0

```
\\ Directory ultraFORTH 4of4   26oct87re
.                          0
..                         0
C16-Tape-Demo              2
C64-Grafik-Demo            6
cload/csave              &13
Tape-Version:LoadScreen  &16
Ramdisk                  &21
Supertape                &32
auto-Decompiler          &51
Screentausch             &61
Grafik                   &64
Mathematik               &90
Sieve Benchmark         &138
Grafik-Demo             &144
Sprite-Demo             &160
Sprite-Data             &165
Sprite-Editor           &166
```

```
\ Tu heading left right        20oct87re

| Variable xpos     | Variable ypos
| Variable deg      | Variable pen

| : 100*/  ( n1 n2 n3 - n4)  &100 */ ;

: heading    ( - deg)    deg @ ;
: setheading ( deg -)    deg ! ;

: right  ( deg -)
 deg @ swap - &360 mod deg ! ;

: left   ( deg -)
 deg @ + &360 mod deg ! ;

' clrscreen  Alias cs
' pencolor   Alias pc
' background Alias bg
' hires      Alias fullscreen
' window     Alias splitscreen
```

1

```
\\ Inhalt ultraFORTH 4of4      26oct87re
Directory                       0
Inhalt                          1

C16-Tape  -Demo           &2-  &5
C64-Grafik-Demo           &6- &12
cload csave              &13- &15
Tape-Version:LoadScreen  &16- &20
Ramdisk                  &21- &30
                              &31 frei
Supertape                &32- &50
automatischer Decompiler &51- &60
Screens via UserPort C64 &61- &63
Grafik      nur C64 !!   &64- &88
                              &89 frei
Mathematik               &90- &96
                         &97-&100 frei
Tape Ramdisk Supertape  &101-&135 shadow

Sieve Benchmark             &138
Grafik-Demo nur C64 !!  &144-&155
Sprite-Demo nur C64 !!  &160-&164
Sprite-Data                 &165
Sprite Editor           &166-&168
```

```
\ Tu positions pen home        20oct87re

: xcor   ( - x)     xpos @ 100u/ ;
: ycor   ( - y)     ypos @ 100u/ ;

: setx   ( x -)     100* xpos ! ;
: sety   ( y -)     100* ypos ! ;
: setxy  ( x y -)   sety setx ;

: pendown  pen on ;
: penup    pen off ;

: home
 &160 &96 setxy &90 setheading pendown ;

: draw     clrscreen home &20 window ;
: nodraw   text page ;

' left       Alias lt
' right      Alias rt
' setheading Alias seth
' pendown    Alias pd
' penup      Alias pu
```

2

```
\ DemoL:C16Tape-Demo ?dload   clv10oct87
\ Demo: insgesamt ca. 80 Screens !!!
\ prueft, ob ein Wort vorhanden ist:

| : exists? ( string--flag)
      cr capitalize dup find nip under
      0= IF ." not " THEN ." found: "
      count type ;

\ Die letzte zugegriffene Diskette:

| Variable LastDisk   -1 LastDisk !

\ Laedt SCR von DISK, wenn das Wort
\ namens STRING noch nicht vorhanden
| : ?dload  ( string scr disk--)
  2 pick exists?
  IF drop drop drop exit THEN
  dup LastDisk @ -
  IF flush ."  Insert #" dup .
     key drop  dup LastDisk ! THEN
  drop ."  scr#" dup . cr
  load exists? 0= error" ???" ;
-->
```

```
\ Tu forward back              20oct87re

: tline   ( x1 y1 x2 y2 -)
 >r >r >r  100u/  r> 100u/
           r> 100u/  r> 100u/  line ;

: forward  ( distance -)
 >r xpos @ ypos @
 over deg @ cos r@ 100*/ + dup xpos !
 over deg @ sin r> 100*/ + dup ypos !
 pen @ IF tline ELSE 2drop 2drop THEN ;

: back     ( distance -)
 negate forward ;

: turtlestate ( - pen bg pc)
 pen c@ colram c@ dup
 &15 and swap &16 / ;

' forward     Alias fd
' back        Alias bk
' turtlestate Alias ts
```

3 88

```
0 \ DemoL:?reloc                clv10oct87
1
2 \ verschiebt das System ggf./ macht COLD
3 : ?reloc  ( s0 r0 limit --)
4  dup          limit =
5  2 pick origin $a + @ = and
6  3 pick origin  8 + @ = and
7
8  IF drop drop drop exit THEN
9        [’] limit >body ! \ limit
10            origin $A + ! \ r0
11     dup 6 + origin   1+ ! \ task
12            origin  8 + ! \ s0
13  cold ;
14
15 \ Kompiliert Woerter, die erst spaeter
16 \ geladen werden.
17 : (forward"     "lit capitalize find
18  IF execute
19  ELSE count type ." unsatisfied" quit
20  THEN ; restrict
21 : forward"  compile (forward" ," ;
22  immediate  restrict
23 -->
24
```

```
\ Gr arc ellipse circle        20oct87re

: arc     ( hr vr strt end -)
 >r >r 2dup max &360 swap /
 r> 2* 2* r> 1+ 2* 2* swap rot >r
  DO over I 2/ 2/ cos &10005 */
     over I 2/ 2/ sin &10005 */
     plot
  r@ +LOOP
 r> 2drop drop ;

: ellipse ( x y hr vr -)
 2swap c-y ! c-x ! m-flag on
 0 &90 arc m-flag off ;

: circle  ( x y r -)
 dup 3 4 */ ellipse ;
```

4 89

```
0 \ DemoL:64kb C16Demo          clv10oct87
1
2 \ stellt das System auf 64kB wenn vorh.
3
4 : 64kb $533 @ $fd00 -  ?dup
5  IF cr u. ." too small" exit THEN
6  limit $fd00 -
7  IF $8000 $8400 $fd00 ?reloc THEN ;
8
9 \ wird als 'RESTART installiert:
10
11 : c16demo  cr ." c16-Demo"
12   forward" tapeinit"
13   0 drive   forward" floppy"
14   cr ." Type 'help' to get help"
15   cr ." Type '64kb' to use 64kb" ;
16
17
18
19
20
21
22
23 -->
24
```

5 90

```
0 \ DemoL:C16DemoLoad          cclv14oct87
1
2 \ Dieses Wort laedt die komplette
3 \ Demo-Version. Wird als 'COLD install.
4 \ und installiert spaeter C16DEMO
5
6 : c16DemoLoad
7   $9000 $9400 $c000 ?reloc
8   Forth      " Code"       5 3 ?dload
9   Forth      " Editor"   $13 3 ?dload
10  Forth      " debug"    $2f 3 ?dload
11  Forth      " help"      $a 1 ?dload
12  Forth      " Tapeinit" $10 4 ?dload
13  [’] noop    Is 'cold
14  [’] c16demo Is 'restart
15  forward" Editor" forward" Ediboard"
16  1 scr !  0 r# !       save
17  $7a00 $7bf0 $8000 ?reloc ;
18
19 ' c16DemoLoad Is 'cold  save
20
21 cr .( Type : cold)
22 cr .( after all: savesystem!!!)
23
24
```

```
\ Mathematik Load-Screen       20oct87re

Onlyforth

base @  decimal

   1  2 +thru  \ Trigonometrie
   3  4 +thru  \ Wurzeln
   5  6 +thru  \ 100* 100u/

base !
```

6 91

```
0 \ Graphik-Demo fuer C64        23oct87re
1
2 (16 .( Nicht fuer C16!) \\ C)
3
4 Onlyforth
5
6 \needs buffers      .( Buffers?!)    \\
7 \needs demostart    .( Demostart?!)  \\
8 \needs tasks        .( Tasker??!)    \\
9 \needs help         .( help??!)      \\
10 \needs graphic     &58 +load
11 \needs .message2   1 2 +thru
12 Graphic also
13 \needs moire       6 +load
14
15 \needs slide  &154 +load  \ the Demo
16
17  3 5 +thru
18
19 1 Scr !  0 R# !
20
21 save
22
23
24
```

```
\ Ma sinus-table                 20oct87re
\    Sinus-Tabelle nach FD Vol IV/1

: table    ( values n -)
 Create 0 DO , LOOP
 ;code      ( n - value)
 SP X) lda  clc  1 # adc  .A asl  tay
 W )Y lda  SP X) sta
 iny  W )Y lda  1 # ldy  SP )Y sta
 Next jmp  end-code

10000 9998 9994 9986 9976 9962 9945 9925
 9903 9877 9848 9816 9781 9744 9703 9659
 9613 9563 9511 9455 9397 9336 9272 9205
 9135 9063 8988 8910 8829 8746 8660 8572
 8480 8387 8290 8192 8090 7986 7880 7771
 7660 7547 7431 7314 7193 7071 6947 6820
 6691 6561 6428 6293 6157 6018 5878 5736
 5592 5446 5299 5150 5000 4848 4695 4540
 4384 4226 4067 3907 3746 3584 3420 3256
 3090 2924 2756 2588 2419 2250 2079 1908
 1736 1564 1392 1219 1045 0872 0698 0523
 0349 0175 0000

@ &91 ¦ table sintable                =
```

7 92

```
0 \ demo-version                 06nov87re
1
2 ¦ : (center."  "lit count
3  C/L over - 2/ spaces type cr ;
4 restrict
5
6 ¦ : c."  compile (center." ," ;
7 immediate restrict
8
9 ¦ : .FGes c." Forth Gesellschaft e.V." ;
10
11 ¦ : .uF83  c." *** ultraFORTH-83 ***" ;
12
13 ¦ : .(c)   c." (c) 1985/86/87"
14 c." Bernd Pennemann  Klaus Schleisiek"
15 c." Georg Rehfeld  Dietrich Weineck"
16 c." Claus Vogt" ;
17
18 ¦ : .bezug   c." rainer mertins"
19              c." antilopenstieg 6"
20       cr     c." 2000  hamburg 54"  ;
21
22 ¦ : wait   BEGIN  key 3 -  UNTIL ;
23
24
```

```
\ Ma sin, cos, tan               20oct87re

¦ : s180   ( deg -- sin*10000:sin 0-180)
 dup &90 >
   IF &180 swap - THEN
 sintable ;

: sin      ( deg -- sin*10000)
 &360 mod dup 0< IF &360 + THEN
 dup &180 >
    IF &180 - s180 negate
    ELSE s180 THEN ;

: cos      ( deg -- cos*10000)
 &360 mod &90 + sin ;

: tan      ( deg -- tan*10000)
 dup sin swap cos ?dup
   IF &100 swap */ ELSE 3 * THEN ;

@                                    1
```

8 93

```
0 \ demo-version                 20oct87re
1
2 : .message1  ( -- )   singletask
3  page .uF83 cr .(c) cr
4  c." Das Kopieren und Verschenken"
5  c." dieses Programms ist ausdruecklich"
6  cr c." * erlaubt ! *"
7  cr c." Jeglichen Missbrauch zum"
8  c." Zwecke der Bereicherung"
9  c." werden wir nach besten Kraeften"
10 c." verfolgen und verhindern."
11 cr c." Die Mitglieder der" .FGes
12 multitask wait ;
13
14 : .message2  ( -- )
15  page c." Du hast jetzt ein"
16  c." arbeitsfaehiges System mit"
17  c." Editor, Debugger und Assembler!"
18  c." Nach Einlegen einer formatierten"
19  c." Diskette kannst Du es mit"
20  c." SAVESYSTEM <name> (z.B. FORTH)"
21  c." als Programmfile abspeichern."
22  cr .uF83 cr
23  c." Bezug und Mitgliedschaft in der"
24  .FGes c." ueber:" cr .bezug wait ;    =
```

```
\ Ma sqrt 1                      20oct87re

Code d2*  ( d1 - d2)
 2 # lda setup jsr
 N 2+ asl N 3 + rol  N rol N 1+ rol
 SP 2dec N 3 + lda SP )y sta
 N 2+ lda SP x) sta
 SP 2dec N 1+ lda SP )y sta
 N lda SP x) sta
 Next jmp end-code

: du< &32768 + rot &32768 + rot rot d< ;
¦ : easy-bits  ( n1 -- n2)
 0 DO
  >r d2* d2*  r@ -  dup 0<
    IF   r@ +   r> 2* 1-
    ELSE        r> 2* 3 +
    THEN LOOP ;

¦ : 2's-bit
 >r d2* dup 0<
  IF    d2* r@ - r> 1+
  ELSE  d2* r@ 2dup u<
    IF drop r> 1-  ELSE -  r> 1+  THEN
  THEN ;                                 ╪
```

9

```
0 \ demo-version                20oct87re
1
2 graphic  also
3
4 ¦ Variable end?
5
6 : killdemo  ( -)
7  killsprites endslide
8  singletask  .message2
9  ['] 1541r/w Is r/w
10 ['] noop Is 'cold
11 ['] noop Is 'restart
12 ['] (quit   Is 'quit
13 nographic
14 [ ' demostart >name 4 - ] Literal
15 (forget  save  &16 buffers ;
16
17
18
19
20
21
22
23
24
```

94

```
\ Ma sqrt 2                     20oct87re

¦ : 1's-bit
 >r dup 0<
  IF 2drop r> 1+
  ELSE d2* &32768 r@  du< 0=
    negate R> +
  THEN ;

: sqrt    ( ud1 - u2)
 0 1  8 easy-bits
 rot drop 6 easy-bits
 2's-bit 1's-bit ;

\\

: xx
 &16 * &62500 um*
 sqrt 0 <# # # # # ascii . hold #s #>
 type space ;
```

@ =

10

```
0 \ demo-version                06nov87re
1
2 ¦ : demor/w  ( adr blk r/wf - f)
3  end? @  0 max  dup small  red colored
4  -1 end? +!  sprite push  killsprites
5  1541r/w ;
6
7 ¦ : demoquit
8   BEGIN .status cr query interpret
9    state @ IF  ." compiling"
10           ELSE ." uF83" THEN
11   end? @ 0< dup
12   IF drop
13    cr ." Kill the Demo? n/y "
14    key capital Ascii Y =
15    dup not IF  del del del  THEN
16   THEN
17  UNTIL  killdemo ;
18
19
20
21
22
23
24
```

95

```
\ 100*                          20oct87re

Code 100*  ( n1 - n2)
 SP X) lda  N sta  SP )Y lda  N 1+ sta
 N asl N 1+ rol  N asl N 1+ rol

 N lda N 2+ sta  N 1+ lda N 3 + sta

 N 2+ asl N 3 + rol  N 2+ asl N 3 + rol
 N 2+ asl N 3 + rol

 clc N lda N  2+ adc N sta
  N 1+ lda N 3 + adc N 1+ sta

 N 2+  asl N 3 + rol

 clc N lda N  2+ adc  SP X) sta
  N 1+ lda N 3 + adc  SP )Y sta

 Next jmp end-code
```

@ ⊥

11

```
0 \ demo-version                20oct87re
1
2 : demonstration
3  Onlyforth graphic
4  ['] demor/w Is r/w
5  ['] killdemo Is 'cold
6  slide multitask pause   4 end? !
7  ['] demoquit Is 'quit
8  ['] (error errorhandler !
9  ['] noop Is 'abort
10 .message1  linien text
11 key drop  moire text  key drop
12 ." help" row 1- 0 at abort ;
13
14 ' demonstration Is 'cold
15 ' killdemo Is 'restart
16
17
18
19
20
21
22
23
24
```

96

```
\ 100/                          20oct87re

Label 4/+
 N 7 + lsr N 6 + ror N 5 + ror N 4 + ror
 N 7 + lsr N 6 + ror N 5 + ror N 4 + ror
 clc N  lda N 4 + adc N     sta
  N 1+  lda N 5 + adc N 1+  sta
  SP X) lda N 6 + adc SP X) sta
  SP )Y lda N 7 + adc SP )Y sta  rts

Code  100u/  ( u - n)
 N stx  N 4 + stx
 SP X) lda  .A asl N 1+  sta  N 5 + sta
 SP )Y lda  .A rol SP X) sta  N 6 + sta
 txa .A rol       SP )Y sta  N 7 + sta
 4/+ jsr
 N 7 + lsr N 6 + ror N 5 + ror N 4 + ror
 4/+ jsr
 Next jmp end-code
```

≡ ╪

12 97

```
0 \ hires demo worte            06nov87re
1
2 : linien
3  clrscreen yel blu colors hires
4  &320 0 DO
5     &320 0 DO I &198 J 0 line &35 +LOOP
6  &35 +LOOP ;
7
8 : moire
9  clrscreen ora red colors hires
10 &320 0 DO
11  I &198 &319 I - 0 line
12 3 +LOOP
13 &199 0 DO
14  &319 &198 I - 0 I line
15 2 +LOOP ;
16
17
18
19
20
21
22
23
24                              @                                =
```

13 98

```
0 \ cSave cLoad..               clv10oct87  \\ zu: csave cload          clv10oct87
1
2 Onlyforth
3 \needs Code   .( ?! Code ?!) quit        Der Assembler muss schon da sein
4
5 $ff90 >label setMsg   $90 >label status
6 $ffba >label setlfs $ffbd >label setNam   Labels setzen
7 $FFD8 >label BSAVE  $FFD5 >label BLOAD
8 Label slPars
9  setup jsr (16 rom C)                     Parameter ab N ablegen
10 $80 # lda setMsg jsr 0 # lda status sta  SystemMeldungen ein   Status auf 0
11 N lda sec 8 # sbc  (drv    sta           (drv setzen fuer derror?
12       CC ?[ dex ]? (drv 1+ stx           Geraete#, Sek.Adresse, File#
13 N ldx     N  1+ ldy 1 #  lda setlfs jsr  Adresse-des-Filenamens Laenge
14 N 4 + ldx N 5 + ldy N 2+ lda setnam jsr  Adresse in XY
15 N 6 + ldx N 7 + ldy
16 rts end-code
17 Label slErr \ AR=Kernalfehler            Einer der 8 Kernalfehler?
18 CC ?[ 0 # lda ]? pha                     Status begucken/EOI-Bit vernichten
19 status lda $bf # and                     beides zusammen als Fehler# zurueck
20 (16 ram C) push jmp end-code
21 -->
22
23
24 FORTH-GESELLSCHAFT (c) bp/ks/re/we/clv  @                              1
```

14 99

```
0 \ ..cSave cLoad               clv10oct87  \\ zu: ..csave cload        clv10oct87
1
2 Code cSave ( f t+1 Name Nlen dev--err)
3  5 # lda    SLPars jsr                    Parameter vorbereiten (XR=to+1)
4  N 8 + # lda bsave jsr                    Zeiger auf from in AR und BSAVE
5  slErr jmp end-code                       Fehler?
6
7 Code cLoad ( f Name Nlen dev--t+1 err)
8  4 # lda    SLPars jsr                    Parameter vorbereiten (XR=from)
9  0 # lda     bload jsr                    Load (nicht Verify) BLOAD
10 php pha tya pha txa pha 0 # ldy          to+1 wird zurueckgegeben und
11 SP 2dec pla SP )y sta iny pla SP )Y sta  sorgfaeltig auf den Forth-Stack getan
12 pla plp slErr jmp  end-code              Fehler?
13
14 -->
15
16 \\ moegliche Fehler                      Fehlerfundstellen bei CBM-Routinen:
17  AR CF ST                Basic  Forth    (1) Kernal-Rueckgabe
18  xx  L 00 kein Fehler      0      0      (2) Status-Register
19  00  H 00 stop-taste      1e     1e      (3) Disketten-Fehlerkanal
20  00  L 60 end-of-tape     04     00
21  00  L 10 load/verify-error 1d/1c 1d
22  00  L 60 Pruefsummenfehler 1d    1d
23 0-8  H 00 Kernal-Fehler    0-8   0-8
24 FORTH-GESELLSCHAFT (c) bp/ks/re/we/clv  =                              +
```

15 100

```
0 \ ..cSave cLoad Luxus          clv10oct87  \\ zu: ..csave cload Luxus    clv10oct87
1
2
3 Code .err ( err#-err# ) \ druckt Meldung
4  SP x) lda 0>=                             Dieser Routine proekelt sich die
5  ?[ (16 tax dex rom $8654 jsr C)           Basic-Fehlermeldungen raus, damit
6     (64 .A asl tax rom                     ich nicht alles nochmal abschreiben
7     $a326 ,x lda $24 sta                   muss. Dazu benutzt sie das Basic-Rom,
8     $a327 ,x lda $25 sta dey C) dey        was man nur machen sollte, wenn sicher
9     [[ iny $24 )y lda php  $7f # and       keine Seiteneffekte auftreten!
10       $ffd2 jsr plp 0< ?]                 Dies ist hier der Fall.
11    (16 ram C) (64 ram C)
12   ]? xyNext jmp end-code
13
14
15 : derr?  ( err# -- flag)
16  dup IF cr dup u. .err ." error" THEN     macht aus der zurueckgegebenen Nummer:
17  dup $ff and 5 = not                      eine Fehlerausgabe
18  (drv @ -1 > and                          Wenn nicht grade "device not presen"
19  IF derror? or THEN                       ist, fragt es auch noch den seriellen
20  (drv @ 0 max (drv ! ;                    Bus nach des Geraetes Fehlermeldung
21
22 \\ zur Benutzung nach CSAVE und CLOAD.
23    Die letzte Zeile ist nur fuer
24    Kompatibilitaet zur alten Version.  @                              =
```

16 101

```
0 \ TapeVersion:LoadScreen       clv12oct87  \\ zu TapeVersion          clv01aug87
1
2 Onlyforth                                  Die Kassettenversion wurde speziell
3                                            fuer C16 mit 64kB entwickelt, laueft
4 \needs Code  .( ?! Code !?)   quit         aber auch auf C64.
5
6              5 +load    \ Ramdisk         Sie besteht auf 3 Teilen:
7             -3 +load    \ csave/load        Einer im Speicher simulierten Floppy
8           1  3 +thru    \ Tape              Einer Schnittstelle zum externen
9        (16 $10 +load C) \ superTape            Geraet (i.a.Recorder)
10             4 +load    \ savesys           Einem Schnelllader (Supertape)
11                                               (allerdings nur fuer C16)
12 Onlyforth
13 Variable autoload   autoload off
14
15 : tapeInit  cr cr ." Tape2.00 "
16  \if supertape supertape                  Initialisieren:
17  ['] ramr/w Is r/w  1 drive                ggf. Supertape initialisieren
18  autoload @                                R/W umdefinieren und aktivieren
19  IF autoload off loadramdisk THEN ;        falls AUTOLOAD gesetzt ist, gleich
20                                            noch eine ramdisk laden.
21 save
22 ' tapeInit Is 'restart
23 \ restart
24                                  @                                   1
```

17 102

```
0 \ store restore                clv24jul87
1
2 \ wie push pull abort"
3
4 ¦ Create restore 0  ] r> r> ! ;
5
6 : store ( addr -- )
7  r> swap dup >r @ >r  restore >r >r  ;
8  restrict
9 \ rstack:   restore date adresse ....
10
11 ¦ : back \ -- \ rewinds rstack
12  r> BEGIN rdepth WHILE
13       r> restore =
14       IF r> r> ! THEN REPEAT >r ;
15
16 : (restore"    "lit swap IF
17    >r clearstack r> back
18    errorhandler perform
19  exit THEN  drop ;  restrict
20
21
22 : restore"  compile (restore" ," ;
23  immediate  restrict
24                                  ≡                                   ╪
```

18 103

```
0 \ tape-interface              clv01aug87
1
2 \needs cload   .( ?! cload ?!)   quit
3 \needs restore .( ?! restore ?!) quit
4
5 Variable device      0 device !
6 : commodore     1 device ! ; \ Geraet..
7 : floppy        8 device ! ;
8
9
10 : bload  ( [from name count -- ]to)
11   device @ cload derr? restore" load" ;
12
13 : bsave  ( [from ]to name count--)
14   device @ csave derr? restore" save" ;
15
16 : n" ( -- adr count) Ascii " parse ;
17
18
19
20
21
22
23
24
```

19 104

```
0 \ Ramdisk TapeInterface       clv29jul87
1
2 Onlyforth Ramdisk also
3
4 : saveRamDisk
5   rd behind id count bsave ;
6
7
8 : loadRamDisk
9  rd? 0=
10  IF range memtop  rdnew rd THEN
11   " RD." count bload drop ;
12
13
14
15
16
17
18
19
20
21
22
23
24
```

20 105

```
0 \ \if savesystem"             clv01aug87
1
2 \needs restore" .( ?! restore" ?!) quit
3
4 Onlyforth
5
6 : \if name find 0=
7  IF [compile] \ THEN drop ; immediate
8
9 : savesystem \ -- Name muss folgen
10  \ Forth-Kernal a la boot:
11    scr store 1 scr ! r#  store 0 r# !
12  \ Editor  a la boot
13   \if Editor  [ Editor ]
14   \if Editor stamp$ store stamp$ off
15   \if Editor (pad   store (pad   off
16    save
17  \ Supertape? dann andere Routine
18   \if supertape device @ 7 =
19   \if supertape IF stSavSys exit THEN
20  \ nun geht's los
21    origin $17 - here n" bsave ;
22
23
24
```

21

```
\ RD: loadscreen              clv01aug87

Onlyforth

(16 $fd00 C) (64 $c000 C)
Constant memtop

Vocabulary Ramdisk
Ramdisk also definitions

      1   9 +thru

Onlyforth
```

\\ zu RD: loadscreen clv05aug87

Die hier vorgestellte Ramdisk benutzt ein komprimierendes Format.

Um ein Umschalten zwischen mehreren Ramdisk's zu ermoeglichen, enthaelt die einzige Variable (RD einen Zeiger auf die Ramdisk, alle weiteren Variablen stehen im Speicherbereich der Ramdisk.

Binaerbloecke muessen mit BINARY deklariert werden, ansonsten unterstuetzt die Ramdisk alle Forth-Worte, die ueber R/W gehen.

106

22

```
\ RD: Grundlagen              clv01aug87

Variable (rd     (rd off
$31 constant plen

: adr>   ( adr--ofs) (rd @ -          ;
: >adr   ( ofs--adr) (rd @ +          ;
: adr@   ( ofs--adr) >adr @ >adr      ;
: rd?    ( -- adr flag )
   (rd @ dup   dup   @ plen =     and ;
: rd     ( -- adr)
   rd? 0= abort" no Ramdisk" ;

: take   ( adr--   ) adr> 2 >adr !  ;

: adr    ( --adr   ) 2   adr@        ;
: data   ( --adr   ) adr 4 +         ;

: end    ( --adr   ) 4   adr@        ;
: behind ( --adr   ) end 4 +         ;
: end+   ( len--   ) 4   >adr +!     ;

: blk#   ( --adr   ) 8   >adr        ;
: id     ( --adr   ) $10 >adr        ;
```

\\ zu RD: clv01aug87

\ Alle Zeiger sind offsets auf First

rd ==0 ==> keine Ramdisk

rd -->Laenge des Parameterblock
+2 -->aktueller Block
+4 -->Ende des letzten Blocks+1
+6 -->Ende des Ramdisk-Bereich+1
+8 -->Nummer des aktuellen Blocks
+16-->Name
Ende des Parameterblocks
     1.RD-Block
     2.RD-Block
       .
     0000

adr-->aktueller RD-Block (absolute Adr.)
   -->Laenge (incl. 4 bytes Verwaltung)
 2+-->Blocknummer
 2+-->..Daten..

107

23

```
\ RD: new delete len@ len!    clv01aug87

: ?full      end 6 adr@ b/blk - 4 -
           u> abort" Ramdisk full" ;

: new ( --)  end take ?full ;

: len! ( len--) \ neuen Block beenden
 ?dup 0= ?exit
 blk# @   end 2+ !  4 + dup end !
 end+  end off ;

: len@ ( --len) \ Laenge ermitteln
 adr @ dup 0= ?exit 4 - ;


: delete  ( --)    \ Loeschen
 adr dup @ under + adr behind over -
 cmove
 negate end+ ;
```

\\ zu RD: clv01aug87

NEW prueft ob genug Platz ist
und setzt aktuellen Block
auf den ersten freien Platz

LEN! Traegt die Laenge des neuen
Blocks ein und veraendert END
Wenn die Laenge=0 ist, geschieht nix
Erzeugt 0000 am Ende der Ramdisk

LEN@ gibt die Laenge des aktuellen
Blocks zurueck. Wenn er nicht
vorhanden ist, gibt es 0 zurueck

DEL Loescht den aktuelen Block
Veraendert END

108

24 109

```
0 \ RD: search binary          clv01aug87  \\ zu RD:                    clv01aug87
1
2 : search ( blk --) \ setzt akt. Block     SEARCH setzt akt. auf gesuchten Block
3  rd BEGIN dup @ + dup @ WHILE               Wenn nicht vorhanden: auf END
4   ( blk adr ) 2dup 2+ @ = UNTIL             Legt BlockNummer in BLK# ab.
5  take  blk# ! ;
6
7 : notRD? ( blk--flag) blk/drv u< ;
8
9
10
11
12
13
14
15
16 Onlyforth Ramdisk also
17
18 : binary ( blk--blk) \ no ComPand         BINARY untersagt das Komprimieren des
19  dup offset @ + notRD? ?exit               Blocks z.B. fuer Binaerdaten.
20  dup block drop update                     Wird durch $400 Bytes Laenge erkannt.
21  delete new b/blk len! ;
22
23
24                                  @                                   =
```

25 110

```
0 \ RD: cbm>7bit 7bit>cbm      clv01aug87  \\ zu RD: c>7 7 >c          clv01aug87
1
2 Label cbm>7b \ AR=char -- 7bitChar        Umwandeln der CBM-Zeichen in 7bit
3  $80 # cmp 0< ?[ rts ]?                   Die Zeichen $c0..$e0 werden zu $60..80
4  $c0 # cmp CS ?[ $e0  # cmp CC ?[         Alle anderen Zeichen >=$80  zu $00..20
5       $a0 # adc rts ]? ]?
6  $1f # and       rts end-code
7 Label 7b>cbm \ AR=7bitChar -- char
8  $60 # cmp CC ?[ rts ]?
9  $a0 # sbc rts end-code
10
11 Code c>7 sp x) lda cbm>7b jsr putA jmp
12 Code 7>c sp x) lda 7b>cbm jsr putA jmp
13 end-code
14
15
16
17
18
19
20
21
22
23
24                                  @                                   1
```

26 111

```
0 \ RD: cp1 cp2                clv01aug87  \\ zu RD: cp1 cp2           clv01aug87
1
2 Label cp1 ( from to count--tocount)       Anfangsroutine fuer COMPRESS & EXPAND
3  3 # lda setup jsr N 2+ lda N 6 + sta
4  N 3+ lda N 7 + sta dey  $7f # ldx
5  N lda 0=
6  ?[ N 1+ lda 0= ?[ pla pla 0 # lda
7    push0a jmp ]? ][ N 1+ inc ]? rts
8
9 Label cp2                                 Endroutine fuer COMPRESS & EXPAND
10  sec N 2+ lda N 6 + sbc pha
11      N 3+ lda N 7 + sbc push jmp
12
13
14
15
16
17
18
19
20
21
22
23
24                                  =                                   ╪
```

27 112

```
0 \ RD: expand compress        clv01aug87  \\ zu RD: expand compress     clv01aug87
1
2 Code expand  cp1  jsr
3  [[ [[ N 4 + )y lda 0<
4   ?[ $7f # and tay tax bl # lda
5    [[ N 2+ )y sta dey 0< ?] iny
6    sec txa
7    N 2+ adc N 2+ sta CS  ?[ N 3+ inc ]?
8   ][ 7b>cbm jsr N 2+ )y sta N 2+ winc ]?
9   N 4 + winc  N dec 0= ?] N 1+ dec 0= ?]
10  cp2 jmp end-code
11
12 Code compress  cp1 jsr
13  [[ [[ N 4 + )y lda bl # cmp 0=
14   ?[ inx 0=
15    ?[ dex txa  N 2+ )y sta N 2+ winc
16     $80 # ldx ]?
17    ][ $80 # cpx 0>=
18     ?[ pha txa N 2+ )y sta N 2+ winc
19      $7f # ldx pla ]?
20     cbm>7b jsr N 2+ )y sta N 2+ winc ]?
21   N 4 + winc N dec 0= ?] N 1+ dec 0= ?]
22  $80 # cpx 0>=
23  ?[ txa        N 2+ )y sta N 2+ winc ]?
24  cp2 jmp end-code                    @                              =
```

28 113

```
0 \ RD: ramR/W                 clv01aug87  \\ zu RD:ramR/W              clv01aug87
1
2 ¦ : endwrite ( compLen--)                ENDWRITE entfernt Blanks am Blockende
3  data under + ( [from ]to )                 und setzt LEN!
4  BEGIN 1- dup c@ $7f u> WHILE
5    2dup u> UNTIL 1+ swap - len! ;
6
7 ¦ : endread  ( toAdr expLen--)           ENDREAD fuellt Rest des Blocks mit Blank
8  under + b/blk rot - bl fill ;
9
10 : ramR/W ( adr blk file R/NotW -- error)  RAMR/W  ersetzt die R/W-Routine
11  2 pick notRD?                             (binaere) Bloecke mit voller Laenge
12  IF 1541r/w                                werden per CMOVE uebertragen.
13  ELSE swap abort" no file"                 Kuerzere werden beim Schreiben
14   swap search len@ b/blk = ( adr r? b?)    mit COMPRESS und beim Lesen mit
15   IF   0= IF data ELSE data swap THEN      EXPAND verarbeitet.
16           b/blk cmove
17   ELSE 0= IF   delete new data b/blk
18                compress endwrite
19           ELSE dup data swap len@
20                expand endread
21
22  THEN THEN false THEN ;
23
24                                    @                                  1
```

29 114

```
0 \ RD: id rduse/del/new       clv01aug87  \\ zu RD:id rduse..          clv01aug87
1
2 : .rd  ( --)     (rd @ u. rd drop        .RD  gibt die zentralen Kenndaten
3   end u. 6 adr@ u. id count type ;            der Ramdisk aus
4
5 : id! ( adr count--)                     ID!  setzt den Namen
6   $20 id c! id count bl fill
7   $1a umin id 3 + place
8    " RD." count id 1+ swap cmove ;
9
10 : id" Ascii " parse id! ; \ Name" folgt  ID"  liest den Namen ein
11
12 : rduse ( from --) (rd ! ;              RDUSE schaltet (ohne Pruefung) um
13 : rddel  ( --)                          RDDEL loescht die Ramdisk
14   rd @ dup 2 >adr ! 4 >adr ! end off ;
15 ¦ : range ( adr--adr)
16   limit umax memtop umin ;
17 : rdnew ( from to--)                    RDNEW erzeugt eine neue Ramdisk
18   range swap range swap                       und prueft (fast) alles
19   2dup $500 - u> abort" range!"
20   over plen over ! rduse
21   swap - 6 >adr !
22   rddel 0 0 id! ;
23
24                                    ≡                                  ╪
```

30

```
0 \ RD: rdcheck                clv01aug87
1
2 ¦ : ?error IF ." error " THEN ;
3
4 : rdcheck
5  .rd
6  rd BEGIN
7   dup @   dup 0 b/blk 5 + uwithin
8                          not ?error
9   +       dup cr u.
10  dup @   dup 3 u.r space
11  WHILE   dup 2+ @ blk/drv u/mod
12          1 u.r . ." : " 2 u.r
13          dup 4 + &26 type
14          stop? ?exit
15  REPEAT  end -                ?error ;
16
17
18
19
20
21
22
23
24
```

115

```
\\ zu RD: rdcheck            clv01aug87

RDCHECK prueft die Zeiger der Ramdisk
        und gibt eine Inhaltsuebersicht
```

@ =

31

116

@ ⊥

32

117

```
0 \ ST:Supertape LoadScreen    clv01aug87
1
2 (64 .( nix fuer C64!! ) quit C)
3
4 \needs Code .( ??! Code !??) quit
5
6 Assembler
7 \needs rom  .( ??! rom  !??) quit
8 Onlyforth
9
10    1  $12 +thru \ supertape laden
11
12
13 \\ Supertape wird von der Zeitschrift
14    c't fuer alle gaengigen Rechner
15    verbreitet.
16      Wir danken dem Verlag fuer die
17    freundliche Genehmigung, es fuer
18    unsere Zwecke anpassen und
19    weiterverbreiten zu duerfen
20
21
22
23
24
```

```
\\ zu ST:LoadScreen          clv01aug87

Supertape ist ein Schnell-Lader, der
mit 3600 Bd oder 7200 Bd ca. 10 mal
so schnell wie die entsprechende CBM-
Routine ist.
 Bedienung:
  GeraeteNummer =  7 ==> Supertape
  SekundaerAddr>=$80 ==> 7200 Bd
                <$80 ==> 3600 Bd
                ..sonst wie CBM

Aufzeichnung:8 Bit pro Byte, Niederwerti
ge bit zuerst. Selbsttaktierend: An
jeder Bitgrenze ist ein Flankenwechsel.
Wenn in der Mitte dazwischen auch einer
ist ist das Bit=0, sonst=1.

Format: sync #$2a 25b:Header 2b:checksum
        sync #$c5 len:Daten  2b:checksum
Sync  = 64b:#$16
Header=16b:Name
       1b:SekAdd 2b:from 2b:len 4b:#$00
```

≡ ⊥

33 118

```
 0 \ ST:Labels..                clv16jun87  \\ zu ST:Labels..           clv16jun87
 1
 2 \ ------ hardware-Adressen -----------    -------- hardware-Adressen -----------
 3 $0001 >Label pCass                       1  Cassettenport
 4 $ff02 >Label pTimerB                     2  Zeit fuer Timer2
 5 $ff09 >Label pTimerBCtrl                 1  kontrollregister fuer Timer2
 6 $ff3f >Label pRamOn                      1  Schreibzugriff schaltet auf RAM
 7 $ff3e >Label pRamOff                     1  Schreibzugriff schaltet auf ROM
 8
 9 \ ------ System-Vektoren -------------    -------- System-Vektoren -------------
10 $0330 >Label vSave                       2  Save-Vektor des Systems wird verbogen
11 $032e >Label vLoad                       2  Load-Vektor des Systems wird verbogen
12
13 \ --- Eingabe-Parameter Load/Save-----    ----- Eingabe-Parameter Load/Save-----
14 $ae >Label zGeraeteNr                    1  Geraete-Nummer
15 $ad >Label zSekadd                       1  Sekundaeradresse (steuert Geraet)
16 $af >Label zFilenameZ                    2  Zeiger auf Filenamen
17 $ab >Label zFileNameC                    1  Anzahl der Zeichen im Filenamen
18 $b4 >Label zBasLadeAdd                   2  Anfangsadresse fuer LOAD
19 $b2 >Label zIOStartZ                     2  Anfangsadresse fuer SAVE
20 $9d >Label zProgEndeZ                    2  Endadresse+1   fuer SAVE
21
22 \ --- Ausgabe-Parameter von Load/Save --  ----- Ausgabe-Parameter von Load/Save --
23 $90 >Label zStatus                       1  Status Flags des Betr.sys
24                                  @                                   =
```

34 119

```
 0 \ ST:..Labels                clv16jun87  \\ zu ST:..Labels           clv16jun87
 1
 2 \ ------ benutzte System-Routinen -----   -------- benutzte System-Routinen -----
 3 $e38d >Label xCassMotorOn                   Kassetten-Motor einschalten
 4 $e3b0 >Label xCassMotorOff                  Kassetten-Motor einschalten
 5 $e364 >Label xCassPrtOn                     Kassetten-Port init
 6 $e378 >Label xCassPrtOff                    Kassetten-Port init
 7 $f050 >Label xLoad                          normale Load-Routine
 8 $f1a4 >Label xSave                          normale Save-Routine
 9 $f189 >Label xMsgLoadVerify                 gibt 'Loading' oder 'Verifying' aus
10 $e31b >Label xPressplay                     gibt 'Press play..' aus
11 $e319 >Label xPressRec                      gibt 'Press Record..' aus
12 $ebca >Label xFoundFile                     gibt 'Found' aus
13 $f160 >Label xSearching                     gibt 'Searching' aus
14 $ffd2 >Label kOutput                        gibt ein Zeichen aus
15
16 \ ------ benutzte Zeropage-Adressen----   -------- benutzte Zeropage-Adressen----
17
18 $5f >Label zAnfangZ                      2  Adresse des aktuellen I/O  Bytes
19 $61 >Label zEndeZ                        2  Adresse des letzten   i/O  Bytes+1
20 $93 >Label zVerifyFlag                   1  naechsten Block: Verify/-Laden
21 $59 >Label ZBlockArt                     1  naechster Block: Header/Daten
22 $58 >Label zBit                          1  letzter Zustand des Kassettenports
23 $57 >Label zByte                         1  bereits geladener Teil des akt.Bytes
24 $ff >Label zTmp                      @   1  letztes geladenes Byte                 ⊥
```

35 120

```
 0 \ ST:..Labels                clv16jun87  \\ zu ST:..Labels           clv16jun87
 1
 2
 3 $d8 >Label zReservAA                      2  kurzer/langer Impuls bei Save
 4 $5d >Label zCheckSum                      2  Checksumme
 5 $63 >Label zPruefSummeB
 6 $da >Label zTmpSP                         1  Stackpointer fuer Fehleraustritt
 7
 8
 9 \ --- sonstige Systemadressen -------    ----- sonstige Systemadressen -------
10 $07c8          >Label sTime              1  Zeit fuer naechsten TimerBStart
11 $0332  dup     >Label sCassBuffer        c0 Puffer fuer Kassettenoperationen
12 $19 + $100 mod >Label cCassBufferEnd     -  Ende des Puffers, Low-Byte
13
14 \ --------- Konstanten --------------    ----------- Konstanten -------------
15 $07 >Label cGeraetST                        GeraeteNummer von Supertape
16 $2a >Label cHeaderMark                      1.Byte eines Headerblocks
17 $c5 >Label cDatenMark                       1.Byte eines Datenblocks
18 $4f >Label chsl                             Zeit 7600 Baud laden
19 $b5 >Label clsl                             Zeit 3600 Baud laden
20 $78 >Label chssh    $34 >Label chssl        Zeit 7600 Bd saven lang/kurz-Impuls
21 $ff >Label clssh    $78 >Label clssl        Zeit 7600 Bd saven lang/kurz-Impuls
22 $16 >Label cSyncByte                        Byte fuer Synchronisierungs-Vorspann
23 $0b >Label cSyncBytesLoad                   min. Anzahl von SyncBytes beim Laden
24 $40 >Label cSyncByteZahl             ≡      Anzahl von SyncBytes beim Saven      ╪
```

36 121

```
0 \ ST:verschiedenes            clv28jul87  \\ zu ST:verschiedenes        clv28jul87
1
2 Label btlBeg                              Beginn des Bootstraploaders
3 Label puffinit \ LadeZeiger auf Puffer
4  sCassbuffer $100 u/mod
5  # lda    zAnfangZ 1+ sta zEndeZ 1+ sta
6  # lda    zAnfangZ    sta
7  cCassbufferEnd # lda     zEndeZ    sta
8  rts end-code
9
10 Label timerBStart
11  sTime lda          pTimerB      sta      (1)
12  0   #  lda         pTimerB   1+ sta      Startet Timer Nummer 2
13  $10 #  lda         pTimerBCtrl  sta      (1)
14  rts end-code                            mit Zeit in STIME
15
16 Label delayMotor \ Hochlauf-Verzoegerung
17  0 # ldx 0 # ldy
18  [[ [[ dex 0= ?] dey 0= ?]
19  rts end-code                            WarteSchleife
20
21                                          (1) die Sequenz'brk brk bit brk brk'
22                                              stoppt oft das Selbstueberschreiben
23                                              beim Booten, wenn ein Lesefeher
24                                        @     aufgetreten ist                =
```

37 122

```
0 \ ST:stEnde etc.              clv23jul87  \\ zu ST:stEnde etc.          clv18jun87
1
2 Label stEnde        0 # lda  $2c c,                  kein  Fehler (Bit--)
3 Label loadFehler  $1d # lda  $2c c,                  Load-      "  (Bit--)
4 Label eot         $04 # lda  $2c c,        AR := FehlerNr EOT -      "  (Bit--)
5 Label verFehler   $1c # lda  $2c c,                  Verify-    "  (Bit--)
6 Label brkFehler   $1e # lda                          Stop-      "
7  pRamOff sta  pha                         Auf Rom zurueckschalten, Fehler pushen
8  xCassMotorOff jsr  xCassPrtOff jsr       Port exit
9  zTmpSP ldx   pla  txs                    Stack reparieren
10 zAnfangZ ldx  zAnfangZ 1+ ldy            xr-yr := Lade-EndAdresse
11 01 # cmp cli rts end-code                CF    := Fehler , Interrupt wieder an
12
13
14 \\  cbm: stop: ar=0  cf=1
15       normal ar=0  cf=0  st=0
16         eot                $80
17  load/vererr               $10
18    pruefsum                $60
19                            ...
20 kernal-fehler ar=0..8 cf=1
21
22  s.ROM:$a803
23
24                                        @                                  1
```

38 123

```
0 \ ST:bitLesen                 clv18jun87  \\ zu ST:bitLesen             clv16jun87
1
2 Label bitLesen \ akt.Byte in AR zurueck
3  $10 # lda [[ ptimerBctrl bit 0<> ?]      warten bis Timer abgelaufen       (?)
4  pCass lda  $10 # and  zBit cmp           Carry := 1 , wenn Pegel gleich == Bit=1
5  0<> ?[ clc ]?       zBit sta              abspeichern
6  zByte ror zByte lda                       in Byte rotieren
7  0< ?[ zCheckSum wInc ]?                  wenn Bit=1: Checksumme inkrementieren
8  [[ pCass lda $10 # and  zBit cmp 0<> ?]  Auf taktierende Flanke warten
9  zBit sta timerBStart jsr                 Portzustand merken ,Timer neu setzen
10 zByte lda rts end-code                   akt.Byte zurueckgeben
11
12
13
14
15
16
17
18
19
20
21
22
23
24                                        =                                  ╪
```

39 124

```
\ ST:stRead..                clv05aug87  \\ zu ST:stRead..           clv28jul87

Label stRead \ liest einen Block          Daten/oder Header,Verifyfehler := 0
 zBlockArt sta  0 # ldx
Label syncron
 [[ bitLesen jsr cSyncByte # cmp 0= ?]    einsyncronisieren
 cSyncBytesLoad # ldx
 [[ $08 # ldy                             Byte lesen
    [[ bitLesen jsr dey 0= ?]
    cSyncByte # cmp        syncron bne    kein Syncron.Byte? dann neu suchen
    dex 0= ?]
 [[ $08 # ldy                             bis Vorspann sicher erkannt
    [[ bitLesen jsr dey 0= ?]                Byte Lesen
    cSyncByte # cmp 0<> ?]                   bis Vorspann zuende ist AR=Blockart
 zBlockArt cmp 0<>                        gesuchte BlockArt? dann lesen
 ?[ cDatenMark # cmp       syncron beq    Header gesucht Dat. gef.? weitersuchen
 $10 # lda zStatus sta loadFehler jmp ]?  andre Art? Fehler
 0 # lda zCheckSum sta zCheckSum 1+ sta   Checksumme := 0
 $08 # ldy                               Byte Lesen
    [[ bitLesen jsr dey 0= ?] zTmp sta
```

40 125

```
\ ST:..stRead                clv28jul87  \\ zu ST:..stRead           clv28jul87

 [[ [[                                   --- Schleife von Lade-Anfang bis Ende
  zCheckSum    lda  zPruefSummeB    sta   Pruefsumme
  zCheckSum 1+ lda  zPruefSummeB 1+ sta              := Checksumme
  bitLesen jsr      bitLesen jsr         2 Bit Lesen
  zVerifyFlag lda 0=                     nur Verify?
  ?[ zTmp lda    zAnfangZ )Y sta ]?      sonst: Byte laden
  bitLesen jsr      bitLesen jsr         2 Bit lesen
  zTmp lda       zAnfangZ )Y cmp         Byte vergleichen
  0<> ?[ inx ]?                          Verifyfehler hochzaehlen
  bitLesen jsr      bitLesen jsr         2 Bit Lesen
  zAnfangZ wInc                          Zeiger auf naechstes Byte
  bitLesen jsr      bitLesen jsr         2 Bit lesen
  zTmp sta                               neues Byte
 zAnfangZ 1+ lda  zEndeZ 1+ cmp 0= ?]    --- Schleifenende
 zAnfangZ PF├#KCFâ+s#+└CFFF6kâCF┌F÷┌MFFFF --- Schleifenende
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFNFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF@
```

41 126

```
\ ST:..stRead                clv05aug87  \\ zu ST:..stRead           c2v27jul87

 zTmp lda ZPruefSummeB cmp 0<> ?[         Pruefsummen-Fehler?
Label SFehler zStatus lda  $60 # ora       dann Status
       zStatus sta loadFehler jmp ]?       und  LoadFehler-Exit
 $08 # ldy                               Byte lesen
    [[ BitLesen Jsr dey 0= ?]
 zPruefSummeB 1+ cmp        SFehler bne  Pruefsummen-Fehler?
 0 # cpx  0<> ?[ $10 # lda zStatus sta   Verifyfehler?
                      verFehler jmp ]?
Label ldRTS rts end-code
```

42

```
0 \ ST:stLoad..                clv23jul87
1
2 Label stLoad
3  zVerifyFlag sta  0 # lda zStatus sta
4  zGeraeteNr  lda  cGeraetST # cmp 0<>
5  ?[ xLoad jmp ]? \ CBM-Routine
6 Label loadNext
7  tsx  zTmpSP stx
8  xPressplay jsr ldRTS bcs
9  sei     zVerifyFlag lda pha
10 0 # lda zVerifyFlag sta
11 xSearching jsr
12 Label ldWrongFile
13 xCassMotorOn jsr delayMotor jsr
14 xCassPrtOn   jsr puffInit   jsr
15 clsl # lda   sTime sta
16 cHeaderMark # lda  stRead jsr
17 $63 # ldy  xFoundFile jsr
18 0 # ldy [[ sCassBuffer ,Y lda
19    kOutput jsr  iny  $10 # cpy  0= ?]
20 $ff # ldy
21
22
23
24
```

\\ zu ST:stLoad.. clv16jun87

wird Load-Vektor des Systems
Verify/Load merken; Status loeschen
fuer Supertape?
wenn nicht -> CBM-Routine
Stack fuer Fehlerbeh. sichern
'Press play on Tape' Stop?,dann return
Int. aus
auf Laden stellen
'Searching...'

Initialisieren

3600 Baud/Laden
Header suchen und Laden
'Found ..'
FileNamen ausgeben

@ =

43

```
0 \ ST:..stLoad                clv23jul87
1
2
3 Label ldComp
4  [[ iny zFileNameC cpy 0<>
5   ?[[ pRamOn sta zFilenameZ )Y lda
6   pRamOff sta
7   sCassBuffer ,Y cmp         ldComp beq
8   Ascii ?      # cmp         ldComp beq
9   sCassBuffer $10 + lda  $02 # and 0<>
10  ?[ $80 # lda zStatus sta eot jmp ]?
11  xCassPrtOff jsr       ldWrongFile jmp
12 ]]? pla  zVerifyFlag sta
13 xMsgLoadVerify jsr
14 zBasLadeAdd    lda  zAnfangZ    sta
15 zBasLadeAdd 1+ lda  zAnfangZ 1+ sta
16 zSekAdd lda 0<>
17 ?[ sCassBuffer $11 + lda zAnfangZ   sta
18   sCassBuffer $12 + lda zAnfangZ 1+ sta
19 ]?
20 clc sCassBuffer $13 + lda
21 zAnfangZ    adc zEndeZ    sta
22 sCassBuffer $14 +  lda
23 zAnfangZ 1+ adc zEndeZ 1+ sta
24
```

\\ zu ST:..stLoad clv16jun87

alle angegebenen Zeichen vergleichen

Zeichen wie in angeg. Filenamen?
dann weiter
angeg. Zeichen '?' ? dann weiter
End-Of-Tape ?
dann NotFound
sonst: Bildschirm an, weitersuchen
VerifyFlag reparieren
'loading'/'verifying'
LadeAddresse := vom System uebergeben

SekAdd.=1?

dann Ladeaddresse aus Header

LadeEnde
:=
LadeAddresse
+FileLaenge

@ 1

44

```
0 \ ST:..stLoad                clv20nov87
1
2
3  chsl # lda    sTime sta
4  sCassBuffer $10 + lda 0>=
5  ?[ clsl # lda sTime sta ]?
6  pRamOn sta  cDatenMark # lda stRead jsr
7  stEnde jmp end-code
8
9 Label loadsys \ laedt und startet
10 loadnext jsr CS ?[ brk ]?
11 loadnext jsr CS ?[ brk ]?
12 origin 8 - jmp \ Forth-Cold
13 Label btlEnd
14 base @ hex
15 Create g----    7 allot
16 loadsys 0 <# #s Ascii g hold #cr hold #>
17 g---- place
18 base !
19 : >lower ( str--) count bounds
20  DO I c@ $7f and I c! LOOP ;
21
22 g---- >lower    forget >lower
23
24
```

\\ zu ST:..stLoad c2v27jul87

7200 Baud/Laden setzen
mit 3600 Bd gesaved (==Sekadd>$80)?
dann 3600 Bd/Laden setzen
auf Ram schalten, Datenblock laden
Ende

Wird fuer Bootstraplader gebraucht

Erzeugt einen String der Form 'g78b5',
mit der Adresse LOADSYS
der als Monitor-befehl verwendet,
die Startadresse des Bootstraploaders
angibt. s.a. SAVEBOOTER

Dieser String darf keine Grossbuchstaben
enthalten

= ╪

45 130

```
0 \ ST:wByte w4bits            clv16jun87  \\ zu ST:wByte w4bits        clv16jun87
1
2 Label wByte here 3 + Jsr \ Byte schreib.
3 Label w4bits             \ obere 4 Bits
4  $04 # ldy                                  4 Bits
5  [[ zByte lsr CS                            --- Schleife ueber 4 Bits
6     ?[ zReservAA 1+ lda  sTime sta ]?          bit=1?, dann volle  Zeit setzen
7     $10 # lda [[ pTimerBCtrl bit 0<> ?]        Auf Timer warten
8     timerBStart jsr                            neu starten
9     pCass lda  $02 # eor pCass sta             Flanke schreiben
10    CC ?[ $10 # lda                            bit=0?
11       [[ pTimerBCtrl bit 0<> ?]                  auf Timer warten
12       timerBStart jsr                            und neu starten
13       pCass lda  $02 # eor pCass sta             Flanke schreiben (Bit-Grenze)
14    ][ zCheckSum    lda 0 # adc                bit=1?
15       zCheckSum    sta                           Checksumme hochzaehlen
16       zCheckSum 1+ lda 0 # adc
17       zCheckSum 1+ sta
18       zReservAA lda sTime sta ]?                 halbe Zeit setzen
19    dey 0= ?] rts  end-code                    --- Schleife Ende
20
21
22
23
24                                  @                                =
```

46 131

```
0 \ ST:stWrite                 clv18jun87  \\ zu ST:stWrite             clv18jun87
1
2 Label stWrite \ schreibt einen Block      AR=Blockart
3  pha cSyncByteZahl # ldx                  sichern
4  [[ cSyncByte # lda  zByte sta            SynchronisationsBytes
5     wByte Jsr dex 0= ?]                   ..schreiben
6  pla  zbyte sta  wByte Jsr                Blockart schreiben
7  0 # ldy zCheckSum sty zCheckSum 1+ sty   Checksumme := 0
8  [[ [[                                    --- Schleife von 1. bis letztes Byte
9   zAnfangZ )Y lda  zByte sta w4bits jsr                 untere 4 Bits
10  zAnfangZ wInc              w4bits jsr                 obere 4 Bits schr.
11  zAnfangZ    lda   zEndeZ    cmp 0= ?]   --- Schleife..
12  zAnfangZ 1+ lda   zEndeZ 1+ cmp 0= ?]   --- ..Ende
13 zCheckSum lda  zCheckSum 1+ ldx          Checksumme..
14 zByte sta        wByte jsr               ..Low  Byte schreiben
15 txa   zByte sta  wByte jsr               ..High Byte schreiben
16 wByte jmp end-code                       paar Extrabits, damit Laden immer endet
17
18
19
20
21
22
23
24                                  @                                ⊥
```

47 132

```
0 \ ST:saveName                clv26jul87  \\ zu ST:saveName            clv01aug87
1
2 Label saveName \ prueft nix               schreibt FileNamen in KassettenPuffer
3  bl # lda  $0f # ldy                       CassettenPuffer [0..$10]
4  [[ sCassBuffer ,Y sta  dey 0= ?]                  := <blanks>
5  zFileNameC ldy ram                        CassettenPuffer [0..FileNameLaenge]
6  [[ dey 0>= ?[[ zFileNameZ )Y lda                  :=
7     sCassBuffer ,Y sta ]]? rom                        FileName
8 Label rsRTS rts end-code
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24                                  ≡                                ┼
```

48

```
0 \ ST:stSave..                clv16jun87  \\ zu ST:stSave..           clv01aug87
1
2 Label stSave                             wird als Betr.Sys-Vektor eingetragen
3  zGeraeteNr lda cGeraetST # cmp 0<>       GeraeteNr = Supertape?
4  ?[ sec $0e # and 0= ?[ clc ]?           sonst: was das soll weiss ich nicht
5                         xSave jmp ]?        CBM-Save-Routine
6  tsx  zTmpSP stx                         StackPointer fuer Fehlerbehandlung
7  saveName jsr                            FileName in Buffer
8  clc  xPressRec  jsr          rsRTS bcs  ' Press Play & Record on Tape'  STOP?
9  sei  xCassPrtOn jsr xCassMotorOn jsr    Initialisieren
10 delayMotor jsr
11 zSekAdd lda  sCassbuffer &16 + sta
12 zIOStartZ    lda sCassBuffer &17 + sta  Startadresse in Buffer -- aendern???
13 zIOStartZ 1+ lda sCassBuffer &18 + sta                          -- fuer COPY?
14 sec zProgEndeZ lda  zIOStartZ    sbc    FileLaenge
15 sCassBuffer &19 + sta                   ..berechnen
16 zProgEndeZ 1+ lda   zIOStartZ 1+ sbc    ..und
17 sCassBuffer &20 + sta                   ..in Buffer schreiben
18 0 # lda sCassBuffer &21 +               CassBuffer [$21..$24]
19 dup sta 1+ dup sta 1+ dup sta 1+ sta                          := 0
20 pTimerB 1+ sta                          Zeit-HighByte := 0
21 sCassBuffer $100 u/mod                  SaveAnfangsAdresse
22 # lda zAnfangZ 1+ sta  zEndeZ 1+ sta                  := CassettenPuffer
23 # lda zAnfangZ    sta                  SaveEndeAdresse
24 cCassBufferEnd # lda   zEndeZ    sta   @              := Cassett.Puffer-Ende   =
```

49

```
0 \ ST:..stSave                clv16jun87  \\ zu ST:..STSave           clv01aug87
1
2
3  clssh # lda  zReservAA 1+ sta           3600Baud/kurzer SaveImpuls (==Bit=0)
4  clssl # lda  zReservAA    sta                  /langer SaveImpuls (==Bit=1)
5  pTimerB sta                             setzen
6  $10 # lda  pTimerBCtrl sta              TimerNummer2   an
7  cHeaderMark # lda  stWrite jsr          Vorspann(==Buffer) schreiben
8  delayMotor jsr                          Pause schreiben
9  zSekadd bit 0<                          7200Bd gewuenscht (==SekAdd>=$80) ?
10 ?[ chssh # lda  zReservAA 1+ sta        dann 7200Bd/kurzer SaveImpuls
11    chssl # lda  zReservAA    sta                  /langer SaveImpuls
12    pTimerB sta ]?                           setzen
13 zIOStartZ     lda  zAnfangZ    sta       SaveAnfangsAdresse
14 zIOStartZ  1+ lda  zAnfangZ 1+ sta                  := vom System uebergeben
15 zProgEndeZ    lda  zEndeZ      sta       SaveEndeAdresse
16 zProgEndeZ 1+ lda  zEndeZ   1+ sta                  := vom System uebergeben
17 pRamOn sta cDatenMark # lda stWrite jsr  Ram ein , DatenBlock schreiben
18 delayMotor jsr stEnde jmp end-code       Pause schreiben fertig
19
20
21
22
23
24                                  @                                    1
```

50

```
0 \ ST:supertape savebooter    clv10oct87  \\ zu ST:supertape stSavSys  clv10oct87
1
2 : supertape  \ --                        SUPERTAPE
3  7 device !                              .. setzt aktuelles Geraet
4  stLoad vLoad !  stSave vSave !          .. aendert Betriebssystem Vektoren
5  ." ST2.20 " ;                           .. und gibt Meldung aus
6
7 ¦ : (n" >in store n" ;                   !! Ein Supertape-System muss in 3 Teilen
8                                          !! gesichert werden:
9 : btl ( --[from ]to )                    !!  1. Mini-Supertape
10  [ BtlBeg ] Literal [ BtlEnd ] Literal ; !!  2. Teil des Systems davor
11                                         !!  3. Teil des Systems dahinter
12 ¦ : btlName ( --adr count)              !! Der 1.Teil wird im CBM-Format gesich.
13  pad $16 bl fill                        !! und laedt dann 2.&3. im ST-Format.
14  (n" $10 umin pad     swap cmove
15  g---- count pad $a + swap cmove          An den eingelesenen Filenamen
16  pad $10 ;                                das gLOADSYS anhaengen.
17
18 : stSavSys ( --)  \ Name" folgt
19   device store 1 device !                 1. von PUFFINIT bis exkl. BTL
20   btl btlName              bsave          im CBM-Format sichern
21   7 device !                              ST-Format nehmen
22   origin $17 - btl drop (n" bsave         2. sichern
23   btl nip here          n" bsave ;        3.
24                                  ≡                                    ╪
```

51 136

```
0 \ Loadscreen for Decompiler    20oct87re
1 \ nach F83 by Henry Laxen / Mike Perry
2
3 \needs Tools  Vocabulary Tools
4
5 .( Decompiler loading...)
6
7 Onlyforth
8 Tools also definitions
9
10 \needs dis     ' drop ¦ Alias dis
11          \ Disassemble if possible
12
13 &1 &9 +thru
14
15 \\
16
17 clear
18
19
20
21
22
23
24
```

52 137

```
0 \ case defining words        20aug85maWe
1
2 ¦ : case:  ( n -)
3  Create , 0 ]
4  Does> 2+ swap 2* + perform ;
5
6 ¦ : associative:
7  Create ,    ( n -)
8  Does>       ( n - index)
9  dup @ -rot dup @ 0
10    DO 2+ 2dup @ =
11      IF 2drop drop I 0 0 LEAVE THEN
12    LOOP 2drop ;
13
14
15 Defer (see
16 ¦ Variable maxbranch
17 ¦ Variable thenbranch
18
19
20
21
22
23
24
```

53 138

```
0 \ decompile each type of word 29nov85re
1
2 ¦ : .word   ( IP - IP')
3  dup @ >name .name 2+ ;
4
5 ¦ : .lit    ( IP - IP')
6  .word dup @ . 2+ ;
7
8 ¦ : .clit   ( IP - IP')
9  .word dup c@ . 1+ ;
10
11 ¦ : .string ( IP - IP')
12  cr .word count 2dup type ascii " emit
13  space + ;
14
15 ¦ : .do  ( IP - IP')   ." DO " 4 + ;
16
17 ¦ : .loop  ( IP - IP')   ." LOOP " 4 + ;
18
19 ¦ : .exit    ( IP - IP' f)
20  dup maxbranch @ u< IF  .word exit  THEN
21  dup @ [ Forth ] ['] unnest =
22  IF  drop ." ;"  ELSE  .word ." ; -2 allot "
23  THEN  0= ;
24
```

```
\ Sieve benchmark              20oct87re

Onlyforth

: allot  ( u --)
 dup sp@ here - $180 -  u>
 abort" no room" allot ;

&8192 Constant size
Create flags  size allot
: do-prime  ( -- #primes )
 flags size 1 fill    0
 size    0 DO flags I + c@
           IF  I 2* 3+ dup I +
             BEGIN  dup size <
             WHILE  0 over flags + c!
                    over +
             REPEAT  2drop 1+
           THEN
         LOOP ;
: benchmark  9 0 DO do-prime drop  LOOP
 do-prime . ." Primzahlen" ;
: .primes   size 0 DO  flags I + c@
 IF  I 2* 3+ .  THEN ?cr
 stop? IF  LEAVE  THEN  LOOP ;
```

54 139

```
0 \ branch, ?branch            29nov85re
1
2 ¦ : .to
3  ." back to " .word drop ;
4
5 ¦ : .branch ( IP - IP')
6  2+ dup @ 2dup +  swap 0<
7  IF   cr ." REPEAT to " .exit
8    0<> swap 2+  and  exit
9  THEN cr ." ELSE " dup thenbranch !
10 dup maxbranch @ u>
11 IF  maxbranch !  ELSE  drop  THEN 2+ ;
12
13 ¦ : .?branch  ( IP - IP')
14 2+ dup @ 2dup +
15 swap 0<
16 IF cr ." UNTIL " .to 2+ exit THEN
17 cr dup 4 - @ [ ' branch ] literal =
18 over 2-  @ 0< and
19 IF   ." WHILE "
20 ELSE ." IF "   dup thenbranch !
21 THEN  dup maxbranch @ u>
22    IF maxbranch ! ELSE drop THEN  2+ ;
23
24                                  @                    =
```

55 140

```
0 \ decompile does> ;code ;      20oct87re
1
2 ¦ : does?    ( IP - IP' f)
3  dup 3 + swap
4  dup c@ $4C  =  swap      \ jmp-opcode
5  1+ @  [ ] Forth @ 1+ @ = \ (dodoes>
6  and ;
7
8 ¦ : .(;code  ( IP - IP' f)
9   2+ does?
10   IF cr ." Does> "
11   ELSE  ." ;Code " 3 - dis 0 THEN ;
12
13 ¦ : .compile  ( IP -- IP' )
14   .word .word ;
15
16
17
18
19
20
21
22
23
24                                  @                    1
```

56 141

```
0 \ classify each word          20oct87re
1
2 &18 associative: execution-class
3 Forth
4 ' lit ,       ' clit ,      ' ?branch ,
5 ' branch ,    ' (DO ,       ' (.' ,
6 ' (abort" ,   ' Does> 4 + @ ,  \ (;code
7 ' exit ,      ' abort ,     ' quit ,
8 ' 'quit ,     ' (quit ,     ' unnest ,
9 ' (" ,        ' (?DO ,      ' (LOOP ,
10 ' compile ,
11
12 &19 case: .execution-class
13 .lit         .clit         .?branch
14 .branch      .do           .string
15 .string      .(;code
16 .exit        .exit         .exit
17 .exit        .exit         .exit
18 .string      .do           .loop
19 .compile     .word    ;
20
21
22
23
24                                  =                    ╪
```

57 142

```
0 \ decompile a :-definition   20aug85mawe
1
2 : .pfa  ( cfa -)
3  >body
4  BEGIN ?cr dup
5    dup thenbranch @ =
6    IF ." THEN " ?cr THEN
7    @ execution-class .execution-class
8    dup 0= stop? or UNTIL
9  drop ;
10
11 : .immediate  ( cfa -)
12  >name c@ dup
13  ?cr $40 and IF ." Immediate " THEN
14  ?cr $80 and IF ." restrict" THEN ;
15
16 : .constant   ( cfa -)
17  dup >body @ . ." Constant "
18  >name .name ;
19
20 : .variable   ( cfa -)
21  dup >body . ." Variable "
22  dup >name .name
23  cr ." Value = "  >body @ . ;
24
```

58 143

```
0 \ display category of word     20oct87re
1
2 : .:       ( cfa -)
3  ." : " dup >name .name cr .pfa ;
4
5 : .does>  ( cfa -)
6  cr ." Does> " 2- .pfa ;
7
8 : .user-variable  ( cfa -)
9  dup >body c@ . ." User-Variable "
10 dup >name .name
11 cr ." Value = "  execute @ . ;
12
13 : .defer  ( cfa -)
14  ." deferred " dup >name .name
15  ." Is " >body @ (see ;
16
17 : .other  ( cfa -)
18  dup >name .name
19  dup @ over >body =
20   IF ." is Code" @ dis exit THEN
21  dup @ does? IF .does> drop exit THEN
22  drop ." maybe Code" @ dis ;
23
24
```

59 144

```
0 \ Classify a word              22jul85we
1
2 5 associative: definition-class
3 ' quit @ ,          ' 0 @ ,
4 ' scr @ ,           ' base @ ,
5 ' 'cold @ ,
6
7 6 case: .definition-class
8 .:                  .constant
9 .variable           .user-variable
10 .defer             .other  ;
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

```
\ Graphic-Demos Loadscreen     20oct87re

Only Forth also definitions

\needs Graphic -&80 +load

Graphic also definitions
 page  .( Loading .....)

 1   4 +thru   \ Demo1,2,3,4 Demo
 5     +load   \ Sinplot
 6 &11 +thru   \ Turtle demos


wellen wellen1 dreieck linien moire
sinplot
ornament circles schnecke spirale
siedlung

&20 window
```

60

```
0 \ Top level of Decompiler    20aug85mawe
1
2 : ((see    ( cfa -)
3  maxbranch off  thenbranch off
4  cr dup dup @
5  definition-class .definition-class
6  .immediate ;
7
8 ' ((see Is (see
9
10 Forth definitions
11
12 : see    ' (see ;
13
14
15
16
17
18
19
20
21
22
23
24
```

145

```
\ Plot wellen                  20oct87re

&100 | Constant &100
&160 | Constant &160
: wellen
 cs red cyn colors hires
 &100 0 DO
 &99 0 DO
   I dup * J dup * + &150 / 1 and
    IF &160 J + &100 I + plot
       &160 J - &100 I + plot
       &160 J - &100 I - plot
       &160 J + &100 I - plot THEN
 LOOP LOOP ;

: wellen1
 cs blu yel colors hires
 &160 0 DO
 &99 0 DO
   I dup * J dup * + 100u/ 1 and 0=
    IF &160 J + &100 I + plot
       &160 J - &100 I + plot
       &160 J - &100 I - plot
       &160 J + &100 I - plot THEN
  LOOP LOOP ;
```

61

```
0 \ Commodore hole Screens     20oct87re
1
2 Onlyforth
3
4 : <init   0 $DD03 c! ;
5
6 : get  ( -- 8b)
7  BEGIN  $DD0D c@ $10 and  UNTIL
8  $DD01 c@ dup $DD01 c! ;
9
10 : <sync  ( --)
11  <init  BEGIN  get $55 =  UNTIL
12  BEGIN  get dup $55 =
13  WHILE  drop  REPEAT  abort" SyncErr" ;
14
15 : sum  ( oldsum n -- newsum n)
16  swap over + swap ;
17
18 : check  ( sum.int  8b.sum.read --)
19  swap $FF and - abort" ChSumError" ;
20
21 -->
22
23
24
```

146

```
\ lineplot dreieck             20oct87re

: grinit
 clrscreen
 yel blu colors hires ;

: dreieck
 grinit
 0 2 DO
   &160 0 DO
      I &199 &160 I 2/     J + flipline
   &320 I - &199 &160 I 2/ J + flipline
   2 +LOOP
  -1 +LOOP text ;
```

62

```
0 \ Commodore hole Screens     20oct87re
1
2 : download  ( n --)
3  <sync  0 swap buffer b/blk bounds
4  DO  get  sum  I c!  LOOP
5  get check update ;
6
7 : downthru  ( start count --)
8  bounds DO  I download  LOOP ;
9
10 -->
11
12 \\ sync needs: xx $55 $55 00 data
13
14
15
16
17
18
19
20
21
22
23
24
```

147

```
\ lineplot linien moire        20oct87re

: linien
 grinit
 &320 0 DO
    &320 0 DO I &198 J 0 line &35 +LOOP
 &35 +LOOP ;

: moire
 clrscreen ora red colors hires
 &320 0 DO
  I &198 &319 I - 0 line
 3 +LOOP
 &199 0 DO
  &319 &198 I - 0 I line
 2 +LOOP ;
```

63

```
0 \ Commodore sendscreens        20oct87re
1
2 : >init  $FF $DD03 c! ;
3
4 : put ( 8b -)
5  $DD01 c!  BEGIN  stop?
6   IF  <init true abort" terminated" THEN
7   $DD0D c@ $10 and  UNTIL ;
8
9 : >sync  ( --)
10  >init $10 0 DO $55 put  LOOP  0 put ;
11
12 : upload  ( n --)
13  >sync  0 swap  block b/blk bounds
14  DO  I c@ sum  put  LOOP
15  $FF and  put  <init  ;
16
17 : upthru  ( from to -- )
18  1+ swap DO  I . cr  I upload  LOOP ;
19
20
21
22
23
24
```

148

```
\ lineplot boxes                 20oct87re

Variable x0       Variable y0
Variable x1       Variable y1

: box   ( x1 y1 x0 y0 -)
 y0 ! x0 ! y1 ! x1 !
 x1 @ y0 @ x0 @ over flipline
 x1 @ y1 @ over y0 @ flipline
 x0 @ y1 @ x1 @ over flipline
 x0 @ y0 @ over y1 @ flipline ;

Create colortab
 blk c, 1bl c, red c, lre c,
 pur c, grn c, blu c,

: boxes
 grinit
 &10 3 DO
  &160 0 DO I dup &318 I - &198 I - box
        J +LOOP
   I 3 - colortab + c@  pencolor
   LOOP ;
```

64

```
0 \ Graphic Load-Screen          20oct87re
1
2 (16 .( C64 Only ) \\ C)
3
4 Onlyforth
5
6 \needs Code          .( Assembler??!) \\
7
8 \needs 1byte          1 +load
9
10 \needs 100u/        &26 +load
11
12 Vocabulary graphic
13
14 ' graphic ¦ Alias Graphics
15
16 Graphics also definitions
17
18   2 &15 +thru  \ hires graphic
19 &16 &20 +thru  \ sprites
20 &21 &23 +thru  \ turtle graphic
21
22 Onlyforth
23
24
```

149

```
\ Graphic sinplot                20oct87re

&10000 Constant 10k

: sinplot
 grinit
 &319  &96   0 &96 line
 &160 &197 &160  0 line
 &152 &160 negate DO
  I &160 + &96 I sin &96 10k */ +
  I &168 + &96 I 8 + sin &96 10k */ +
  line
 8 +LOOP
 &152 &160 negate DO
  I &160 + &96 I cos &96 10k */ +
  I &168 + &96 I 8 + cos &96 10k */ +
  line
 8 +LOOP  ;
```

65

```
0 \ >byte hbyte 1byte            20oct87re
1
2 Code >byte   ( 16b - 8bl 8bh)
3  SP )Y lda pha txa SP )Y sta SP 2dec
4  txa SP )Y sta pla Puta jmp   end-code
5
6 : hbyte >byte nip ;
7 : 1byte >byte drop ;
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

150

```
\ Turtle demos                   20oct87re

¦ : tinit    ( -- )
 clrscreen  hires   \ showturtle
 red cyn colors ;

¦ : shome  ( -- )
 tinit &65 0 setxy &90 seth pendown ;

: vieleck   ( length edges -- )
 &360 over /
 swap 0 DO  over forward
            dup right     LOOP 2drop ;

¦ : ring  ( edges -- )
 &200 over / swap
 &18 0 DO  2dup vieleck
          &20 right  LOOP  2drop ;

: ornament  ( -- )
 tinit home
 &10 3 DO  clrscreen  I dup 7 -
          IF  ring  ELSE  drop  THEN
          LOOP ;
```

66

```
0 \ Gr Constanten              20oct87re
1
2 ¦ $0288 Constant scrpage
3 ¦ $E000 Constant bitmap
4 ¦ $D800 Constant charset
5 ¦ $C400 Constant colram
6 ¦ $C000 Constant vidram
7 \ $C800 Constant sprbuf
8
9   bitmap hbyte $40 /mod  3 swap -
10 ¦ Constant bank
11   $20 / 8 *
12   colram  hbyte $3F and 4 / $10 * or
13 ¦ Constant bmoffs
14   vidram  hbyte $3F and 4 / $10 *
15   charset hbyte $3F and 4 / or
16 ¦ Constant tmoffs
17
18 $0314 ¦ Constant irqvec
19 $EA31 >label oldirq
20 $EA81 >label irqend
21 $FF6E >label settimer
22
23
24                                    @
```

151

```
0 \ Turtle demos 1             20oct87re
1
2
3 : circles  ( -- )
4  tinit
5   2 -1 DO home
6    &10 0 DO
7     &20 I 2* - &20 vieleck
8     xcor &10 I 2/  - - setx
9     ycor &10 I - J * - sety
10   LOOP
11  2 +LOOP ;
12
13
14
15
16
17
18
19
20
21
22
23
24                                    =
```

67

```
0 \ Gr movecharset clrscreen   20oct87re
1
2 ¦ Code movecharset
3  sei  $32 # lda  1 sta   dey  8 # ldx
4  N sty  N 2+ sty  $D8 # lda  N 1+ sta
5  charset hbyte # lda  N 3 + sta
6   [[
7    [[  N )Y lda  N 2+ )Y sta  iny  0= ?]
8   N 1+ inc  N 3 + inc  dex  0= ?]
9  $36 # lda   1 sta  cli  iny
10 Next jmp  end-code
11
12
13 : clrscreen ( -- ) bitmap &8000 erase ;
14
15
16
17
18
19
20
21
22
23
24                                    @
```

152

```
0 \ Turtle demos 2             20oct87re
1
2 ¦ : (schnecke  ( length -- )  recursive
3  dup 5 < IF  drop exit  THEN
4  dup forward &90 right
5  2- (schnecke ;
6
7 : schnecke  ( -- )
8  shome &190 (schnecke ;
9
10 ¦ : (spirale  ( length -- )  recursive
11  dup 5 < IF drop exit THEN
12  dup forward &91 right
13  2- (spirale ;
14
15 : spirale  ( --)
16  shome 5 forward &190 (spirale ;
17
18
19
20
21
22
23
24                                    @
```

68

```
0 \ Gr Variablen (text (hires  20oct87re
1
2 ¦ Variable cbmkey
3 ¦ Variable switchflag
4 ¦ Variable textborder
5 ¦ Variable hiresborder
6 ¦ Variable switchline
7 ¦ Variable chflag
8
9 Label (text
10  $1B # lda  $D011 sta
11  tmoffs # lda  $D018 sta
12  textborder lda  $D020 sta  rts
13
14 Label (hires
15  $3B # lda  $D011 sta
16  bmoffs # lda  $D018 sta
17  hiresborder lda  $D020 sta  rts
18
19
20
21
22
23
24                                    ≡
```

153

```
0 \ Turtle demos 3             20oct87re
1
2 ¦ : startpos
3  &15 0 setxy &90 setheading ;
4
5 ¦ : weiter  ( -- )
6  &90 right penup &55 forward
7  pendown &90 left ;
8
9 ¦ : kamin
10  xcor ycor
11  &50 fd &30 rt &15 fd &30 lt
12  &30 fd &90 rt &12 fd &90 rt  8 fd
13  setxy &90 setheading ;
14
15 ¦ : haus
16  &50 4 vieleck &50 forward  &30 right
17  &50 3 vieleck &30 left  &50 back
18  &90 right &15 forward &90 left
19  &20 4 vieleck
20  &90 left &15 forward &90 right
21  kamin ;
22
23
24                                    ≠
```

69

```
0 \ Gr rasterirq graphicirq      20oct87re
1
2 Label windowhome
3  switchline lda sec $30 # sbc
4  .A lsr .A lsr .A lsr  sec 1 # sbc
5   $D6 cmp CC ?[ rts ]?
6   tax inx 2 # ldy $CC sty $CD sty
7   $CE lda $D3 ldy $D1 )Y sta
8   0 # ldy $CF sty clc $FFF0 jsr
9   0 # ldy $D1 )y lda $CE sta $CC sty
10  rts
11
12 Label graphicirq
13 $28D lda  2 # and 0= ?[ oldirq jmp ]?
14 [[ $FF9F jsr $28D lda 0= ?] cbmkey ) jmp
15
16 Label rasterirq
17  $D019 lda $D019 sta
18  $15 # ldx [[ dex 0= ?] N lda ( Blind!!)
19  chflag lda  1 # eor chflag sta tax
20  0= ?[ (hires jsr ][ (text jsr ]?
21  switchline ,x lda $D012 sta
22  windowhome jsr
23  $DC0D lda  1 # and graphicirq bne
24  irqend jmp                          @
```

154

```
\ Turtle demos 4              20oct87re

: hausreihe
 tinit startpos
 4 0 DO haus weiter LOOP haus ;

: fenster
 xcor ycor
 penup &30 fd &90 rt &10 fd &90 lt
 pendown
 &10 4 vieleck &90 rt
 penup &20 fd &90 lt
 pendown &10 4 vieleck
 setxy ;

: siedlung   hausreihe
 startpos 4 0 DO fenster weiter LOOP
 fenster ;

                                     =
```

70

```
0 \ Gr IRQ-Behandlung (window    20oct87re
1
2 Label setirq
3  sei graphicirq >byte
4  # lda irqvec 1+ sta    # lda irqvec sta
5  $F0 # lda $D01A sta $81 # lda $DC0D sta
6  cli rts
7
8 ¦ Code resetirq
9  sei oldirq >byte
10 # lda irqvec 1+ sta    # lda irqvec sta
11 $F0 # lda $D01A sta $81 # lda $DC0D sta
12 cli Next jmp end-code
13
14 Label (window
15  rasterirq >byte
16  # lda irqvec 1+ sta  # lda irqvec sta
17  $7F # lda $DC0D sta $F1 # lda $D01A sta
18  switchflag stx chflag stx
19  windowhome jmp
20
21
22
23
24                                  @
```

155

```
\ Turtle demos 5              20oct87re

: (orden   ( len grad -- ) recursive
 stop? 0= and ?dup
 IF  over 3 / swap 1-
  4 0 DO 2dup (orden 2 pick forward
         &90 right  LOOP 2drop
  THEN drop ;

: orden
 tinit shome &192 5 (orden ;

\\

: (6orden   ( len grad -)  recursive
 ?dup IF  over 3 / swap 1-
  6 0 DO 2dup (6orden 2 pick forward
         &60 right  LOOP 2drop
  THEN drop ;

: 6orden
 tinit shome &80 &55 setxy
 &85 3 (6orden ;

                                     1
```

71

```
0 \ Gr text hires window switch  20oct87re
1
2 Code text    1 # lda switchflag sta
3  setirq jsr  (text jsr  Next jmp
4  end-code
5
6 Code hires   2 # lda switchflag sta
7  setirq jsr  (hires jsr  Next jmp
8  end-code
9
10 ¦ Code setwindow  ( row -)
11  sei (window jsr cli xyNext jmp
12  end-code
13
14 : window   ( row -)
15  8 * $30 + switchline c! setwindow ;
16
17 Label switch    switch cbmkey !
18  switchflag ldx
19         0= ?[ inx switchflag stx
20      setirq jsr (text  jsr oldirq jmp ]?
21  1 # cpx 0= ?[ inx switchflag stx
22      setirq jsr (hires jsr oldirq jmp ]?
23  0 # ldx  switchflag stx
24  (window jsr  oldirq jmp  end-code     ≡
```

156

```
                                     ┼
```

72 157

```
0 \ Gr graphic forth            20oct87re
1
2 Forth definitions
3
4 : graphic
5  Graphics movecharset
6  $DD00 c@  $FC and  bank or  $DD00 c!
7  vidram hbyte scrpage c!
8  colram c@ hiresborder c!
9  $D020 c@ textborder c!
10 $10D0 switchline !
11 text ;
12
13 : nographic
14 Onlyforth resetirq
15 $1B $D011 c!  $17 $D018 c! 4 scrpage c!
16 textborder c@   $D020 c!
17 $DD00 c@  3 or  $DD00 c! ;
18
19 Graphics definitions
20
21
22
23
24                                  @                                   =
```

73 158

```
0 \ Gr Colors                   20oct87re
1
2 0  Constant blk      1 Constant wht
3 2  Constant red      3 Constant cyn
4 4  Constant pur      5 Constant grn
5 6  Constant blu      7 Constant yel
6 8  Constant ora      9 Constant brn
7 $A Constant lre     $B Constant gr1
8 $C Constant gr2     $D Constant lgr
9 $E Constant lbl     $F Constant gr3
10
11 : border      ( color -)
12  dup textborder c! $D020 c! ;
13
14 : screen      ( color -)   $D021 c! ;
15
16 : colors      ( bkgrnd  foregrnd -)
17  over hiresborder c!
18  $10 *  or  colram $03F8  rot fill ;
19
20 : background  ( color -)
21  colram c@ $10 /  colors ;
22
23 : pencolor    ( color -)
24  colram c@ $F and swap  colors ;       @                                   1
```

74 159

```
0 \ Gr Bittab Labels            20oct87re
1
2 Label bittab
3  $80 c, $40 c, $20 c, $10 c,
4  $08 c, $04 c, $02 c, $01 c,
5
6 ¦ : >laballot  ( adr n - adr+n)
7  over >label + ;
8
9 $60 Constant pointy  $62 Constant pointx
10
11 Assembler
12
13 N
14  2 >laballot y0      2 >laballot x0
15  2 >laballot y1      2 >laballot x1
16  2 >laballot offset  2 >laballot dy
17  2 >laballot dx      2 >laballot ct
18  1 >laballot iy      1 >laballot ix
19  1 >laballot ay      1 >laballot ax
20  2 >laballot bytnr
21 drop
22
23
24                                  =                                   ╪
```

75

```
0 \ Gr (plot compute            20oct87re
1
2 Label (plot  ( x y -)
3  2 # lda  setup jsr      3 # ldx
4  [[ y0 ,X lda  pointy ,X sta  dex  0< ?]
5  $C7 # lda sec y0 sbc y0 sta
6 Label compute  sei  1 dec
7  y0 lda  $F8 # and  pha
8  bytnr sta  0 # lda  bytnr 1+ sta  clc
9  bytnr asl  bytnr 1+ rol
10 bytnr asl  bytnr 1+ rol
11 pla  bytnr adc  bytnr sta
12   CS ?[  bytnr 1+ inc  ]?
13 bytnr asl  bytnr 1+ rol
14 bytnr asl  bytnr 1+ rol
15 bytnr asl  bytnr 1+ rol
16 y0 lda  7 # and  bytnr ora  bytnr sta
17
18 clc  x0 lda  $F8 # and  bytnr adc
19 bytnr sta
20   x0 1+ lda  bytnr 1+ adc  bytnr 1+ sta
21 bitmap hbyte # lda
22 bytnr 1+ ora  bytnr 1+ sta
23 x0 lda  7 # and  tax  bittab ,X lda
24 0 # ldy  clc rts                    @
```

160

```
\ Sprite-Demo                23oct87re

\needs graphic   -&96 +load

Onlyforth graphic also  Forth

.( Loading...)

   1   4 +thru
```

76

```
0 \ Gr plot flip clpx           20oct87re
1
2 Code plot  ( x y -)
3  (plot jsr
4  bytnr 1+ ldx bitmap hbyte # cpx
5  cs ?[ bytnr )Y ora  bytnr )Y sta ]?
6 Label  romon
7  1 inc  cli  xyNext jmp  end-code
8
9 Code flip  ( x y -)
10 (plot jsr
11 bytnr 1+ ldx bitmap hbyte # cpx
12 cs ?[ bytnr )Y eor  bytnr )Y sta ]?
13 romon jmp  end-code
14
15 Code unplot ( x y -)
16  (plot jsr
17  bytnr 1+ ldx bitmap hbyte # cpx cs ?[
18  $FF # eor  bytnr )Y and bytnr )Y sta ]?
19  romon jmp  end-code
20
21 \\ compute schaltet Interrupt aus, die
22 Worte plot, flip, unplot und line schal-
23 ten ihn wieder ein. Unschoen, aber nicht
24 vermeidbar wegen branch in 'line'.    @
```

161

```
\ Sprite-Demo                20oct87re

Create Shapes  5 $40 * allot
 blk @ 4 +  block
 Shapes  5 $40 * cmove

: init  ( -)
 graphic page
 blu border  blu background
 5 0 DO
     Shapes  I $40 * +  I getform LOOP
 grn wht sprcolors
 5 0 DO  I 0 0 wht I setsprite  LOOP
 5 0 DO  I small  I high  I 3colored set
         I behind LOOP  ;
```

77

```
0 \ Gr line 1                   20oct87re
1
2 Code line  ( x1 y1 x0 y0 -)
3  4 # lda  setup jsr
4 Label (drawto
5  3 # ldx
6  [[ y0 ,X lda  pointy ,X sta  dex  0< ?]
7  $C7 # lda sec y1 sbc y1 sta
8  $C7 # lda sec y0 sbc y0 sta
9
10 ix sty  iy sty  ct sty  dey
11 ax sty  ay sty  ct 1+ sty  dey
12 x1 lda  x0 cmp  x1 1+ lda  x0 1+ sbc
13 CC ?[  sec  x0 lda  x1 sbc  dx sta
14        x0 1+ lda  x1 1+ sbc  dx 1+ sta
15        ix sty
16    ][  x1 lda  x0 sbc  dx sta
17        x1 1+ lda  x0 1+ sbc  dx 1+ sta
18    ]?  y1 lda  y0 cmp
19 CC ?[  sec  y0 lda  y1 sbc  dy sta
20        iy sty
21    ][              y0 sbc  dy sta
22    ]?  dx 1+ lda
23
24                                    =
```

162

```
\ Sprite-Demo                20oct87re

: ypos  ( spr# - y)  sprpos drop ;

: xpos  ( spr# - x)  sprpos nip ;

&26 Constant Distance

: 1+0-1  ( n - +1/0/-1)
 dup 0= not swap 0< 2* 1+ and ;

: follow-sprite ( spr# -)
>r r@ xpos  r@ 1- xpos  Distance +
   over -  1+0-1 + &344 min  r@ xmove
   pause
   r@ ypos  r@ 1- ypos
   over -  1+0-1 +          r> ymove
   pause ;
```

78

```
0 \ Gr line 2                    20oct87re
1
2  0= ?[  dx lda  dy cmp
3   CC ?[  dy ldx  dy sta  dx stx
4          ix lda  ay sta  iy lda  ax sta
5          iny  ix sty  iy sty  ]?  ]?
6  dx 1+ lda  .A lsr  offset 1+ sta
7  dx lda     .A ror  offset sta
8 sec  CC ?[  .( Trick!! )
9   [[ ix lda
10     0<> ?[ 0>= ?[  x0 winc
11                 ][  x0 wdec  ]?  ]?
12     clc  y0 lda  ax adc  y0 sta
13     clc  offset lda  dy adc  offset sta
14     CS ?[  offset 1+ inc  ]?   ct winc
15     dx lda  offset cmp  dx 1+ lda
16     offset 1+ sbc
17     CC ?[  sec  offset lda  dx sbc
18            offset sta  offset 1+ lda
19            dx 1+ sbc  offset 1+ sta
20            ay lda
21            0<> ?[  0>= ?[  x0 winc
22                        ][  x0 wdec ]? ]?
23            clc  y0 lda  iy adc  y0 sta
24     ]?                                  @
```

163

```
\ Sprite-Demo                    20oct87re

: follow-cursor  ( spr# -)
>r r@ xpos  Col 8 * &33 +
   over -  1+0-1 +  r@ xmove  pause
   r@ ypos  Row 8 * &59 +
   over -  1+0-1 +  r> ymove  pause ;

: follow  ( spr# -)
pause  dup IF   follow-sprite
           ELSE follow-cursor THEN ;

: killsprites  ( -)  0 sprite c! ;

: slide-sprites  ( -)
5 0 DO  I follow  I 1+ 0 DO  I follow
LOOP LOOP ;

\\

: testslide ( -)
init BEGIN  slide-sprites
            key dup con!  3 = UNTIL ;
                                        =
```

79

```
0 \ Gr line 3 flipline           20oct87re
1
2  swap  ]?  .( des Trickes 2. Teil )
3  compute jsr
4  bytnr 1+ ldx bitmap hbyte # cpx cs ?[
5 Label mode
6  bytnr )Y ora  bytnr )Y sta ]?
7  1 inc  cli
8     dx lda  ct cmp
9     dx 1+ lda  ct 1+ sbc  CC ?]
10  xyNext jmp  end-code
11
12 Code drawto  ( x1 y1 -)
13  3 # ldy
14  [[ pointy ,Y lda  y1 ,Y sta  dey  0< ?]
15  2 # lda setup jsr  (drawto jmp
16 end-code
17
18 : flipline   ( x1 y1 x0 y0 -)
19  $51 ( eor )  mode c! line
20  $11 ( ora )  mode c! ;
21
22 \ bad self-modifying code
23
24                                      @                                   1
```

164

```
\ Sprite-Demo                    20oct87re

\needs tasks   .( Tasker? ) \\

$100 $100 Task Demo

: slide  ( -)
 Demo activate
 init BEGIN  slide-sprites  REPEAT ;

: endslide  ( -)
 killsprites  Demo activate  stop ;
```

80

```
0 \ Spr Constanten               20oct87re
1
2   $C800 Constant sprbuf
3 ¦ $D000 Constant sprbase
4 ¦ $D010 Constant xposhi
5   $D015 Constant sprite
6 ¦ $D017 Constant yexpand
7   $D01C Constant 3colored
8 ¦ $D01D Constant xexpand
9 ¦ $D025 Constant sprmcol
10 ¦ $D027 Constant sprcol
11
12 ¦ Create sbittab
13  $01 c, $02 c, $04 c, $08 c,
14  $10 c, $20 c, $40 c, $80 c,
15
16
17
18
19
20
21
22
23
24                                      ≡                                   ╪
```

81

```
0 \ Spr setbit set formsprite    20oct87re
1
2 ¦ Code setbit   ( bitnr adr fl  -)
3  3 # lda  setup jsr  dey
4  N 4 + ldx  sbittab ,X lda
5  N ldx
6  0= ?[ $FF # eor  N 2+ )Y and
7     ][  N 2+ )Y ora  ]?
8  N 2+ )Y sta  xyNext jmp  end-code
9
10 : set   ( bitnr adr  -)   True  setbit ;
11
12 : reset ( bitnr adr  -)   False setbit ;
13
14 : getform  ( adr mem#  -)
15  $40 * sprbuf + $40 cmove ;
16
17 ¦ : sprite!   ( mem# spr# adr -)
18  $3F8 + + c! ;
19
20 : formsprite  ( mem# spr# -)
21  >r sprbuf $3F00 and $40 / + dup
22  r@ vidram sprite!  r> colram sprite! ;
23
24
```

166

```
\ tiny sprite editor          06nov87re

Onlyforth Graphic also definitions

\needs sprbuf  Create sprbuf $100 allot
\needs >byte : >byte $100 /mod ;

¦ Variable cbase  2 cbase !

¦ : width ( -- n )  &16 cbase @ / ;

¦ : (l: ( -- )
 base push  cbase @ base !
 name number  name number drop
 >r  >byte drop  r@ c!
 >byte r@ 1+ c!  r> 2+ c! ;

: l: (l: quit ;

: #.r  ( n width -- )
 >r 0 <#  r> 0 DO  #  LOOP  #> type ;

: arguments  ( n -- )
 depth < not abort" Arguments?" ;
```

@ --> =

82

```
\ Spr move sprpos             20oct87re

: xmove   ( x spr#  -)
 2dup 2* sprbase + c!
 xposhi rot $FF > setbit ;

: ymove   ( y spr#  -)
 2* 1+ sprbase + c! ;

: move    ( y x spr#  -)
 under xmove ymove ;


: sprpos  ( spr# - y x)
 dup >r 2* 1+ sprbase + c@
 r@ 2* sprbase + c@
 r> sbittab + c@ xposhi c@ and
 IF $100 + THEN ;
```

167

```
\ tiny sprite editor          06nov87re

¦ Create savearea $1A allot
¦ Variable xsave    ¦ Variable ysave
¦ Variable saved    saved off

¦ : savesprites  ( -- )
 saved @ ?exit
 sprite savearea $1A cmove  0 sprite c!
 7 sprpos xsave ! ysave ! saved on ;

: fertig  ( -- )
 saved @ not ?exit
 ysave @ xsave @ 7 move
 savearea sprite $1A cmove saved off ;

¦ : sprline  ( adr line -- )
 base push  dup 2* + +  cr
 ." l:"  cbase @ base !
 dup count width #.r  count width #.r
 c@ width #.r  ." . $" hex 4 #.r ;
```

@ --> 1

83

```
\ Sprite Qualities            20oct87re

: high    ( spr# -)   yexpand set ;
: low     ( spr# -)   yexpand reset ;
: wide    ( spr# -)   xexpand set ;
: slim    ( spr# -)   xexpand reset ;
: big     ( spr# -)   dup high wide ;
: small   ( spr# -)   dup low slim ;
: behind  ( spr# -)   $D01B set ;
: infront ( spr# -)   $D01B reset ;
: colored ( spr# col  -)
 swap sprcol + c! ;
```

168

```
\ tiny sprite editor          06nov87re

¦ : slist  ( mem# -- )
 $40 * sprbuf +
 &21 0 DO  dup I sprline
           stop? IF  LEAVE  THEN  LOOP
 drop cr  ." fertig"  0 0 at quit ;

: sed  ( mem# -- )
 1 arguments  &32 min
 page dup . ." sed \ 1 color"
 savesprites  2 cbase !
 dup $40 $128 yel 7 setsprite
 7 3colored reset  7 big  slist ;

: ced  ( mem# -- )
 1 arguments  &32 min
 page dup . ." ced \ 3 colors"
 savesprites  4 cbase !
 blk gr2 sprcolors
 dup $40 $128 yel 7 setsprite
 7 3colored set  7 high  slist ;
```

= ╪

84 169

```
0 \ Spr sprcolors setsprite      20oct87re
1
2 : sprcolors  ( col# col#  -)
3  sprmcol 1+ c! sprmcol c! ;
4
5 : setsprite  ( mem# y x color spr# -)
6  under >r colored   r@ move
7  r@ under formsprite small
8  r@ 3colored reset  r> sprite set ;
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

```
0 \\ Directory ultraFORTH 4of4   26oct87re
1
2 .                          0
3 ..                         0
4 C16-Tape-Demo              2
5 C64-Grafik-Demo            6
6 cload/csave              &13
7 Tape-Version:LoadScreen  &16
8 Ramdisk                  &21
9 Supertape                &32
10 auto-Decompiler          &51
11 Screentausch             &61
12 Grafik                   &64
13 Mathematik               &90
14 Sieve Benchmark         &138
15 Grafik-Demo             &144
16 Sprite-Demo             &160
17 Sprite-Data             &165
18 Sprite-Editor           &166
19
20
21
22
23
24
```


```
0 \\ Directory ultraFORTH 4of4   26oct87re
1
2 .                          0
3 ..                         0
4 C16-Tape-Demo              2
5 C64-Grafik-Demo            6
6 cload/csave              &13
7 Tape-Version:LoadScreen  &16
8 Ramdisk                  &21
9 Supertape                &32
10 auto-Decompiler          &51
11 Screentausch             &61
12 Grafik                   &64
13 Mathematik               &90
14 Sieve Benchmark         &138
15 Grafik-Demo             &144
16 Sprite-Demo             &160
17 Sprite-Data             &165
18 Sprite-Editor           &166
19
20
21
22
23
24
```