BPUT#3;27,75,94,2: PRINT#3;v$
160 BPUT#3;27,51,36
170 PRINT#3;\ DATE - tid: CLOSE#3
```

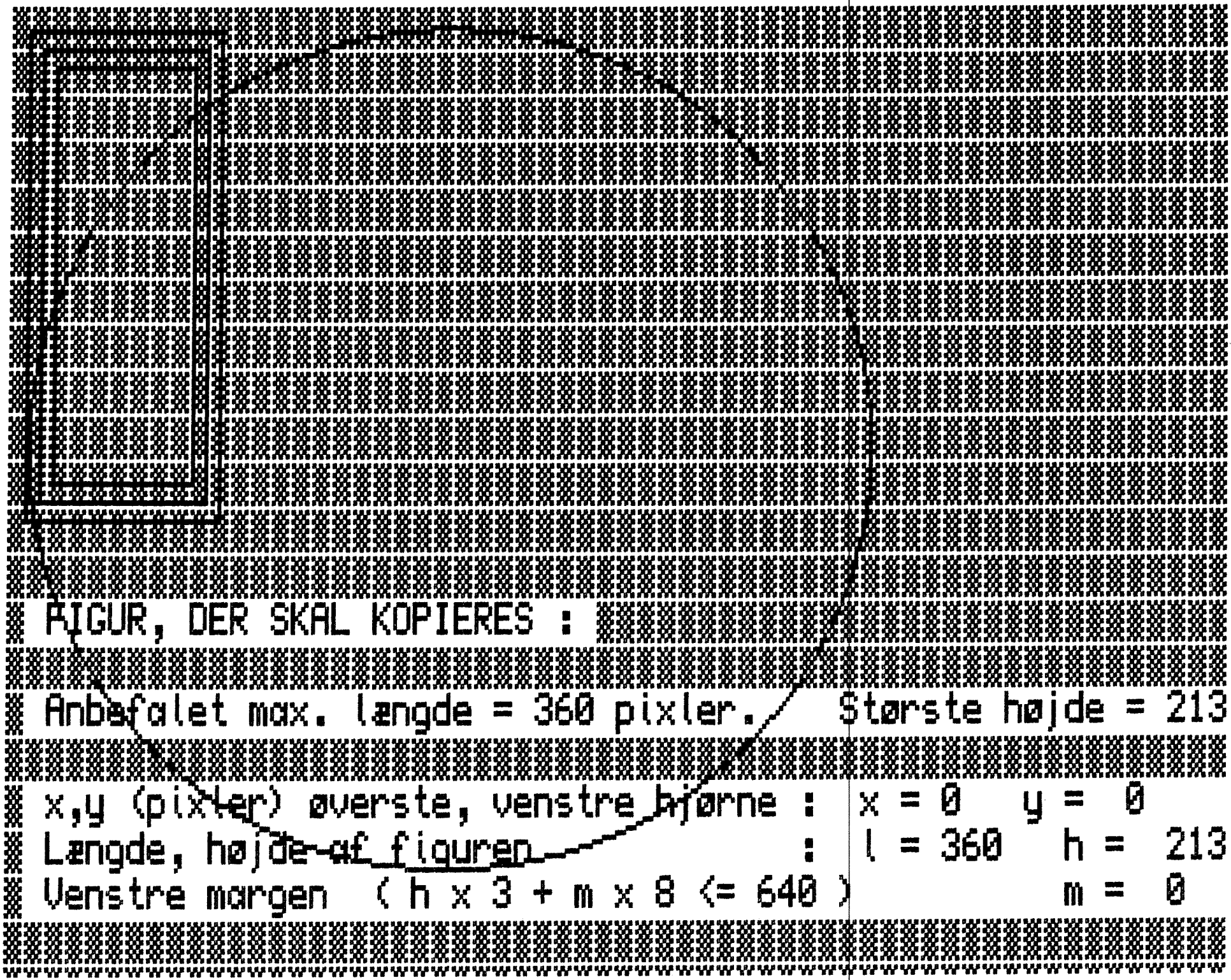
Programmerne er kompileret med TURBO, enkelte med TURBO II, der er lidt hurtigere.

De er optimeret for "SIZE", idet jeg finder, at de bliver ret store, hvis de optimeres for "SPEED", uden at hastigheden øges væsentligt. "SPEED" er især gavnlig ved heltals-beregninger, der ikke kan anvendes på QL, da skærmadresserne er større end 32768.

Programmerne køres som EW-programmer fra SUPERBASIC-hovedprogrammet. BASIC-programmet venter altså på, at skærmkopien er udført, før det fortsætter, fx. med udskift af tekst/tabeller/DATA/forklaring eller lignende til figuren.

Programmerne kan også køres som EX-programmer som vist med "cp80kopi". I så fald fortsætter BASIC-programmet, og der må indlægges en ventefunktion, så brugeren ikke kan starte en ny skærmkopi med den samme EX-kommando, før den forrige kopi er afsluttet. Se eksemplet.

EKSEMPLER


FIGUR, DER SKAL KOPIERES :
Anbefalet max. længde = 360 pixler.
Største højde = 213
x,y (pixler) øverste, venstre hjørne :
x = 0 y = 0
Længde, højde af figuren :
l = 360 h = 213
Venstre margen (h x 3 + m x 8 <= 640)
m = 0

EKSEMPEL "cp80skærm"	STOR KOPI, DREJET 90°

Maximal højde = 213 pixler.
Vist bredde = 360 pixler. Kan være 512 henover perforeringen.

Der kan angives vilkårlige x,y (øverste hjørne)
og ℓ,h (længde, højde).

5 minutter for det viste format.

Afbildningsfejl ~ ½%

I praksis kræves der af hovedprogrammet.
(Sammenlign "cp80skærmboot", næste side):

- FORMATtering af en ram3_4 (boot, linie 130-140)

- PROCedurekald skærmkopi x,y,ℓ,h,m, anbragt de rigtige steder.

- PROCeduren "skærmkopi (boot, linie 500-580),
  der starter det kompilerede program, EW cp80skærmex.

```
100 REMark cp80skærmboot
110 INPUT \' RAM LOADed  J / N  '; y$
120 IF y$== 'j' : GO TO 160
130 TK2_EXT : BAUD 4800 : TRA 1 : DATA_USE flp1_ : PROG_USE flp1_
140 LRESPR flp1_ram_ver1 : FORMAT ram3_4

160 WINDOW    512,256,  0,  0: PAPER   0: INK   7 : CLS
170 WINDOW#2; 436,202,  0,  0: PAPER#2;0: INK#2;4
180 WINDOW#0; 436, 54,  0,202: PAPER#0;0: INK#0;7
190 FOR n= 0 TO 24: INK 2 + n * ( n < 6 ): PRINT FILL$(CHR$(27),83)
200 FOR n= 0 TO 8 STEP 4: BLOCK 42+n*2,1,14-n,14-n,7
210 FOR n= 0 TO 8 STEP 4: BLOCK 42+n*2,1,14-n,105+n,7
220 FOR n= 0 TO 8 STEP 4: BLOCK 1,92+n*2,14-n,14-n,7
230 FOR n= 0 TO 8 STEP 4: BLOCK 1,92+n*2,55+n,14-n,7
240 INK 7 : AT 13,1 : PRINT ' FIGUR, DER SKAL KOPIERES : '
250 AT 15,1 : PRINT ' Anbefalet max. længde = 360 pixler.    ';
260 PRINT 'Største højde = 213 pixler. '
270 AT 17, 1 : PRINT ' x,y (pixler) øverste, venstre hjørne :';
280 PRINT FILL$(' ',29): AT 17,42 : INPUT 'x = '; x ; '   y =  '; y
290 AT 18, 1 : PRINT ' Længde, højde af figuren             :';
300 PRINT FILL$(' ',29): AT 18,42 : INPUT 'l = '; l ; '   h =  '; h
310 AT 19, 1 : PRINT ' Venstre margen  ( h x 3 + m x 8 <= 640 )';
320 PRINT FILL$(' ',27): AT 19,50 : INPUT '  m =  '; m
330 CIRCLE 38,62,36

350 tid = DATE
360 skærmkopi x, y, l, h, m
370 OPEN#3;ser: PRINT#3; \\ ; DATE - tid : BPUT#3;12 : CLOSE#3

500 DEFine PROCedure skærmkopi ( x, y, l, h, m )
510   IF x < 0 OR y < 0 OR x + l > 512 OR y + h > 256 THEN
520     CLS#0 : PRINT#0; \' OUT OF RANGE' : STOP
530   END IF
540   IF h * 3 + m * 8 > 640 : CLS#0 : PRINT#0; \' OUT OF RANGE' : STOP
550   OPEN#3,ser: BPUT#3; 27,108,m,27,81,80,27,67,72,27,78,10: CLOSE#3
560   DELETE ram3_xylh
570   OPEN_NEW#10; ram3_xylh
580   PRINT#10; x: PRINT#10; y: PRINT#10; l: PRINT#10; h : CLOSE#10
590   TRA 0 : EW cp80skærmex : TRA 1
600 END DEFine skærmkopi
```

linie 130 ff	LRESPR(RESPR,LBYTES,CALL), FORMAT osv. kun een gang
linie 190 ff	Vindue med CHR$(27)-tegn, rektangler, cirkel
linie 360	PROCedurekald.
linie 500 ff	PROCeduren "skærmkopi" (x,y,ℓ,h,m) hvor x,y er pixelkoordinater til øverste, venstre hjørne og ℓ,h er (pixel-)længde og højde af billedet og m er margin.
linie 570 f	x,y,ℓ,h sendes til RAM, så "cp80skærmex" kan finde disse værdier.

```
10000 REMark cp80skærm_bas til cp80skærmex
10010 DIM k(4), v$(640), w$(640), a$(16) : start= 0: slut= 0
10020 n= 0: hob= 0: lob= 0: korr1%= 0: korr2%= 0: korr3%= 0
10030 OPEN#10; ram3_xylh : FOR n = 1 TO 4 : INPUT#10; k(n)
10040 CLOSE#10
10050 hob = k(4) * 3 DIV 256 : lob = k(4) * 3  MOD 256
10060 korr1% =   k(1) MOD 8           : xk = k(1) - korr1%
10070 korr2% = ( k(1) + k(3) ) MOD 8  : lk = k(1) + k(3) - korr2%
10080 korr3% = ( 256 - k(2) - k(4) ) * 128
10090 start  = 163712 + xk / 4 - korr3%
10100 slut   = 163712 + lk / 4 - korr3% - 2 * ( korr2% = 0 )
```

GENERELLE PRINCIPPER: Skærmudsnittet læses fra nederste, venstre hjørne og opad, linie for linie, for 8 pixler ad gangen. Informationer omdannes til grafiske tegn, som printeren udskriver til en afbildning, der er drejet 90° i forhold til skærmbilledet.
Se næste sides PROCedurer.

linie 10000 BASIC der TURBO-kompileres til "cp80skærmex"

linie 10010 k(4) er x,y,ℓ,h, indlæst fra RAM i linie 10030. 640 er det maximale antal dots pr. linie på printeren. Sammenlign linie 540 i boot. a$(16) bruges som binær streng til "fordobling" af en læst byte, se variablen linie næste side.

linie 10050 hob og lob er high/low order byte, som printeren får linie 10620-10630.

linie 10060 korr1% og korr2% registrerer de afvigelser, der er til venstre og højre for pixel-værdier (x, x+ℓ), der er multipla af 8.

linie 10080 korr3% er afstanden fra nederste skærmkant til nederste, venstre hjørne af det ønskede skærmudsnit. Nederste, venstre hjørne af skærmen har adressen 163712. En pixellinie optager 128 adressennumre (bytes), idet linien indeholder 2 bytes for hver 8 pixler (512/8 x 2 = 128).

linie 10100 Sidste led korrigerer, så slut bliver korrekt for korr2% = 0. Sammenlign linie 10370 og 10410.

Se endvidere pag. 53 om G_SAVE m.v.

```
10200 OPEN#3,ser1
10210  PRINT#3; CHR$(27); CHR$(51); CHR$(24);
10220  liniestart
10230  PRINT#3; CHR$(27); CHR$(51); CHR$(36);
10240 CLOSE#3

10300 DEFine PROCedure liniestart
10310  LOCal linie, ast%, bst%, bsl%
10320  linie = start : v$ = '' : w$ = ''
10330  ast% = 2^(7 - korr1%) : bst% = 2 + 2 * korr1% : bsl% = 16
10340  kopilinie
10350  IF korr1% + k(3) < 9 :                RETurn
10360  IF korr1% + k(3) < 9 + korr2% :       GO TO 10410
10370  FOR linie = start + 2 TO  slut - 2    STEP 2
10380   v$ = '' : w$ = '' : ast% = 128 : bst% = 2 : bsl% = 16
10390   kopilinie
10400  END FOR linie
10410  linie = slut : v$ = '' : w$ = ''
10420  ast% = 128 : bst% = 2 : bsl% = 16 : IF korr2% : bsl% = 2 * korr2%
10430  kopilinie
10440 END DEFine liniestart

10500 DEFine PROCedure kopilinie
10510  LOCal byte, a%, b%, byte1%, byte2%
10520  FOR byte = linie TO linie - ( k(4) - 1 ) * 128   STEP -128
10530   a% = ast% : b% = bst% : a$ = FILL$('0',16)
10540   REPeat omdan
10550    IF a% && PEEK(byte) OR a% && PEEK(byte+1): a$(b%-1 TO b%)= '11'
10560    a% = a% / 2 : b% = b% + 2    : IF b% > bsl% : EXIT omdan
10570   END REPeat omdan
10580   byte1% = BIN(( a$(1 TO 8))) : byte2% = BIN(( a$(9 TO 16)))
10590   v$ = v$ & CHR$(byte1%) & CHR$(byte1%) & CHR$(byte1%)
10600   w$ = w$ & CHR$(byte2%) & CHR$(byte2%) & CHR$(byte2%)
10610  END FOR byte
10620  PRINT#3; CHR$(27); CHR$(75); CHR$(lob); CHR$(hob); : PRINT#3; v$
10630  PRINT#3; CHR$(27); CHR$(75); CHR$(lob); CHR$(hob); : PRINT#3; w$
10640 END DEFine kopilinie
```

linie 10210 Linieafstand 1/9" til grafik (24/316).

linie 10230 Linieafstand 1/6" som normalt.

linie 10300 PROCedure liniestart.
Se også teksten pag 9.

linie 10320 Der begyndes forneden til venstre = start.

linie 10330 I første omgang læses de første 8 pixler langs venstre billedkant, nedefra og op.

korr1% angiver, fra hvilken pixel, der skal dannes grafiske tegn, startende med "værdien" ast% og "startnummer" bst%. Der skal sluttes med bsl% = 16. Se linie 10520-10570.

---

linie 10370 Dernæst læses "midt i billedet", nedefra og op. Alle 8 pixler, skal bruges, ast% = 128, bst% = 2, bsl% = 16.

linie 10410 Til slut læses langs højre skærmkant. Her skal de sidste pixler måske ikke bruges.

linie 10420 ast% = 128 og bst% = 2 og korr2% sætter afslutningsværdien af bsl%.

linie 10350 Hvis der kun skal læses de første 8 pixler.....

linie 10360 Hvis der kun skal læses 16 pixler.....

linie 10500 PROCedure kopilinie.

linie 10520 Nedefra og op. En skærmbredde er 512 pixler, dvs. 64 x 8 pixler, hvor hvert sæt à 8 pixler kræver 2 bytes, ialt 64 x 2 = 128.

linie 10530 Startværdierne a% og b% sættes, og a$ tildeles 16 nuller.

linie 10550 Hvis (hvid/grøn) eller rød, så skal a$ have to 1-taller på de relevante pladser.
Se QL-vejledningen om den logiske operator &&.

linie 10560 Værdien af a% halveres (128,64.....2), og b% tælles op (2,4.....16).

linie 10580 BIN(a$()) omdanner den "binære streng" til en byte.

linie 10590 Strengen v$ tildeles 3 gange det grafiske tegn, der udskriver en fordobling af de fire første pixler, w$ tilsvarende de fire sidste.

linie 10620 Til slut udskrives to linier på printeren (2 x 8 dots), svarende til en lodret skærmkolonne, 8 pixler bred.

Antal operationer per pixel vid
enbyteladdad max. 8bpp pixelyta, 5 parter
x-led (pixlar) per rad, varav 1 parter
Lagrade halvbytes (nibbles) per rad
Lagrade medelvärden per rad
x = 1 5 10 20 50 100 200 500 1000
h = 2 3 4 5 6 7 8 9 10
n = 8 (5 40)

EKSEMPEL "qlskærm"	STOR KOPI, DREJET 90°

Som "cp80skærm", dog er QL-printeren langsommere, 11 min ! mod CPA80S-printerens 5 minutter.

Udskriften er vist på modstående side.

På næste side vises en QL-kopi, som fordobler på begge leder og benytter 480 dots per line.

Den giver 12% fejl og tager 7 minutter.

FORSKELLENE MELLEM "qlskærm" OG "cp80skærm" ER:

```
boot:     linie 130: BAUD 4800 og TRA 1 udgår.
          linie 590: TRA 0     og TRA 1 udgår.

qlskærm:  linie 10620 og 10630:
          CHR$(27); CHR$(42); CHR$(4), CHR$(lob); CHR$(hob);
          som er ESCAPE-sekvensen for 640 dots per line.
```

boot hedder "qlskærmboot".
BASIC-programmet hedder "qlskærm_bas".
Det kompilerede program hedder "qlskærmex".

EKSEMPEL "qlskærm"	STOR KOPI, DREJET 90°

Se omtalen på forrige side.

Udskriften vises til venstre.

Bemærk, at der er udskrevet et skærmbillede, der er 240 pixler højt = 480 dots på en printerlinie, på knap 8 minutter.

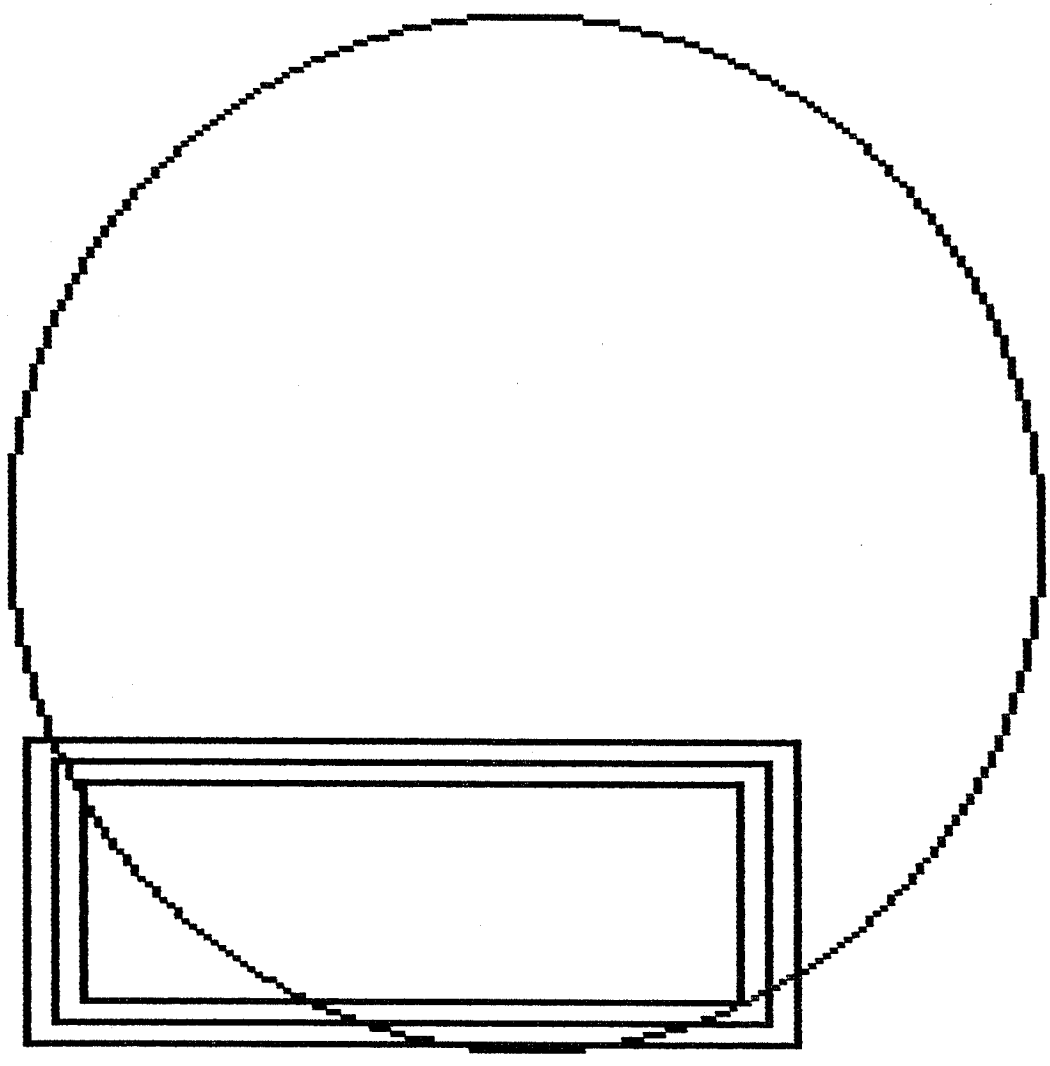
Der er 12% fejl på afbildningen.

Med dette program kan næsten hele skærmen kopieres, hvis man kopierer henover perforeringen: 512 x 240 pixler, 1024 x 480 dots.

FORSKELLENE MELLEM "ql2skærm" og "cp80skærm" er:

```
boot:          linie 130:   BAUD 4800 og TRA 1 udgår
               linie 590:   TRA 0 og TRA 1 udgår:

ql2skærm:      linie 10010: DIM v$(480), w$(480)
               linie 1000: ....k(4) * 2........k(4) * 2
               linie 10590: ....v$ & CHR$(BYTE1%) & CHR$(BYTE1%)
               linie 10600: tilsvarende "fordobling" for w$
               linie 10620 og 10630: Ingen ændring.
               CHR$(27); CHR$(75), som cp80skærm, da  CPA80S-
               printeren giver 640, QL-printeren 480 dots per
               line med CHR$(75).
```

boot hedder "ql2skærmboot".
BASIC-programmet hedder "ql2skærm_bas".
Det kompilerede program hedder "ql2skærmex".



EKSEMPEL "kalkkopi"

Stor kopi, drejet 90°, specielt udarbejdet til stud. polyt. Anders Kalko's eksamensprojekt, der udelukkende kopierer fra en skærm på 400 x 202 pixler, som lige netop kan være på A4.

Programmet forenkles væsentligt.

Printetid knap 6 minutter, dvs. at forenklingen ikke spiller nogen tidsmæssig rolle: "cp80skærm" bruger godt 5 minutter på 360 x 213 pixler.

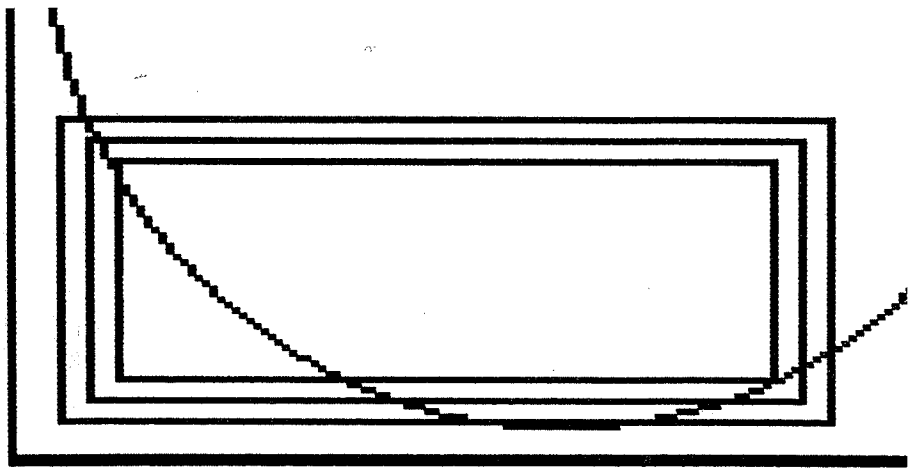
Det er printeren selv, der bestemmer tempoet.

Udskriften er en prøve-udskrift, der intet har med A.K.s projekt at gøre.

Bemærk:
Her er ikke benyttet BORDER 1,7, som giver een pixel på vandrette, 2 pixler på lodrette linier, men LINE (1 pixel bred), bagefter "beskyttet" (med CLS) af en usynlig BORDER 1, se linie 170 i programmet, næste side.

```
100 REMark kalkkopiboot
110 TK2_EXT : BAUD 4800 : TRA 1 : DATA_USE flp1_ : PROG_USE flp1_

130 WINDOW    512,256,  0,  0: PAPER   0 : CLS
140 WINDOW#2; 436,202,  0,  0: PAPER#2;0 : INK#2;4
150 WINDOW#0; 436, 54,  0,202: PAPER#0;0 : INK#0;7
160 WINDOW    400,202,  0,  0: INK     2 : CLS
170 LINE 0,0 TO 146.5,0 TO 146.5,100 TO 0,100 TO 0,0 : BORDER 1
180 INK 7: FOR n= 1 TO 20: PRINT FILL$(' ',36);FILL$(CHR$(27),30)
190 FOR n= 0 TO 8 STEP 4: BLOCK 42+n*2,1,14-n,14-n,2
200 FOR n= 0 TO 8 STEP 4: BLOCK 42+n*2,1,14-n,105+n,2
210 FOR n= 0 TO 8 STEP 4: BLOCK 1,92+n*2,14-n,14-n,2
220 FOR n= 0 TO 8 STEP 4: BLOCK 1,92+n*2,55+n,14-n,2
230 INK 4: CIRCLE 38,62,36: INK 7

250 skærmkopi
```

```
500 DEFine PROCedure skærmkopi
510   OPEN#3,ser: BPUT#3; 27,108,2,27,81,80,27,67,72,27,78,0: CLOSE#3
520   TRA 0 : EW kalkkopi : TRA 1 : OPEN#3,ser : BPUT#3;12 : CLOSE#3
530   REMark Reset venstre, højre margen og skip over til det normale
540 END DEFine skærmkopi
```



linie 170
LINE giver en pænere kantbegrænsning.

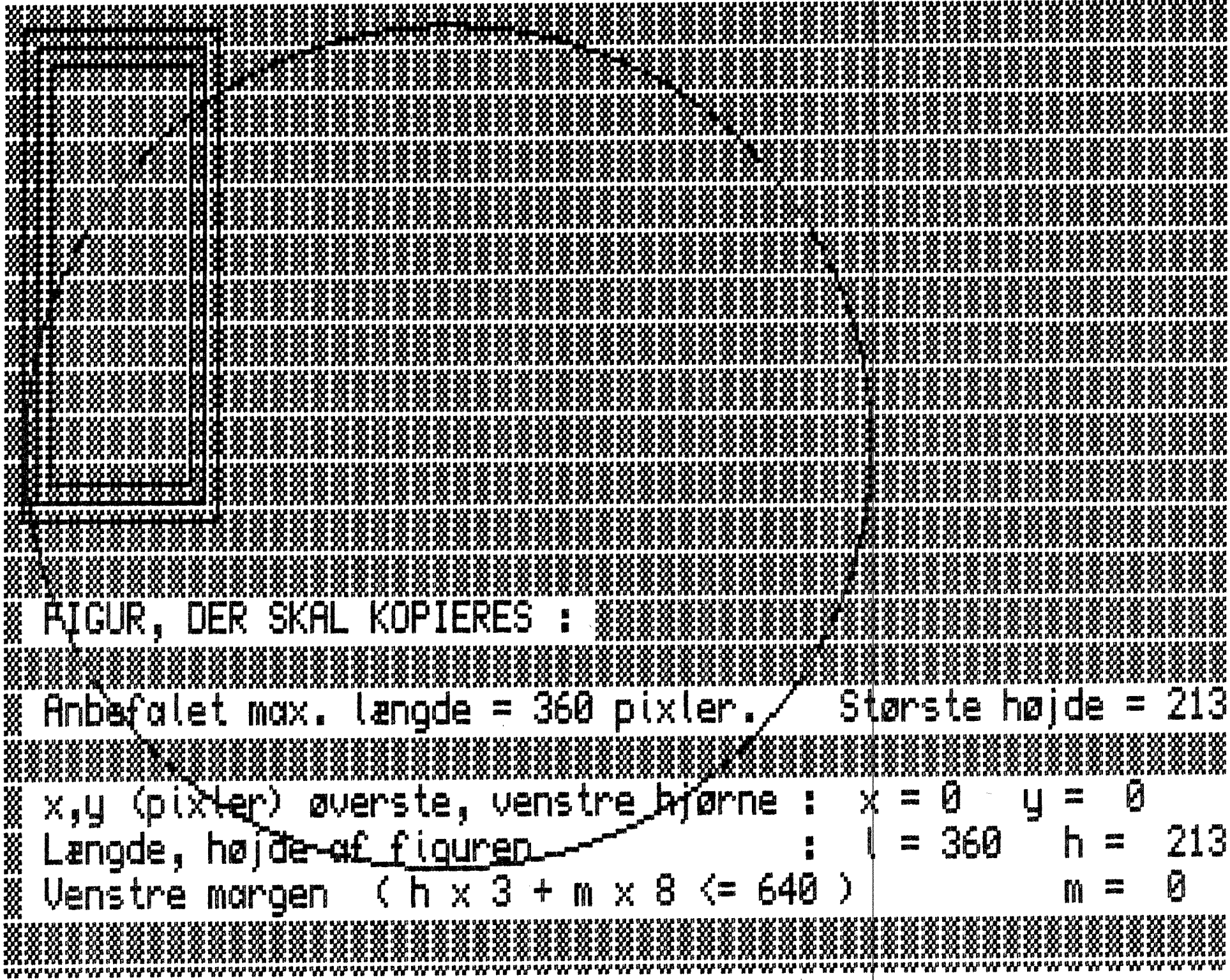
Sammenlign udskriften forrige side med ovenstående udskrift, der har BORDER 1,2.

```
10000 REMark kalkkopi_bas til kalkkopi
10010 DIM k(4), v$(606), w$(606), a$(16) : n= 0

10030 OPEN#3,ser1
10040  PRINT#3; CHR$(27); CHR$(51); CHR$(24);
10050  liniestart
10060  PRINT#3; CHR$(27); CHR$(51); CHR$(36);
10070 CLOSE#3

10090 DEFine PROCedure liniestart
10100  LOCal linie
10110  FOR linie = 156800 TO 156898   STEP 2
10120   v$ = '' : w$ = ''
10130   kopilinie
10140   PRINT#3; CHR$(27); CHR$(75); CHR$(94); CHR$(2); : PRINT#3; v$
10150   PRINT#3; CHR$(27); CHR$(75); CHR$(94); CHR$(2); : PRINT#3; w$
10160  END FOR linie
10170 END DEFine liniestart

10190 DEFine PROCedure kopilinie
10200  LOCal byte, a%, b%, byte1%, byte2%
10210  FOR byte = linie TO linie - 25728   STEP - 128
10220   a% = 128 : b% = 2 : a$ = FILL$('0',16)
10230   REPeat omdan
10240    IF a% && PEEK(byte) OR a% && PEEK(byte+1): a$(b%-1 TO b%)= '11'
10250    a% = a% / 2 : b% = b% + 2   : IF b% > 16 : EXIT omdan
10260   END REPeat omdan
10270   byte1% = BIN(( a$(1 TO 8))) : byte2% = BIN(( a$(9 TO 16)))
10280   v$ = v$ & CHR$(byte1%) & CHR$(byte1%) & CHR$(byte1%)
10290   w$ = w$ & CHR$(byte2%) & CHR$(byte2%) & CHR$(byte2%)
10300  END FOR byte
10310 END DEFine kopilinie
```

linie 10110 156800 = 131072 + 201 · 128
             156898 = 156800 + 400/4 - 2

linie 10210   25728 = 201 · 128

I øvrigt som "cp80skærm", idet start og slut kan udgå, da både x og ℓ er multipla af 8 (korr1% og korr2% bliver 0).


FIGUR, DER SKAL KOPIERES :
Anbefalet max. længde = 360 pixler.
Største højde = 213
x,y (pixler) øverste, venstre hjørne : x = 0 y = 0
Længde, højde af figuren : l = 360 h = 213
Venstre margen (h x 3 + m x 8 <= 640) m = 0

EKSEMPEL "flerkopi"	STOR KOPI, DREJET 90°
Gentagne kopier af forskellige skærme	

Principielt som "cp80skærm", dog SAVE's skærmbilledet med G_SAVE, hvorefter pixelinformationerne hentes fra G_SAVE's adresse. Se G_SAVE mv. pag.53.

Hvis det kompilerede program benyttes med kommandoen EW, går BASIC-programmet i stå, mens printeren kører, og så er der intet vundet.

Benyttes kommandoen EX flp1_scrkopi, starter printeren, mens BASIC-programmet fortsættes. De to programmer kører samtidigt (multi-tasking), med samme prioritet, og det betyder, at printetiden bliver ret lang.

Jeg benytter derfor ET (flp1_)scrkopi: SPJOB scrkopi,127 til at give printer-jobbet topprioritet, samtidig med at BASIC-programmet kører lidt langsommere - det vil af og til føles lidt "sløvt" (linie 32200).

Printetiden bliver 10-15 sekunder længere end "cp80skærm", ca. 5½ minut.

Der må indlægges en spærring, så brugeren ved kommandoer til printeren - herunder især fornyet brug af "scrkopi" - tvinges ind i en kø til printeren. BASIC-programmet stopper da, og frigives først, når printeren er klar til det næste job (linie 32140, 31260).

```
100 REMark flerkopiboot
110 INPUT \' RAM LOADed  J / N  '; y$ : IF y$== 'j' : GO TO 190
120 IF NOT y$== 'n' : GO TO 110

140 TK2_EXT : DATA_USE flp1_ : PROG_USE flp1_ : BAUD 4800 : TRA 1
150 LRESPR ram_ver1 : INK 2: FORMAT ram3_4: INK 7
160 LRESPR savescr
170 klar = RESPR(2) : POKE klar,1

190 REPeat loop

210 WINDOW    512,256,  0,  0: PAPER   0 : INK   7 : CLS
220 WINDOW#2; 436,202,  0,  0: PAPER#2;0 : INK#2;4
230 WINDOW#0; 436, 54,  0,202: PAPER#0;0 : INK#0;7
240 FOR n= 0 TO 24: INK 2 + n * ( n < 6 ): PRINT FILL$(CHR$(27),83)
250 FOR n= 0 TO 8 STEP 4: BLOCK 42+n*2,     1,14-n, 14-n,7
260 FOR n= 0 TO 8 STEP 4: BLOCK 42+n*2,     1,14-n,105+n,7
270 FOR n= 0 TO 8 STEP 4: BLOCK 1      ,92+n*2,14-n, 14-n,7
280 FOR n= 0 TO 8 STEP 4: BLOCK 1      ,92+n*2,55+n, 14-n,7 : INK 7
290 AT 13, 1 : PRINT ' FIGUR, DER SKAL KOPIERES : '
300 AT 10,45 : PRINT ' DEMO af kopi med G_SAVE '
310 AT 15, 1 : PRINT ' Anbefalet max. længde = 360 pixler.   ';
320 PRINT ' Største højde = 213 pixler. '
330 AT 17, 1 : PRINT ' x,y (pixler) øverste, venstre hjørne :';
340 PRINT FILL$(' ',29): AT 17,42 : INPUT 'x = '; x ; '   y =  '; y
350 AT 18, 1 : PRINT ' Længde, højde af figuren             :';
360 PRINT FILL$(' ',29): AT 18,42 : INPUT 'l = '; l ; '   h =  '; h
370 AT 19, 1 : PRINT ' Venstre margen  ( h x 3 + m x 8 <= 640 )';
380 PRINT FILL$(' ',27): AT 19,50 : INPUT '  m =  '; m
390 CIRCLE 38,62,36

410 x$ = x & ', ' & y & ', ' & l & ', ' & h : lf = 10
420 REMark lf = 12 giver FormFeed efter kopi , = 10 giver LineFeed.
430 cp80kopi x, y, l, h, m, x$, lf
440 AT 23,0: PRINT 'Nu kunne et BASIC-program fortsætte. ';
450 PRINT 'Tast, når du vil lave en ny kopi.' \ 'BASIC: DATE$= '
460 REPeat lp: AT 24,14: PRINT DATE$: y$= INKEY$: IF y$ <> '': EXIT lp

500 END REPeat loop
```

"flerkopiboot" omfatter endvidere
PROCeduren "cp80kopi", se pag. 34.

linie 100 "flerkopiboot" (DEMO-udgave til CPA-80S printeren). Skal kopieringsrutinen benyttes i et BASIC-program må

-linie 140-170 anbringes i programmets boot.
-linie 410, 430 anbringes de relevante steder.
-linie 32000-32230 PROCeduren "cp80kopi"kopi" MERGE's ind i programmet.

linie 110 Anbragt for at hindre gentagen brug af linie 140-170.

linie 140 Som sædvanlig bruges TK2-EXT og DEFAULT-værdierne sættes/ændres.

linie 150 RAM formatteres, udskriften er usynlig (INK 2).

linie 160 "savescr" giver G_SAVE (og G-LOAD) kommandoerne. (QL-WORLD, september 1987), se pag 53.

linie 170 Der er klar til brug af kopi-rutinen. Adressen klar indeholder 1.

linie 240 Skærmen fyldes med CHR$(27) tegn for kontrol af udskrift samt en cirkel og nogle linier.

linie 310 Alle længder er OK, men fra 360 til 400 pixler udfyldes A4 formatet lovlig meget. 512 er mulig, men skriver over perforeringen.

linie 410 Her sætter jeg ℓf=10 i DEMO-udgaven.
I praksis kan vælges 10 - med en efterfølgende figurtekst - eller 12 med sideskift.

x$ (= parameteren a$) i PROCeduren "cp80kopi" er brugerbestemt. I DEMO-udgaven har jeg valgt at angive x,y,ℓ,h. Med ℓf=10 kan der skrives en længere figurtekst over flere linier, hvis der er plads.

linie 440 Her følger i DEMO-udgaven en kort demonstration af, at BASIC fortsætter - uret kører, til der tastes et eller andet. Forlanges straks en ny kopi, går programmet i stå, til printeren er færdig med den første kopi - men fortsætter samtidig med, at den næste kopi's printning starter.

```
32000 DEFine PROCedure cp80kopi ( x, y, l, h, m, a$, lf )
32010   LOCal addr, dx, dl, læn
32020   IF x < 0 OR y < 0 OR x + l > 512 OR y + h > 256 THEN
32030     CLS#0 : PRINT#0; \' OUT OF RANGE' : STOP
32040   END IF
32050   IF h * 3 + m * 8 > 640  : CLS#0: PRINT#0;\' OUT OF RANGE' : STOP
32060   IF lf <> 10 AND lf <> 12: CLS#0: PRINT#0;\' OUT OF RANGE' : STOP
32070   REMark linie 32020-32070 er normalt overflødige
32080   dx= x MOD 8 : dl= l MOD 8     : læn= l
32090   SELect ON dx
32100     = 1       : IF NOT dl       : læn= l + 8
32110     = 2 TO 7  : IF dx + dl > 8 : læn= l + 8
32120   END SELect  : REMark G_SAVE er ejendommelig i sin afrunding af l
32130   REPeat ll   : IF PEEK(klar)  : EXIT ll
32140   POKE klar,0 : REMark scrkopi sætter klar= 1, når den er færdig.
32150   OPEN#3,ser: BPUT#3; 27,108,m,27,81,80,27,67,72,27,78,0 : CLOSE#3
32160   addr= ALCHP(27300)     : G_SAVE x, y, læn, h, addr
32170   OPEN_OVER#10;ram3_xylh: PRINT#10; addr
32180   PRINT#10; x : PRINT#10; y   : PRINT#10; l  : PRINT#10; h
32190   PRINT#10; a$: PRINT#10; klar: PRINT#10; lf : CLOSE#10
32200   TRA 0: ET flp1_scrkopi: SPJOB 'scrkopi',127
32210   PAUSE 50: TRA 1: RECHP addr
32220 END DEFine cp80kopi
```

linie 32000 PROCdure "cp80kopi" er en del af "flerkopiboot". Proceduren skal indgå i et program, der vil udnytte kopi-rutinen.

linie 32020 6 linier med kontrol er overflødige, hvis kopi-rutinens parametre er fastlagt af programmøren -men nødvendige, hvis programmets bruger kan vælge.

Længden, bredden og margin skal vælges med omhu for at placere tekst og tegning ordentligt.

linie 32080 4 linier, der forklares i G_SAVE mm, pag. 53.

linie 32130 Kun hvis adressen klar er >0(1), kan der fortsættes.

linie 32140 Nu starter kopi-rutinen og klar sættes til 0. klar bliver sat til 1 i det kompilerede program "scrkopi", linie 31260.

linie 32160 Adresse til skærmbilledet (512·213/4 = 27260). Må sættes op ved tilsvarende anvendelse med en "ql2skærm" rutine.

linie 32170 Information til "scrkopi" placeres i RAM, idet eventuel tidligere information slettes: OPEN_OVER # 10; ram3_xylh.

linie 32200 TRA 0 for at "oversætteren" for æ,ø,å ikke skal forvanske grafiktegn.

linie 32210 PAUSE 50 (1 sekund), så "scrkopi" kan nå at læse sin information i RAM, mens addr eksisterer. Den fjernes efter TRA 1.

<u>BASIC-PROGRAM, næste side.</u>
"scrkopi-bas", kompileret som "scrkopi", se næste side. Da det oprindelige x,y nu er G_SAVE's "nulpunkt", ændres linie 31000-31110 noget i forhold til "cp80skærm's" tilsvarende linie 10000, pag. 19.

linie 31050 Information fra RAM om file$=a$ og lf.

linie 31090 Variablen trin indføres (antal bytes pr. linie i det ønskede skærmudsnit). Start og slut bliver korrekt, uanset om korr2% = 0.

linie 31200 Resten er magen til "cp80Skærm", pag.20.

Forenklingsmulighed:
linie 31040 k(2) bruges ikke. k(1) kan undværes, hvis k(1) er et multiplum af 8. k(3) og k(4) findes i G_SAVE på start + 2 og start + 6 (start, se linie 31040).

Ofte vil alle RAM-operationer kunne undværes - næsten, se pag. 51, linie 1040, hvor en fast adresse kunne overvejes, fx. via #0's information, som vist i IFH-rapport 182, hovedprogrammets start.

```
31000 REMark scrkopi_bas
31010 DIM k(4), v$(640), w$(640), a$(16), file$(25), y$(17)
31020 korr1%= 0: korr2%= 0: vent%= 0
31030 n= 0: tid= 0: start= 0: slut= 0: hob= 0: lob= 0: trin= 0
31040 OPEN#10; ram3_xylh: INPUT#10; start: FOR n= 1 TO 4: INPUT#10; k(n)
31050 INPUT#10; file$ : INPUT#10; klar : INPUT#10; lf: CLOSE#10
31060 hob = k(4) * 3  DIV 256 : lob = k(4) * 3  MOD 256
31070 korr1% =   k(1) MOD 8
31080 korr2% = ( k(1) + k(3) ) MOD 8
31090 trin   = ( k(3) + korr1% + ( 8 - korr2% ) * ( korr2% > 0 ) ) / 4
31100 start  = start + 8 + ( k(4) - 1 ) * trin
31110 slut   = start + trin - 2

31200 OPEN#3,ser1
31210  PRINT#3; CHR$(27); CHR$(51); CHR$(24);
31220  liniestart
31230  PRINT#3; CHR$(27); CHR$(51); CHR$(36);
31240  y$= DATE$: PRINT#3;\ file$;'    '; y$(1 TO 17): PRINT#3;CHR$(lf);
31250 CLOSE#3
31260 POKE klar,1

31400 DEFine PROCedure liniestart
31410  LOCal linie, ast%, bst%, bsl%
31420  linie = start : v$ = '' : w$ = ''
31430  ast% = 2^(7 - korr1%) : bst% = 2 + 2 * korr1% : bsl% = 16
31440  kopilinie
31450  IF korr1% + k(3) < 9 :                RETurn
31460  IF korr1% + k(3) < 9 + korr2% :       GO TO 31510
31470  FOR linie = start + 2 TO  slut - 2    STEP 2
31480   v$ = '' : w$ = '' : ast% = 128 : bst% = 2 : bsl% = 16
31490   kopilinie
31500  END FOR linie
31510  linie = slut : v$ = '' : w$ = ''
31520  ast% = 128 : bst% = 2 : bsl% = 16 : IF korr2% : bsl% = 2 * korr2%
31530  kopilinie
31540 END DEFine liniestart

31600 DEFine PROCedure kopilinie
31610  LOCal byte, a%, b%, byte1%, byte2%
31620  FOR byte = linie TO linie - ( k(4) - 1 ) * trin   STEP -trin
31630   a% = ast% : b% = bst% : a$ = FILL$('0',16)
31640   REPeat omdan
31650    IF a% && PEEK(byte) OR a% && PEEK(byte+1): a$(b%-1 TO b%)= '11'
31660    a% = a% / 2 : b% = b% + 2    : IF b% > bsl% : EXIT omdan
31670   END REPeat omdan
31680   byte1% = BIN(( a$(1 TO 8))) : byte2% = BIN(( a$(9 TO 16)))
31690   v$ = v$ & CHR$(byte1%) & CHR$(byte1%) & CHR$(byte1%)
31700   w$ = w$ & CHR$(byte2%) & CHR$(byte2%) & CHR$(byte2%)
31710  END FOR byte
31720  PRINT#3; CHR$(27); CHR$(75); CHR$(lob); CHR$(hob); : PRINT#3; v$
31730  PRINT#3; CHR$(27); CHR$(75); CHR$(lob); CHR$(hob); : PRINT#3; w$
31740 END DEFine kopilinie
```

BASIC-udgave, kompileres som "scr-kopi".
Se noter på forrige side.

EKSEMPEL "experiment".	SMÅ, RETVENDTE KOPIER

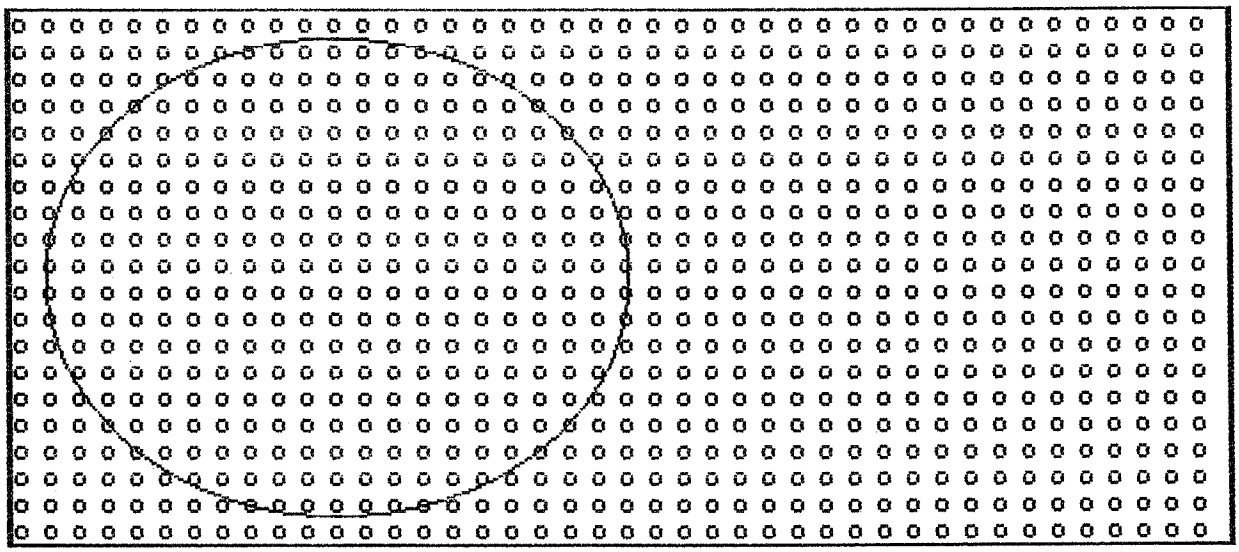
Nedenfor er vist de maskinkodede programmer, IFH benytter.

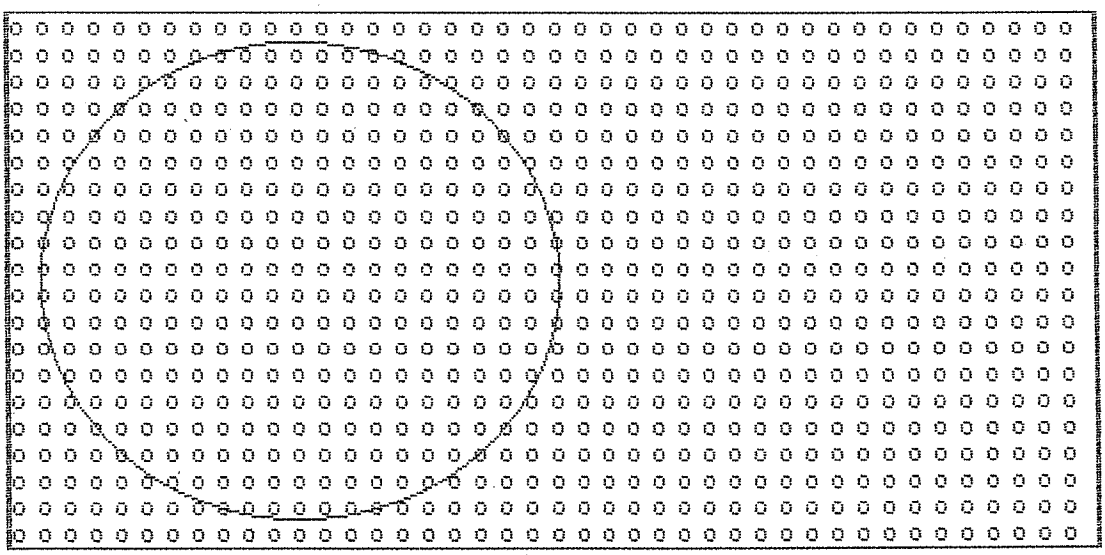
På de følgende sider er vist et BASIC-program, der i kompileret udgave giver en lille skaleringsfejl (1%), men er langsommere.

Nedenfor vises:

"scopy". 65 sec. 21% fejl TIL CPA80S - printeren. 640 dots/line

"qcopy". 112 sec. 8% fejl TIL QL - printeren. 720 dots/line





CPA80S-printeren har faciliteter for DOWNLOAD af bruger-definerede karakterer.

"experiment" udnytter denne mulighed, og bruger "ELITE", 96 karakterer pr. linie, dvs 96x8 dots per line.

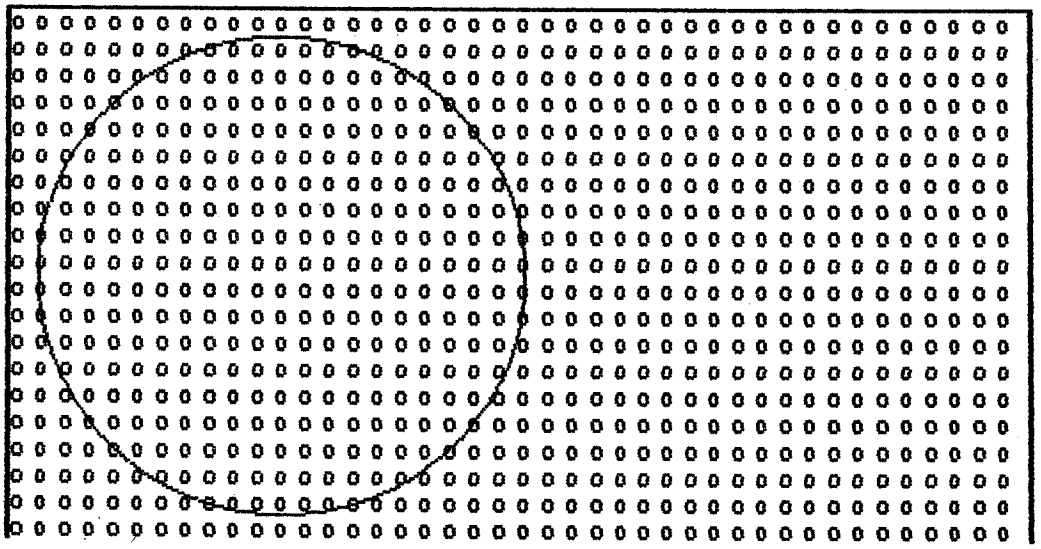
Som nævnt tidligere, under "KOPI-MULIGHEDER" kan der så opnås en lille, retvendt kopi med kun 1% fejl.

"experiment" er kun udarbejdet for et skærmfelt, hvis længde og bredde begge er multipla af 8, her 512x200 (derfor mangler nederste BORDER, h=202).

Dette forenkler programmet, men alligevel er printetiden 139 sekunder, bl.a. fordi beregningerne er omfattende, men også fordi udprintning i ELITE er en langsom karakter-printemetode.

For CPA-80SA80S-printeren er printetiderne (samme felt):

Almindelige karakterer, PICA	:	26 sekunder
do do , ELITE	:	56 sekunder
Prædef. down-load karakterer, udskrevet i ELITE	:	56 SEKUNDER
Skærmaflæst down-load, udskrevet i ELITE	:	139 SEKUNDER
"scopy"	:	65 sekunder



```
100 REMark "experiment" = BASIC-DEMO of "scrcopy"
110 WINDOW 512,202,0,0 : WINDOW#2; 436,202,0,0
120 PAPER 0 : INK 2 : BORDER 1,7 : CLS
130 FOR n = 1 TO 20 : INK 2 +2*(n>5) +3*(n>12): PRINT FILL$('o ',84)
140 CIRCLE 50,50,45
150 BAUD 4800 : TRA 0 : OPEN#3,ser        : REMark TRA 0 = Internat.chars
160 PRINT#3; CHR$(27);CHR$(51);CHR$(24); : REMark Line spacing 24/216"
170 PRINT#3; CHR$(27);CHR$(77);           : REMark ELITE, 96 chars/line
180 PRINT#3; CHR$(27);CHR$(37);CHR$(1);   : REMark Use dwnld. char.set

200 FOR start = 131072  TO  155648      STEP 1024

220 FOR charact = start TO start + 126  STEP 2

240 DIM c%(7) : char = charact : val% = 128
250 REPeat lin
260  pos% = 0 : div% = 128
270  REPeat loop
280    IF ( div% && PEEK( char ) ) OR ( div% && PEEK( char + 1 ) ) THEN
290       c%(pos%) = c%(pos%) + val%
300    END IF
310    IF pos% > 6 : EXIT loop
320    pos% = pos% + 1 : div% = div% / 2
330  END REPeat loop
340  IF val% < 2   : EXIT lin
350  char = char + 128 : val% = val% / 2
360 END REPeat lin
370                                       : REMark Define dwnld.char.200
380 PRINT#3; CHR$(27);CHR$(38);CHR$(200);CHR$(200);CHR$(0);
390 FOR ch = 0 TO 7 : PRINT#3; CHR$(c%(ch));
400 PRINT#3; CHR$(200);                   : REMark Print  dwnld.char.200

420 END FOR charact
430 PRINT#3                               : REMark New line

450 END FOR start

470 PRINT#3; CHR$(27);CHR$(82);CHR$(14); : REMark Use Danish char.set
480 PRINT#3; CHR$(27);CHR$(80);           : REMark PICA, 80 chars/line
490 PRINT#3; CHR$(27);CHR$(51);CHR$(36); : REMark Line spacing 36/216"
500 PRINT#3; CHR$(12);                    : REMark No FF when compiled
510 CLOSE#3 : TRA 1                       : REMark Danish characters
```

KOMMENTAR:
Se de følgende sider, hvor dette program deles i en generel BASIC-del og et BASIC-program til kompilering.

```
100 REMark "experiment2". DEMO2 TURBO-compiled "scrcopy" ("scrcopy_bas")
110 WINDOW 512,202,0,0 : WINDOW#2; 436,202,0,0
120 PAPER 0 : INK 2 : BORDER 1,7 : CLS
130 FOR n = 1 TO 20 : INK 2 +2*(n>5) +3*(n>12): PRINT FILL$('o ',84)
140 CIRCLE 50,50,45
150 TK2_EXT : BAUD 4800
160 time = DATE
170 OPEN#3,ser: BPUT#3;27,108,5,27,81,77,27,67,72,27,78,10: CLOSE#3
180 EW scrcopy
190 OPEN#3,ser: PRINT#3; \ DATE - time ; ' sec': BPUT#3;12: CLOSE#3
```

Ovenfor vises startprogrammet. "scrcopy" er en kompileret udgave af scrcopy_bas, næste side.

"scrcopy" aflæser 8 x 8 pixler som omdannes til et bruger-defineret bogstav, der downloades og derefter printes. En omstændelig proces, der givetvis kunne gøres hurtigere i maskinkode/med bedre kendskab til QL'ens styresystem m.v.

Skal der udvides til vilkårlige x,y,l,h, kræves fire IF'er med specielle printrutiner langs feltets fire kanter.

Bogstaverne udprintes i ELITE (96 karakterer/linie) med en fejl på 1%. Fjernes linie 130, printes i PICA (80 karakterer/linie) med en fejl på 21% som SCOPY. PICA er hurtigere end ELITE.

Der læses 8 x 8 pixler, lodret nedad, fra øverste, venstre hjørne. Skærmen aflæses, 8 vandrette pixler ad gangen, som et tal mellem 0 og 255. Disse 8 linier skal omdannes til 8 søjler, igen repræsenteret ved et tal mellem 0 og 255.

Eksempel:

Skærm aflæst

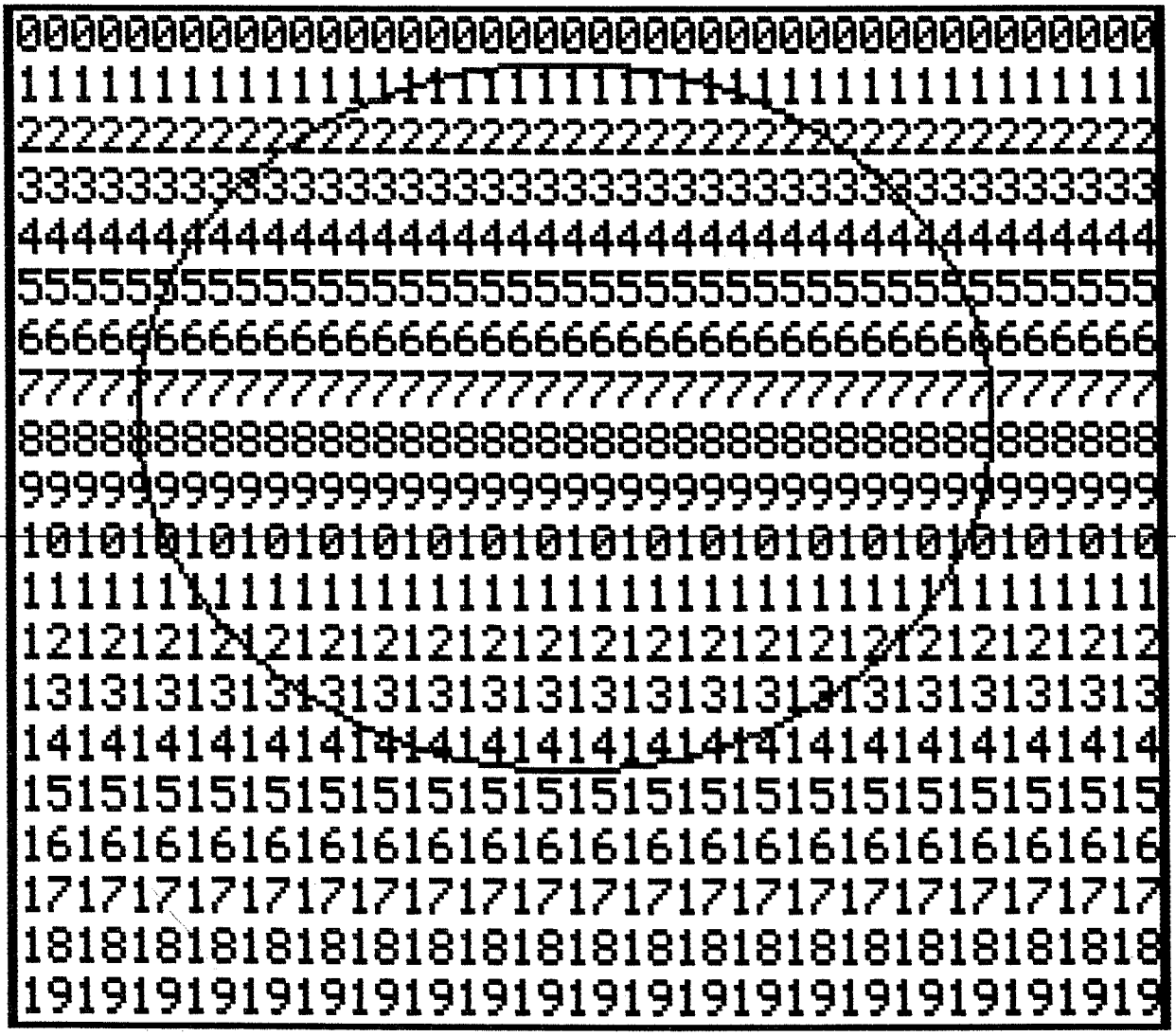
Værdi 128	64	32	16	8	4	2	1	SUM
0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	127
0	1	1	0	1	0	1	0	106
0	1	1	1	0	1	0	1	117
0	1	1	0	1	0	1	0	106
0	1	1	1	0	1	0	1	117
0	1	1	0	1	0	1	0	106
0	1	1	1	0	1	0	1	117

Tilsvarende download bogstav

								Værdi
0	0	0	0	0	0	0	0	128
0	1	1	1	1	1	1	1	64
0	1	1	0	1	0	1	0	32
0	1	1	1	0	1	0	1	16
0	1	1	0	1	0	1	0	8
0	1	1	1	0	1	0	1	4
0	1	1	0	1	0	1	0	2
0	1	1	1	0	1	0	1	1
0	127	127	85	105	85	105	85	SUM

```
100 REMark scrcopy_bas
110 TRA 0 : OPEN#3,ser
120 PRINT#3; CHR$(27);CHR$(51);CHR$(24);
130 PRINT#3; CHR$(27);CHR$(77);
140 PRINT#3; CHR$(27);CHR$(37);CHR$(1);
150 DIM v$(96)
160 FOR start= 131072 TO 155648  STEP 1024
170  v$='': nr= 0
180  REPeat loop
190   DIM c%(7): char= start + nr * 2: val%= 128
200   REPeat lin
210    pos%= 0: div%= 128
220    REPeat loop2
230     IF (div% && PEEK(char)) OR (div% && PEEK(char+1)) THEN
240      c%(pos%)= c%(pos%) + val%
250     END IF
260     IF pos% > 6: EXIT loop2
270      pos%= pos% + 1: div%= div% / 2
280    END REPeat loop2
290    IF val% < 2: EXIT lin
300    char= char + 128: val%= val% / 2
310   END REPeat lin
320   PRINT#3; CHR$(27);CHR$(38);CHR$(nr + 34);CHR$(nr + 34);CHR$(0);
330   FOR ch= 0 TO 7: PRINT#3; CHR$(c%(ch));
340   v$= v$ & CHR$(nr + 34): nr= nr + 1 : IF nr > 63: EXIT loop
350  END REPeat loop
360  PRINT#3;v$
370 END FOR start
380 PRINT#3; CHR$(27);CHR$(82);CHR$(14);
390 PRINT#3; CHR$(27);CHR$(80);
400 PRINT#3; CHR$(27);CHR$(51);CHR$(36);
410 CLOSE#3 : TRA 1
```

linie 110 TRA 0. For ikke at få fejloversættelser af grafiktegn
linie 120 Linieafstand 24/216 = 1/9"
linie 130 ELITE, dvs. 96 karakterer pr. linie
linie 140 Brug download-karaktersæt
linie 150 Øverste, venstre hjørne har adresse 131072
           8 linier ned har startadresse 131072+8·128=131072+1024
           192 linier ned har startadresse 155648
linie 180 Læs en linie
linie 190 Øverste 8 pixler har værdien 128 i download-søjlerne
linie 210 Hvis hvid/grøn eller hvis rød, skal der printes
linie 250 Positionernes værdi halveres
linie 280 char + 128 er næste linie, liniernes "værdi" halveres
linie 300 Definer download-karakter 200
linie 320 Print download-karakter 200
linie 340 Ny linie
linie 360 Brug dansk karaktersæt igen
linie 370 Retur til PICA (80 karakterer/linie)
linie 380 Retur til 36/216 = 1/6" linieafstand
linie 390 Retur til TRA 1, danske karakterer æ,ø,å.



EKSEMPEL "retkopi"	STOR, RETVENDT KOPI

Maximal bredde 320 pixler.

Forudsætter, at x (startkoordinaten) er et multiplum af 8, mens y,l,h kan vælges frit.

"Fordobling" lodret og vandret af pixler til 2 x 2 dots.

Afbildningsfejl 22% som "scopy".
Printetid knap 3 minutter for 256 x 202 pixler (CPA80S).

Boot indlæser "cp80retkopi" til CPA80S-printeren.
Et tilsvarende "qlretkopi"-program er ikke udarbejdet.

Se også det følgende eksempel "cperkopi" og "qperkopi".

SKÆRM - princip. Læsning af 8 x 8 pixler. Se pag. 47.

pos%	1	2	3	4	5	6	7	8
peak	128	64	32	16	8	4	2	1
peak + 128	128							1
	128							1
	128			a%				1
	128							1
	128							1
peak + 768	128							1
peak + 896 (peak+1024)	128	64	32	16	8	4	2	1

PRINTER - princip. Grafiske karakterer.

To dots	val%	korr%	sℓ14%	sℓ58%	
1	128	1	128	256	
2	32	2	32	256	byte1% (pos%)
3	8	3	8	256	val% * 1.5
4	2	4	2	256	
5	128	5	2	128	
6	32	6	2	32	byte2% (pos%)
7	8	7	2	4	val% * 1.5
8	2	0	2	2	

```
100 REMark retkopiboot
110 WMON 4: FOR n=1 TO 2: PAPER#n;0: INK#n; 7: BORDER#n; 1,7: CLS#n
120 INPUT \ ' RAM LOADed  J / N   '; y$
130 IF y$ == 'j' : GO TO 170
140 TK2_EXT : BAUD 4800 : TRA 1
150 LRESPR ram_ver1 : FORMAT ram3_2

170 CLS : PRINT \ ' Indtast x, y, l, h ( x mod 8 = 0 )'
180 INPUT ' x= '; x ; '        y= '; y
190 INPUT ' l= '; l ; '        h= '; h
200 IF x MOD 8 OR x + l > 512 OR l > 320 : GO TO 170
210 IF y + h > 256 : GO TO 170
220 INPUT \ ' Venstre margen= '; m
230 IF 2 * l + 8 * m > 640 : GO TO 220
240 OPEN#3,ser: BPUT#3;27,108,m,27,81,80,27,67,72,27,78,10: CLOSE#3
250 CLS : FOR n= 0 TO 19: PRINT FILL$(n+1,42): PRINT#2;FILL$(n,42)
260 FOR n=1 TO 2: CIRCLE#n; 45,60,35
```

```
280 skærmkopi x, y, l, h

500 DEFine PROCedure skærmkopi ( x, y, l, h )
510   DELETE ram3_xylh
520   OPEN_NEW#10; ram3_xylh
530   PRINT#10; x: PRINT#10; y: PRINT#10; l: PRINT#10; h: CLOSE#10
540   TRA 0 : EW cpretkopi : TRA 1
550 END DEFine skærmkopi



1000 REMark cpretkopi_bas til kompilering som cpretkopi
1010 DIM k(4), v$(640), w$(640), byte1%(8), byte2%(8)
1020 b= 0: begynd= 0: slut= 0: val%= 0: a%= 0: pos%=0
1030 korr%= 0: l14%= 0: l58%= 0: ned= 0: hen= 0: hob= 0: lob= 0
1040 OPEN#10; ram3_xylh : FOR b= 1 TO 4: INPUT#10; k(b)
1050 CLOSE#10
1060 begynd= 131072 + k(1) DIV 4 + k(2) * 128
1070 korr% = k(4) MOD 8: ned = k(4) - korr% + 8 * ( korr% > 0 )
1080 slut  = begynd + ( ned - 8 ) * 128
1090 hen   = k(3): IF k(3) MOD 8: hen= k(3) + 8 - ( k(3) MOD 8 )
1100 hen   = hen DIV 4 - 2
1110 hob   = k(3)*2 DIV 256: lob = k(3)*2 MOD 256
1120 l14% = 512 / 2^(2 * korr%) : IF l14% < 2 : l14% = 2
1130 IF korr% < 5 : l58% = 256  : ELSE l58% = 512 / 2^(2 * (korr%-4))
1140 :
1150 OPEN#3,ser
1160 PRINT#3; CHR$(27);CHR$(51);CHR$(24);

1200 FOR start= begynd TO slut - 1024  STEP 1024
1210  beregn (2), (2)
1220  udskriv
1230 END FOR start

1250 start= slut
1260 IF korr%= 0 : beregn (2), (2): ELSE beregn (l14%), (l58%)
1270 udskriv

1290 PRINT#3;CHR$(27);CHR$(51);CHR$(36);
1300 PRINT#3;CHR$(12);: CLOSE#3
```

BOOT:

"retkopiboot" ligner de øvrige boot, bortset fra kravet om, at x skal være et multiplum af 8 (linie 170,200).

HOVEDPROGRAM:

linie 1000 BASIC-udgave af "cpretkopi".

linie 1010 640 er det maksimale antal dots per line på CPA80S-printeren med normal udskrift.

linie 1040 x,y,ℓ,h hentes fra RAM som k(1)....k(4).

linie 1060 Startpositionen udregnes:
131072 = øverste, venstre hjørne
+2 pr. 8 pixler i x-retningen (vandret)
+128 pr. pixel-linie (y,lodret)

linie 1070 Der læses fra venstre mod højre på skærmen, 8 x 8 pixler ad gangen.

korr% angiver, hvor meget der ikke skal bruges i sidste læsning: ned skal korrigeres, hvis korr% > 0.

linie 1080 slut beregnes (pixellinie 0 til (ned-8)).

linie 1090 hen beregnes, idet der korrigeres, hvis hen ikke er et multiplum af 8. Omsættes til byte-adresser i linie 1100.

linie 1120 De følgende linier ℓ14%, ℓ58%, se næste side.

linie 1160 Som sædvanlig: 1/9" linieafstand til grafik(24/216").

linie 1200 Først alle de sæt à pixler, hvor alt skal med. (8 * 128 = 1024).

linie 1250 De sidste sæt à 8 pixler, har måske for meget med.

linie 1290 Tilbage til normal linieafstand, 1/6".

```
1400 DEFine PROCedure beregn ( sl14%, sl58% )
1410  LOCal n : v$ = '': w$ = ''
1420  FOR blok= start TO start + hen STEP 2
1430   DIM byte1%(8), byte2%(8): peak= blok: val%= 128
1440   REPeat linie14
1450    a%= 128: pos%= 1
1460    REPeat omdan
1470     IF a% && PEEK(peak) OR a% && PEEK(peak+1) THEN
1480        byte1%(pos%)= byte1%(pos%) + val% * 1.5
1490     END IF
1500     pos%= pos% + 1: IF pos% > 8: EXIT omdan
1510     a%  = a% / 2
1520    END REPeat omdan
1530    val%= val% / 4: peak= peak + 128: IF val% < sl14%: EXIT linie14
1540   END REPeat linie14
1550   val%= 128
1560   REPeat linie58
1570    IF sl58% > 128 : EXIT linie58
1580    a%= 128: pos%= 1
1590    REPeat omdan
1600     IF a% && PEEK(peak) OR a% && PEEK(peak+1) THEN
1610        byte2%(pos%)= byte2%(pos%) + val% * 1.5
1620     END IF
1630     pos%= pos% + 1: IF pos% > 8: EXIT omdan
1640     a%  = a% / 2
1650    END REPeat omdan
1660    val%= val% / 4: peak= peak + 128: IF val% < sl58%: EXIT linie58
1670   END REPeat linie58
1680   FOR n = 1 TO 8: v$ = v$ & CHR$(byte1%(n)) & CHR$(byte1%(n))
1690   FOR n = 1 TO 8: w$ = w$ & CHR$(byte2%(n)) & CHR$(byte2%(n))
1700  END FOR blok
1710 END DEFine beregn
```

```
1800 DEFine PROCedure udskriv
1810  v$ = v$( 1 TO k(3)*2 ) : w$ = w$( 1 TO k(3)*2 )
1820  PRINT#3; CHR$(27);CHR$(75);CHR$(lob);CHR$(hob);: PRINT#3; v$
1830  PRINT#3; CHR$(27);CHR$(75);CHR$(lob);CHR$(hob);: PRINT#3; w$
1840 END DEFine udskriv
```

linie 1400 PROCedure beregn, se diagrammer pag. 43.

linie 1420 Der læses 8 pixler ad gangen, 2 bytes, i blokke à 8 linier.

linie 1430 byte1%(8) skal indeholde information om de fire
linie 1440 øverste liniers 4 x 8 pixler omsat til 8 grafiske tegn à 8 dots (lodret). Fordobling, lodret (idet vandret fordobling sker i linie 1680).

byte2% indeholder tilsvarende de fire nederste linier.

linie 1450 Hver linie har i sine 2 bytes 2·8 bits med information om farven, se pag. 4, med værdierne a%=128,64,32 16,8,4,2,1 (vandret, linie for linie).

Disse 8 linier à 8 pixler, skal omdannes til:

linie 1480 8 søjler à (4x2) dots (byte1%(1....8)) plus

linie 1610 8 søjler à (4x2) dots (byte2%(1....8)).

I søjlerne er værdierne ligeledes 128,64....4,2,1 men lodret. Der skal ske fordobling lodret af en hvid/grøn/rød pixel, der læses ved:
linie 1470 a% && (peak) = hvid/grøn, a% && (peak+1) = rød
Det medfører, at to dots i søjlen skal sættes, fx. 128+64 / 32+16 / 8+4 / 2+1.

linie 1480 Dette svarer til val% * 1.5.

linie 1530 val%/4 i stedet for sædvanlig halvering.
peak forøges med 128 (1 skærmlinie = 64·2).

linie 1500 Der læses 8 positioner i de fire første linier.

linie 1530 sℓ14% standser læsningen, hvis der er overflødige linier.

linie 1560 De sidste 4 linier (5....8) læses tilsvarende,

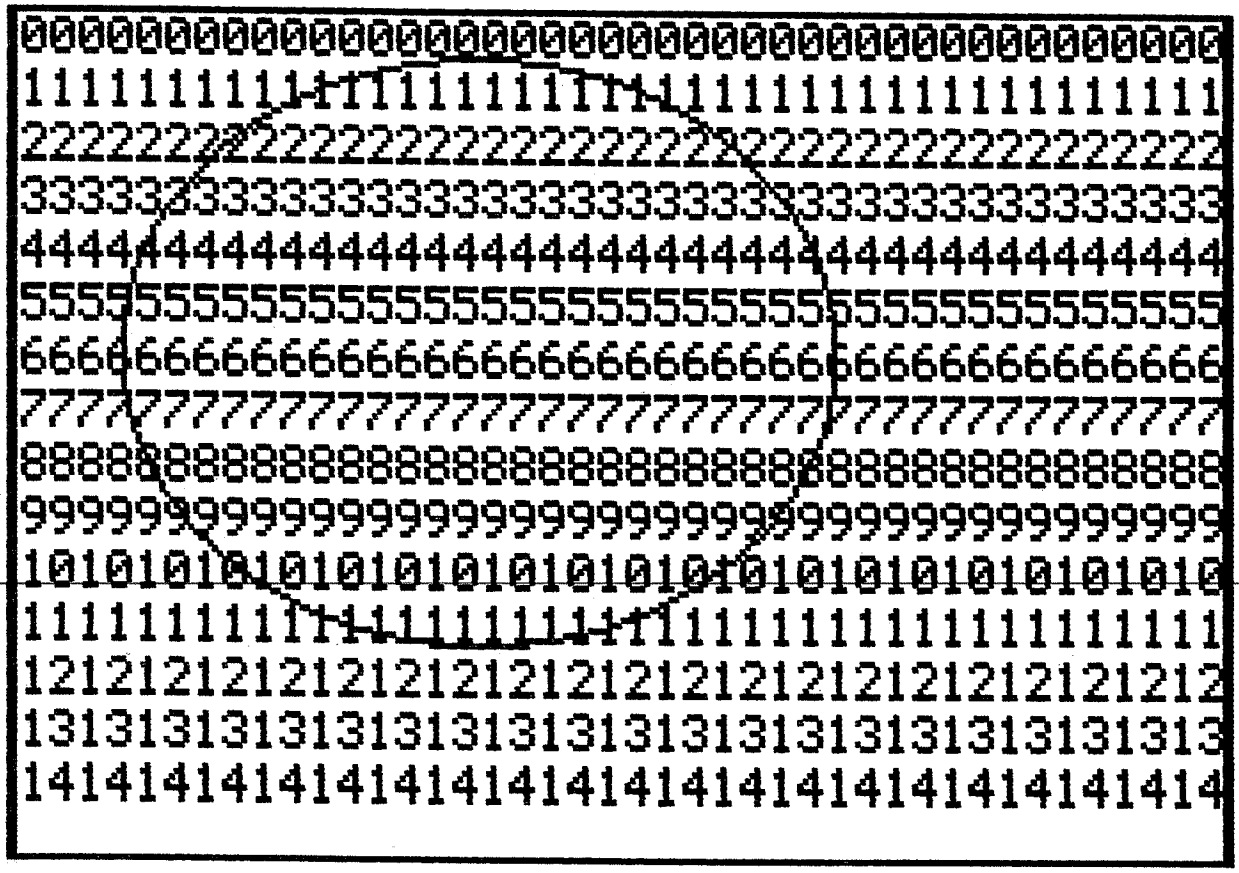
linie 1570 hvis de skal bruges,

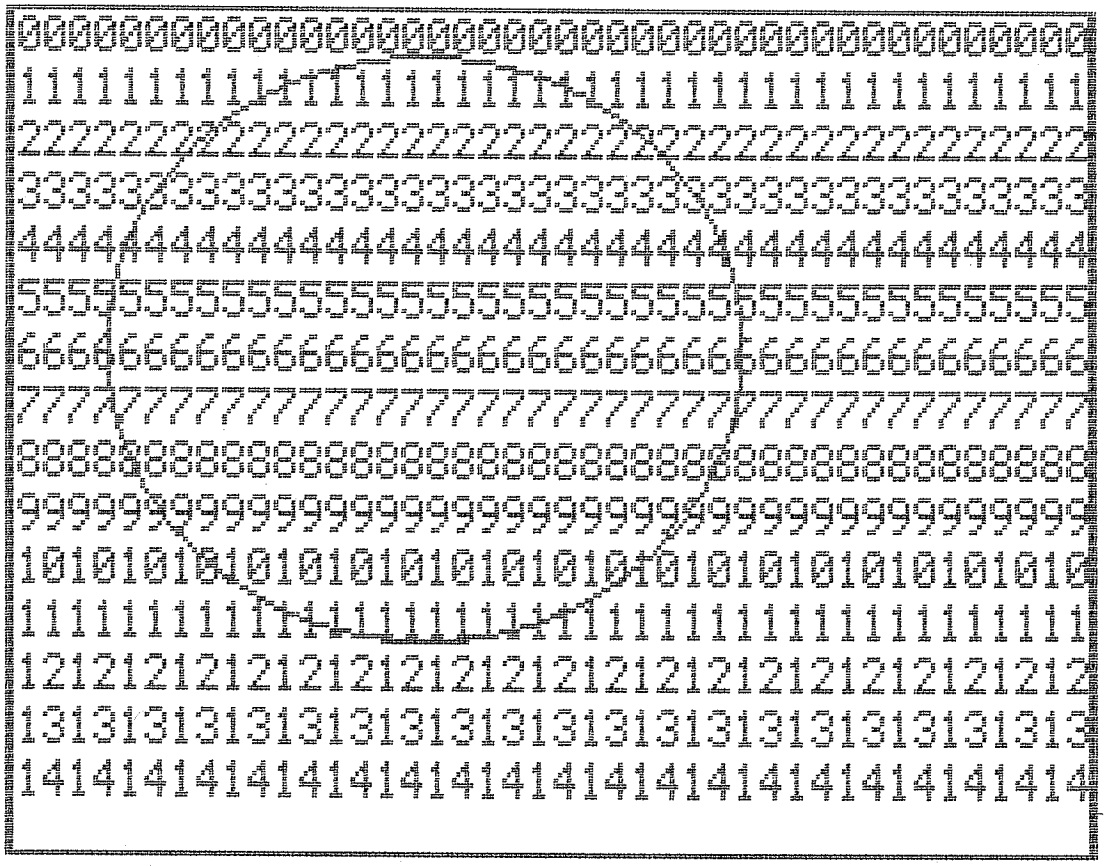
linie 1660 eller delvist bruges (sℓ58%).

linie 1680 v$ og w$ tilføjes hver 8 grafiske tegn.

linie 1810 Simpel strengslicing til at fjerne eventuelt overflødigt til højre.

linie 1820 CHR$(75), 640 dots per line.





EKSEMPEL "cperkopi" og "qperkopi"

Stor, retvendt kopi, specielt udarbejdet til stud. polyt. Per Grydgaard's eksamensprojekt der kopieres fra venstre eller højre skærm, begge 256 x 160 pixler med øverste, venstre hjørne i 0,42 hhv. 256,42.

Forenkling af "retkopi", da x,ℓ og h alle er multipla af 8.

"cperkopi" til CPA80S printeren.
640 dots per line. 22% afbildningsfejl (som "scopy").
Printetid 2 minutter.

"qperkopi" til QL-printeren.
720 dots per line. 8% afbildningsfejl (som "qcopy").
Printetid 4 minutter.

De to skærmudskrifter er prøver, der intet har med P.G.'s projekt at gøre.

Bemærk:
Sammenlign med "kalkkopi": "cperkopi" prøven har BORDER 1,7.

```
100 REMark boot // programdel ( "demo_cperkopi" )
110 WINDOW 512,256,0,0: PAPER 2: CLS
120 WINDOW 256,160,0,42 : WINDOW#2; 256,160,256,42
130 FOR n=1 TO 2: PAPER#n;0: INK#n; 7: BORDER#n; 1,7: CLS#n
140 INPUT \ ' RAM LOADed  J / N   '; y$
150 IF y$ == 'j' : GO TO  210

170 TK2_EXT : BAUD 4800 : TRA 1
180 LRESPR ram_ver1 : FORMAT ram3_2
190 REMark Anbringes i programmets boot

210 INPUT \ ' Højre eller venstre vindue  H / V  '; hv$
220 IF NOT ( hv$ INSTR 'hv' ) : GO TO 210
230 CLS : FOR n= 0 TO 14: PRINT FILL$(n+1,42): PRINT#2;FILL$(n,42)
240 FOR n=1 TO 2: INK#n; 2: CIRCLE#n; 45,60,35: INK#n; 7
250 :
260 OPEN#3,ser: BPUT#3; 27,108,5,27,81,77,27,67,72,27,78,10: CLOSE#3
270 REMark Venstre + højre margen = 5 + 3 tegn = 8 * 8 dots, dvs
280 REMark "normal" udskrift. 64 + 2 * 256 = 576 dots < 640 dots/line
290 skærmkopi hv$ : REMark ******************************
300 REMark Anbringes ad hoc i programmet


400 DEFine PROCedure skærmkopi ( hv$ )
410   begynd = 136448 : IF hv$ == 'h' : begynd = 136512
420   REMark Venstre 131072 + 42 * 128 = 136448 : Højre + 64 = 136512
430   DELETE ram3_vh : tid = DATE
440   OPEN_NEW#10; ram3_vh: PRINT#10; begynd: CLOSE#10
444   REMark Per Grydes skærme, til CPA80S-printeren
450 REMark TRA 0 : EW cperkopi : TRA 1 : REMark Altid TRA 0 før grafik !
460   TRA 0 : GO SUB 1000 : TRA 1 : REMark Benyttes til BASIC prøve
470   OPEN#3,ser: PRINT#3; \ DATE - tid : BPUT#3;12 : CLOSE#3
480 END DEFine skærmkopi

500 REMark til det kompilerede program, cperkopi :
510 REMark Linieafstanden reduceres til 24/216" = 1/9"     (linie 1020).
520 REMark Læsningen begynder for hver 8.linie : STEP 8 * 128 = 1024,
530 REMark Lodret ialt ( 160 - 8 ) * 128 = 19 * 1024 = 19456 (lin 1060).
540 REMark Læsningen i hver linie sker med STEP 2 pr. 8 pixler, dvs
550 REMark ialt fra start til start + ( 256 / 4 ) - 2  = 62  (lin 1080).
560 REMark Fordobling af pixler til dots på begge leder.
570 REMark Der læses 8 x 8 pixler ( PEEK, linie 1130, første fire linier
580 REMark med 8 pixler, og PEEK, linie 1250, sidste fire linier ).
590 REMark Disse omdannes til grafiktegn, to lodrette søjler med 8 dots,
600 REMark byte1%(pos%) og byte2%(pos%)                (linie 1140,1260).
610 REMark Dette gentages vandret, hvorefter de grafiske tegn printes.
620 REMark De første fire liner x 8 pixler på skærmen bliver til v$,
630 REMark med 8 x 2 ens søjler, med 4 x 2 ens bits   (linie 1100-1200).
640 REMark w$ printer de nederste fire skærmlinier    (linie 1220-1320).
650 REMark v$ og w$ får 16 søjler (tegn) pr.læsning   (linie 1330-1340).
660 REMark 256 pixler vandret bliver til 512 dots.
670 REMark 160 pixler ( linier) bliver til 20 x 2 søjler med 8 dots.
680 REMark Skærmen forudsættes sort. Hvid, grøn, rød printes som sort
690 REMark på hvidt papir. Derfor læses både den lige adresse og den
700 REMark efterfølgende ulige. Kun sort vil returnere 0 ved BEGGE PEEK,
710 REMark der analyseres bit for bit med &&          (linie 1130,1250).
720 REMark a% er 128,64,32,..,2,1 ( skærmens 8 vandrette pixler ).
730 REMark val% er 128,64,32,..,2,1 ( grafiktegnenes lodrette søjler,
740 REMark v$ og w$ udprinter to linier, ialt (8+8) x 512 dots, der
750 REMark fordobler 8 x 256 skærmpixler              (linie 1360,1370).
760 REMark Normal linieafstand 36/216" = 1/6", TRA 1  (linie 1390, 450).
```

```
1000 REMark cperkopi_bas til kompilering som cperkopi
1010 DIM v$(512), w$(512), byte1%(8), byte2%(8)
1020 begynd = 0 : peak= 0 : val% = 0 : a% = 0 : pos% = 0 : x = 0
1030 OPEN#3,ser: PRINT#3;CHR$(27);CHR$(51);CHR$(24);
1040 OPEN#10; ram3_vh: INPUT#10; begynd : CLOSE#10

1060 FOR start= begynd TO begynd + 19456 STEP 1024
1070  v$= '': w$= ''
1080  FOR blok= start TO start + 62 STEP 2
1090   DIM byte1%(8), byte2%(8): peak= blok: val%= 128
1100   REPeat linie14
1110    a%= 128: pos%= 1
1120    REPeat omdan
1130     IF a% && PEEK(peak) OR a% && PEEK(peak+1) THEN
1140       byte1%(pos%)= byte1%(pos%) + val% * 1.5
1150     END IF
1160     pos%= pos% + 1: IF pos% > 8: EXIT omdan
1170     a%  = a% / 2
1180    END REPeat omdan
1190    val%= val% / 4: peak= peak + 128: IF val% < 2: EXIT linie14
1200   END REPeat linie14
1210   val%= 128
1220   REPeat linie58
1230    a%= 128: pos%= 1
1240    REPeat omdan
1250     IF a% && PEEK(peak) OR a% && PEEK(peak+1) THEN
1260       byte2%(pos%)= byte2%(pos%) + val% * 1.5
1270     END IF
1280     pos%= pos% + 1: IF pos% > 8: EXIT omdan
1290     a%  = a% / 2
1300    END REPeat omdan
1310    val%= val% / 4: peak= peak + 128: IF val% < 2: EXIT linie58
1320   END REPeat linie58
1330   FOR x = 1 TO 8: v$ = v$ & CHR$(byte1%(x)) & CHR$(byte1%(x))
1340   FOR x = 1 TO 8: w$ = w$ & CHR$(byte2%(x)) & CHR$(byte2%(x))
1350  END FOR blok
1360  PRINT#3; CHR$(27);CHR$(75);CHR$(0);CHR$(2);: PRINT#3; v$
1370  PRINT#3; CHR$(27);CHR$(75);CHR$(0);CHR$(2);: PRINT#3; w$
1380 END FOR start
1390 PRINT#3;CHR$(27);CHR$(51);CHR$(36);: CLOSE#3
1400 PRINT#3;CHR$(12);: CLOSE#3
```

Forenklet udgave af "cpretkopi". 640 dots per line (linie 1360). Det udnyttes, at skærmens x,ℓ og h alle er multipla af 8.

"qperkopi_bas" (og tilhørende boot m.v.) har ikke TRA 1, TRA 0, BAUD 4800, men bruger DEFAULT-værdierne BAUD 9600 og TRA 0. Der benyttes 720 dots per line. Derfor må linie 1360 og 1370 ændres: CHR$(75); erstattes af CHR$(42); CHR$(6);

G_SAVE mm. (Marcus Jeffrey, QL WORLD, september 1981)

"G_SAVE" (LRESPR savescr, se bl.a. "flerkopi", pag. 31 ff) er en lynhurtig rutine, der SAVEr en given del af skærmbilledet ved kommandoen:

G_SAVE x,y,ℓ,h,adr hvor x,y er (pixel-)koordinaterne til øverste, venstre hjørne, ℓ,h er (pixel-)længde og højde af den del af skærmen, der skal lagres, og adr er en reserveret adresse, der generelt bør etableres med:

adr = RECHP (antalbytes), hvor antalbytes er:

antalbytes = ℓ · h / 4 + 8, idet adr + 2 indeholder ℓ, adr + 6 indeholder h, og selve pixelinformationen, opbygget analogt til skærminformationen, begynder i adr + 8.

(start + 0 og start + 4 er tomme - har vel været overvejet til hhv. x og y ?).

Skærmbilledet kan lynhurtigt placeres hvor som helst på skærmen (ikke udenfor, naturligvis) ved kommandoen:

G_LOAD x,y,adr

Når billedet (skærmudsnittet) ikke mere skal bruges, bør den etablerede adresse fjernes med:

RECHP (adr), således at COMMON HEAP frigiver hukommelse.

Se iøvrigt TOOLKIT2-vejledningen.

Når jeg siger, at en given del kan SAVEs og LOADes, er det imidlertid ikke helt rigtigt:

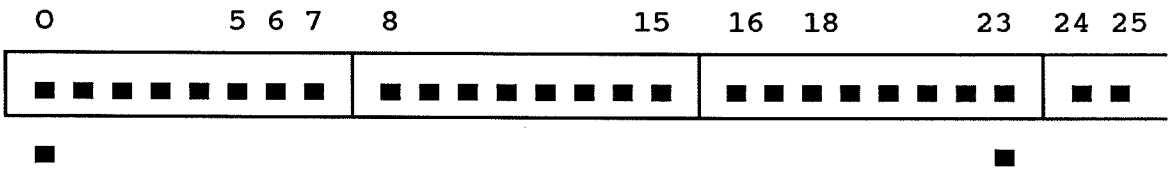
DER GÆLDER KUN, HVIS x og ℓ ER MULTIPLA AF 8.

G_SAVE opfører sig meget ejendommeligt, hvis x og ℓ ikke er multipla af 8, idet x altid rundes nedad (så der eventuelt kommer for meget med til venstre), mens ℓ rundes opad (så der kommer for lidt/korrekt/for meget med). G_LOAD benytter denne information uden nogen korrektion.

Dette bygger på, at der læses og lagres efter skærminformationens opbygning, 8 pixler ad gangen: Kritikløst, enkelt, hurtigt.

Det har jeg "repareret" på. De følgende eksempler viser mine rutiner.

Pixler nummer:



Skærmkopi med x = 0 og ℓ = 24 er 24 pixler (0....23).

"G_SAVE" og "scrkopi" medtager begge nøjagtigt det ønskede.

MEN: ■ x = 6, ℓ = 13 (6..18) ■

"G_SAVE" runder x ned til 0 og runder ℓ op til 24.
"scrkopi" runder x ned til 0 og runder ℓ + x op til 24.
Begge medtager for meget i begge ender (pixler nr. 0....23).
"scrkopi" reducerer billedet til det ønskede udsnit.

MEN: ■ x = 5, ℓ = 21, (5....25) ■

"G_SAVE" runder x ned til 0 og runder ℓ op til 24 (nr. 0....23).
"scrkopi" runder x ned til 0 og runder ℓ + x op til 32
(nr. 0....31).

"G_SAVE" har altså nu 8 pixler for lidt med, til højre. Af disse skulle de første to være udprintet.

"scrkopi" kan ikke læse "G_SAVE"s information, der består af 2 x 8 pixler pr. linie, hvor "scrkopi" forventede 3 x 8 pixler pr. linie.

Jeg vælger at narre G_SAVE med en korrektion til ℓ, ud fra følgende:

Hvis både x og ℓ er multipla af 8, har de to rutiner samme startpixel = x og slutpixel = x + ℓ - 1.

Generelt gælder for "G_SAVE":
startpixel = x -(x MOD 8)
slutpixel = x -(x MOD 8) + ℓ - 1 (ℓ afrundes ikke)
IF (ℓ MOD 8): slutpixel = slutpixel + 8 - (ℓ MOD 8)
(ℓ rundes opad)

Generelt gælder for "scrkopi":
startpixel = x -(x MOD 8)
slutpixel = x + ℓ - 1 (x + ℓ afrundes ikke)
IF ((ℓ + x)MOD 8): slutpixel = slutpixel + 8 -((ℓ + x)MOD 8)
((x + ℓ) rundes opad)

"scrkopi" ønsker en større slutpixel end "G_SAVE" når:

$$x + \ell - 1 + (8 - ((\ell + x) \text{ MOD } 8)) \cdot ((\ell + x) \text{ MOD } 8)$$

er større end:

$$x - (x \text{ MOD } 8) + \ell - 1 + (8 - (\ell \text{ MOD } 8)) \cdot (\ell \text{ MOD } 8)$$

dvs.

$$(x \text{ MOD } 8) + (8 - ((\ell + x) \text{ MOD } 8)) \cdot ((\ell + x) \text{ MOD } 8) > (8 - (\ell \text{ MOD } 8)) \cdot (\ell \text{ MOD } 8)$$

Det er et noget uhåndterligt udtryk.

I PROCeduren "skærmkopi" benyttes korrektionen:

xkorr = x MOD 8 (antal pixler til venstre, der ikke skal med)
ℓkorr = y MOD 8

```
SELECT ON xkorr
= 0 : korr = 0        Hvis x er et multiplum af 8,
                      bliver ℓ altid afrundet korrekt

= 1 : IF NOT ℓkorr: korr = 8.
                      Hvis x = 1 og ℓkorr er et multiplum af 8,
                      skal ℓ altid tillægges 8 pixler.

=REMAINDER: IF ℓkorr > 8 - xkorr : korr = 8,
                      Hvis fx. x = 2....7 er xkorr = x,
                      og hvis ℓkorr > 8 - x, lægges 8 pixler til ℓ

END SELECT
```

Hvis fx x = 7, er tillægget til ℓ: korr = 8,
for alle værdier af ℓ, undtagen for ℓkorr = 1 (fx. ℓ = 17).

Korrektionen benyttes, når skærmudsnittet SAVEs:

G_SAVE x, y, ℓ + korr, h, adresse.