0 85

```
0 \\ Directory ultraFORTH 3of4   26oct87re
1
2 .                    &0
3 ..                   &0
4 rom-ram-sys          &2
5 Transient-Assembler &4
6 Assembler-6502       &5
7 2words              &14
8 unlink              &15
9 scr<>cbm            &16
10 (search            &17
11 Editor             &19
12 .blk               &46
13 Tracer/Tools       &47
14 Multi-Tasker       &57
15 EpsonRX80          &63
16 VC1526             &75
17 CP-80              &78
18
19
20
21
22
23
24
```

1 86

```
0 \\ Inhalt ultraFORTH 3of4      26oct87re
1
2 rom ram sys          2 - 3
3 Transient Assembler  4
4 Assembler-6502       5 - 12
5                     13       frei
6 2words              14
7 unlink              15
8 scr<>cbm            16
9 (search             17
10 Editor             19
11 .blk               46
12 Tracer Tools       47
13 Multi-Tasker       57
14 Printer: EpsonRX80  63
15 Printer: VC1526     75
16 Printer: CP-80      78
17
18 Shadows             85 folgende
19
20
21
22
23
24
```

2 87

```
0 \ rom ram sys              clv/re20aug87
1 \              Shadow mit Ctrl+W--->
2
3 \ wird gebraucht, wenn
4 \ Spruenge ins ROM gehen.
5
6 Assembler also definitions
7 (16 \ Umschalten des Bereichs 8000-FFFF
8 : rom here 9 + $8000 u> abort" not here"
9       $ff3e sta ;
10 : ram  $ff3f sta ;
11 : sys rom jsr ram ;
12 \  wer unter diesem abort" not here"
13 \  leidet: s.naechster Screen --> C)
14
15
16 (64 \ Umschalten des Bereichs A000-BFFF
17 : rom here 9 + $A000 u> abort" not here"
18       $37 # lda 1 sta ;
19 : ram $36 # lda 1 sta ;
20 C)
21
22
23
24
```

```
\\ zu LongJsr fuer C16        clv08aug87

Das Speichermodell:

$0000 - $8000 : LowRAM
$8000 - $ffff : HighRAM  & ROM

Auf ROM schalten   Auf RAM schalten
sys kann wie jsr beutzt werden


ein ROM-Ruf der Art '0ffd2 sys'

 rom jsr ram  == $ff3e sta jsr $ff3f sta

das geht natuerlich nicht, wenn
HERE groesse $8000 ist. Warum wohl?

--- Beim c64 Lassen sich Basic und
 Betriebssystem getrennt schalten.
 Diese Makros sind nur fuer das
 Basic-Rom noetig.
```

3 88

```
\ sysMacro Long              clv20aug87re
\ zu LongJsr fuer C16        clv20aug87re

(64  .( Nicht fuer C64 !) \\ C)
ACHTUNG! bei falscher Benutzung
        Systemabsturz

\ Mit Makro: fuer Fortgeschrittene

here $8000 $20 - u> ?exit \ geht nicht!
das Makro muss immer unter $8000 liegen

' 0 ¦ Alias ???
ein Aufruf der Form ' $ffd2 sysMacro'
gibt:
Label long   ROM
Label long1  ??? jsr  RAM  rts end-code
   pha
   $ff # lda  LONG1 2+ sta
   $d2 # lda  LONG1 1+ sta
   pla  LONG jsr
¦ : sysMacro ( adr -- )
 $100 u/mod  pha  # lda  long1 2+ sta
 # lda  long1 1+ sta  pla  long jsr ;
so hat mittels Umleitung doch noch der
Sprung ins drueberliegende ROM geklappt

: sys ( adr -- ) \ fuer Jsr ins ROM
 here 9 + $8000 u>
 IF  sysMacro  ELSE  sys  THEN ;
sys entscheidet nun selbst, ob Umleitung
oder nicht.

ACHTUNG! DAS ZERO-Flag wird zerstoert!
```

4 89

```
\ transient Assembler        clv10oct87
( transient Forth-6502 Assemclv20aug87re
( Basis: Forth Dimensions VOL III No. 5)

\ Basis: Forth Dimensions VOL III No. 5)

Der Assembler wird komplett auf den
\ internal loading        04may85BP/re)
Heap geladen und ist so nur bis zum
naechsten 'clear' oder 'save' benutzbar,
here   $800 hallot  heap dp !
danach ist er komplett aus dem Speicher
entfernt. Er ist dann zwar nicht mehr
         1  +load
zu benutzen, aber er belegt auch nicht
unnoetig Speicherplatz.

dp !

Onlyforth
```

5 90

```
\ Forth-6502 Assembler       clv10oct87

\ Basis: Forth Dimensions VOL III No. 5)

Onlyforth  Assembler also definitions

1 7  +thru
 -3  +load \ Makros: rom ram sys

Onlyforth
```

6 91

```
\ Forth-83 6502-Assembler      20oct87re

: end-code   context 2- @  context ! ;

Create index
$0909 , $1505 , $0115 , $8011 ,
$8009 , $1D0D , $8019 , $8080 ,
$0080 , $1404 , $8014 , $8080 ,
$8080 , $1C0C , $801C , $2C80 ,

¦ Variable mode

: Mode:  ( n -)   Create c,
  Does>  ( -)     c@ mode ! ;

0  Mode: .A        1    Mode: #
2 ¦ Mode: mem      3    Mode: ,X
4  Mode: ,Y        5    Mode: X)
6  Mode: )Y       $F    Mode: )
```

7 92

```
\ upmode  cpu                  20oct87re

¦ : upmode ( addr0 f0 - addr1 f1)
 IF mode @  8 or mode !   THEN
 1 mode @  $F and ?dup IF
 0 DO  dup +  LOOP THEN
 over 1+ @ and 0= ;

: cpu  ( 8b -)   Create  c,
  Does>  ( -)    c@ c, mem ;

 00 cpu brk $18 cpu clc $D8 cpu cld
$58 cpu cli $B8 cpu clv $CA cpu dex
$88 cpu dey $E8 cpu inx $C8 cpu iny
$EA cpu nop $48 cpu pha $08 cpu php
$68 cpu pla $28 cpu plp $40 cpu rti
$60 cpu rts $38 cpu sec $F8 cpu sed
$78 cpu sei $AA cpu tax $A8 cpu tay
$BA cpu tsx $8A cpu txa $9A cpu txs
$98 cpu tya
```

8 93

```
\ m/cpu                        20oct87re

: m/cpu  ( mode opcode -)  Create c, ,
 Does>
 dup 1+ @ $80 and IF $10 mode +! THEN
 over $FF00 and upmode upmode
 IF mem true Abort" invalid" THEN
 c@ mode @ index + c@ + c, mode @ 7 and
 IF mode @  $F and 7 <
  IF c, ELSE , THEN THEN mem ;

$1C6E $60 m/cpu adc $1C6E $20 m/cpu and
$1C6E $C0 m/cpu cmp $1C6E $40 m/cpu eor
$1C6E $A0 m/cpu lda $1C6E $00 m/cpu ora
$1C6E $E0 m/cpu sbc $1C6C $80 m/cpu sta
$0D0D $01 m/cpu asl $0C0C $C1 m/cpu dec
$0C0C $E1 m/cpu inc $0D0D $41 m/cpu lsr
$0D0D $21 m/cpu rol $0D0D $61 m/cpu ror
$0414 $81 m/cpu stx $0486 $E0 m/cpu cpx
$0486 $C0 m/cpu cpy $1496 $A2 m/cpu ldx
$0C8E $A0 m/cpu ldy $048C $80 m/cpu sty
$0480 $14 m/cpu jsr $8480 $40 m/cpu jmp
$0484 $20 m/cpu bit
```

9 94

```
0 \ Assembler conditionals       20oct87re
1
2 ¦ : range?   ( branch -- branch )
3  dup abs  $7F u> Abort" out of range " ;
4
5 : [[  ( BEGIN)  here ;
6
7 : ?]  ( UNTIL)  c, here 1+ - range? c, ;
8
9 : ?[  ( IF)     c,  here 0 c, ;
10
11 : ?[[ ( WHILE)  ?[ swap ;
12
13 : ]?  ( THEN)   here over c@  IF swap !
14  ELSE over 1+ - range? swap c! THEN ;
15
16 : ][  ( ELSE)   here 1+   1 jmp
17  swap here over 1+ - range?  swap c! ;
18
19 : ]]  ( AGAIN)  jmp ;
20
21 : ]]? ( REPEAT) jmp ]? ;
22
23
24
```

10 95

```
0 \ Assembler conditionals       20oct87re
1
2 $90 Constant CS      $B0 Constant CC
3 $D0 Constant 0=      $F0 Constant 0<>
4 $10 Constant 0<      $30 Constant 0>=
5 $50 Constant VS      $70 Constant VC
6
7 : not    $20 [ Forth ] xor ;
8
9 : beq    0<> ?] ;    : bmi   0>= ?] ;
10 : bne   0=  ?] ;    : bpl   0<  ?] ;
11 : bcc   CS  ?] ;    : bvc   VS  ?] ;
12 : bcs   CC  ?] ;    : bvs   VC  ?] ;
13
14
15
16
17
18
19
20
21
22
23
24
```

11 96

```
0 \ 2inc/2dec   winc/wdec        20oct87re
1
2 : 2inc  ( adr -- )
3  dup lda  clc  2 # adc
4  dup sta  CS ?[  swap 1+ inc  ]?  ;
5
6 : 2dec  ( adr -- )
7  dup lda  sec  2 # sbc
8  dup sta  CC ?[  swap 1+ dec  ]?  ;
9
10 : winc  ( adr -- )
11  dup inc  0= ?[  swap 1+ inc  ]?  ;
12
13 : wdec  ( adr -- )
14  dup lda  0= ?[  over 1+ dec  ]?  dec  ;
15
16 : ;c:
17  recover jsr  end-code ]  0 last !  0 ;
18
19
20
21
22
23
24
```

12 97

```
0 \ ;code Code code>          bp/re03feb85
1
2 Onlyforth
3
4 : Assembler
5  Assembler   [ Assembler ] mem ;
6
7 : ;Code
8  [compile] Does>  -3 allot
9  [compile] ;      -2 allot   Assembler ;
10 immediate
11
12 : Code  Create here dup 2- ! Assembler ;
13
14 : >label  ( adr -)
15  here ! Create  immediate  swap ,
16  4 hallot heap 1 and hallot ( 6502-alig)
17  here 4 - heap  4  cmove
18  heap last @ count $1F and + !  dp !
19   Does>  ( - adr)   @
20   state @ IF  [compile] Literal  THEN ;
21
22 : Label
23  [ Assembler ]  here >label Assembler ;
24
```

13 98

```
0 \ frei                      20oct87re  \ frei                      20oct87re
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

14 99

```
0 \ 2! 2@ 2variable 2constant clv20aug87re
1
2 Code 2!  ( d adr --)
3  tya  setup jsr  3 # ldy
4  [[  SP )Y lda  N )Y sta  dey  0< ?]
5  1 # ldy  Poptwo jmp  end-code
6
7 Code 2@  ( adr -- d)
8  SP X) lda  N sta  SP )Y lda  N 1+ sta
9  SP 2dec  3 # ldy
10 [[  N )Y lda  SP )Y sta  dey  0< ?]
11 xyNext jmp  end-code
12
13 : 2Variable  ( --)   Create 4 allot ;
14              ( -- adr)
15
16 : 2Constant  ( d --)   Create , ,
17   Does> ( -- d)   2@ ;
18
19 \ 2dup  exists
20 \ 2swap exists
21 \ 2drop exists
22
23
24
```

15 100

```
0 \ unlink                    clv20aug87re
1
2 $FFF0 >label plot
3
4 (64
5
6 Code unlink  ( -- )
7   $288 lda  $80 # ora  tay  txa
8   [[  $D9 ,X sty  clc  $28 # adc
9    CS ?[  iny  ]?  inx  $1A # cpx  0= ?]
10  $D3 lda  $28 # cmp
11  CS ?[  $28 # sbc  $D3 sta  ]?
12  $D3 ldy  $D6 ldx  clc  plot jsr C)
13
14 (16 : unlink  0 0  $7EE 2! ; C)
15
16 Label setptrs
17  0 # ldx  1 # ldy  Next jmp  end-code
18
19
20
21
22
23
24
```

16 101

```
0 ( changing codes              18may85we)
1 ( Wie gut, dass commodore ...          )
2 ( ... besondere screen-codes hat.      )
3
4 Label (scr>cbm
5  N 6 + sta $3F # and  N 6 + asl
6  N 6 + bit  0< ?[ $80 # ora  ]?
7             VC ?[ $40 # ora  ]?  rts
8
9 Label (cbm>scr
10 N 6 + sta $7F # and $20 # cmp
11 CS ?[ $40 # cmp
12    CS ?[ $1F # and  N 6 + bit
13       0< ?[ $40 # ora  ]?  ]?  rts  ]?
14 Ascii . # lda  rts
15
16 Code cbm>scr  ( 8b1 -- 8b2)
17  SP X) lda  (cbm>scr jsr  SP X) sta
18  Next jmp  end-code
19
20 Code scr>cbm  ( 8b1 -- 8b2)
21  SP X) lda  (scr>cbm jsr  SP X) sta
22  Next jmp  end-code
23
24
```

17 102

```
0 \ schnelles search        bp   17jun85re
1
2 \needs Code -$D +load \ Trans Assembler
3
4 Onlyforth
5
6  ' 0< @ 4 +  >label puttrue
7 puttrue 3 +  >label putfalse
8
9 Code (search
10 ( text tlen buffer blen -- adr tf / ff)
11  7 # ldy
12  [[  SP )Y lda  N ,Y sta dey  0< ?]
13  [[ N 4 + lda  N 5 + ora  0<> ?[
14  [[ N     lda   N 1+ ora  0<> ?[
15     N 2+ X) lda  N 6 + X) cmp  swap
16     0<> ?[[  N wdec  N 2+ winc  ]]?
17
18 -->
19
20
21
22
23
24
```

18 103

```
\ Edi schnelles search    bp   17jun85re

 7 # ldy
 [[  N ,Y lda  SP )Y sta  dey  0< ?]
 [[  N 2+ winc  N 6 + winc  N wdec
N 4 + wdec  N 4 + lda  N 5 + ora
0= ?[  SP lda  clc  4 # adc  SP sta
       CS ?[  SP 1+ inc  ]?
       3 # ldy  N 3 + lda  SP )Y sta
       N 2+ lda  dey  SP )Y sta  dey
       puttrue jmp  ]?
N lda  N 1+ ora  0= ?[
3 roll  3 roll ]? ]?
SP lda  clc  6 # adc  SP sta
CS ?[  SP 1+ inc ]?  1 # ldy
putfalse jmp  ]?
N 2+ X) lda  N 6 + X) cmp  0= not ?]
7 # ldy
[[ SP )Y lda  N ,Y sta  dey  0< ?]
N wdec  N 2+ winc
( next char as first )  ]]  end-code
```

19 104

```
\ Editor loadscreen          clv13jul87
\ Idea and first implementation:  WE/re

Onlyforth
\needs .blk       $18 +load \ .blk
\needs Code       -$F +load \ Assembl
\needs (search     -2 +load \ (search

Onlyforth
(64 ¦ : at  at curoff ; C) \ sorry

\needs 2variable  -5 +load
\needs unlink     -4 +load  \ unlink
\needs scr>cbm    -3 +load  \ cbm><scr

Vocabulary Editor
Editor also definitions

                1 $17 +thru  \ Editor
             $18 $19 +thru  \ edit-view
                $1A +load  \ Ediboard

Onlyforth  1 scr !  0 r# !

save
```

20 105

```
\ Edi Constants Variables     clv15jul87

$28 ¦ Constant #col $19 ¦ Constant #row
#col  #row  *           ¦ Constant b/scr
  Variable shadow   $55 shadow !
¦ Variable ascr     1 ascr !
¦  Variable imode   imode off
¦ Variable char     #cr char !
¦ Variable scroll   scroll on
¦ Variable send     1 send !
¦ 2variable chars   ¦ 2variable lines
¦ 2variable fbuf    ¦ 2variable rbuf

(64 $288 C)  (16 $53e C)  >Label scradr
(64 $d800 C) (16 $800 C)  >Label coladr

$d1  (16 drop $c8 C) ¦ Constant linptr
$d3  (16 drop $ca C) ¦ Constant curofs

(64 $D020 C) (16 $ff19 C)
 ¦ Constant border
(64 $286  C) (16 $53b C) ¦ Constant pen
(64 $d021 C) (16 $ff15 C)
 ¦ Constant bkgrnd
```

21 106

```
0 ( Edi special cmoves          clv21.3.87)
1 ( Dank an commodore ...                 )
2
3 Label incpointer
4  N    lda  clc  #col 1+ # adc
5  N    sta  CS ?[  N 1+  inc  ]?
6  N 2+ lda  clc  #col    # adc
7  N 2+ sta  CS ?[  N 3 + inc  ]?  rts
8
9 ¦ Code b>sc   ( blkadr --)
10  tya  setup jsr
11  N 2+ stx  scradr lda  N 3 + sta
12  #row # ldx
13  [[  #col 1- # ldy
14      [[  N    )Y lda  (cbm>scr jsr
15          N 2+ )Y sta  dey  0< ?]
16      incpointer jsr  dex
17  0= ?]
18  pen lda
19  [[ coladr        ,X sta
20     coladr $100 + ,X sta
21     coladr $200 + ,X sta
22     coladr $300 + ,X sta
23     inx  0= ?]  setptrs jmp   end-code
24
```

22 107

```
0 ( Edi special cmoves cont.   clv21.3.87)
1 ( ... fuer dies Bildschirmformat.       )
2
3 ¦ Code sc>b   ( blkadr --)
4  tya  setup jsr
5  N 2+ stx  scradr lda  N 3 + sta
6  #row # ldx
7  [[  0 # ldy
8      [[  N 2+ )Y lda  (scr>cbm jsr
9          N )Y sta  iny  #col # cpy CS ?]
10     dex
11  0<> ?[[
12     bl # lda  N )Y sta
13     incpointer jsr
14  ]]?  setptrs jmp  end-code
15
16 ¦ Code >scrmove  ( from to 8bquan --)
17  3 # lda  setup jsr  dey
18  [[  N cpy  0= ?[  setptrs jmp  ]?
19      N 4 + )Y lda  (cbm>scr jsr
20      N 2+  )Y sta  iny  0= ?]  end-code
21
22
23
24
```

23 108

```
0 ( Edi changed?                clv21.3.87)
1
2 ¦ Code changed?   ( blkadr -- f)
3  tya  setup jsr
4  N 2+ stx  scradr lda  N 3 + sta
5  #row # ldx
6  [[  #col 1- # ldy
7      [[  N )Y lda  (cbm>scr jsr
8          N 2+ )Y cmp
9          0<> ?[  $FF # lda  PushA jmp ]?
10         dey 0<  ?]
11     incpointer jsr  dex
12  0= ?]
13  txa  PushA jmp  end-code
14
15 ¦ : memtop  sp@ #col 2* - ;
16
17
18
19
20
21
22
23
24
```

24 109

```
0 \ Edi c64-specials          clv2:jul87
1
2 ¦ Code scrstart  ( -- adr)
3  txa pha scradr lda  Push jmp end-code
4
5
6 ¦ Code rowadr  ( -- adr)
7  curofs lda  #col # cmp  txa
8  CS ?[  #col 1- # lda  ]?
9  linptr adc pha linptr 1 + lda  0 # adc
10  Push jmp  end-code
11
12 ¦ Code curadr  ( -- adr)
13  clc curofs lda linptr adc  pha
14  linptr 1 + lda 0 # adc Push jmp
15  end-code
16 (64
17 ¦ Code unlinked?     \ -- f
18  $D5 lda  #col # cmp  CC ?[  dex  ]?
19  txa  PushA jmp  end-code C)
20
21
22
23
24
```

25 110

```
0 \ Edi scroll? put/insert/do  clv2:jul87
1
2 ¦ : blank.end?  ( -- f)
3  scrstart [ b/scr #col - ] Literal +
4  #col -trailing nip  0=  scroll @ or ;
5
6 ¦ : atlast?  ( -- f)
7  curadr  scrstart b/scr + 1-  =
8  scroll @ 0=  and ;
9
10 ¦ : putchar  ( -- f)
11  char c@ con! false ;
12
13 ¦ : insert  ( -- f)
14  atlast?  ?dup ?exit
15 (64  unlinked? C) (16 true C)
16  rowadr #col + 1- c@  bl = not  and
17  blank.end? not  and  dup ?exit
18  $94 con! ;
19
20 ¦ : dochar  ( -- f)
21  atlast?  ?dup ?exit
22  imode @ IF insert ?dup ?exit
23  THEN putchar ;
24
```

26 111

```
0 ( Edi cursor control          15may85re)
1
2 ¦ : curdown  ( -- f)
3  scroll @ 0=  row  #row 2-  u>  and
4  dup ?exit $11 con! ;
5
6 ¦ : currite  ( -- f)
7  atlast? dup ?exit $10 con! ;
8
9  ' putchar ¦ Alias curup
10 ' putchar ¦ Alias curleft
11 ' putchar ¦ Alias home
12 ' putchar ¦ Alias delete
13
14 ¦ : >""end  ( -- ff)
15  scrstart b/scr -trailing nip
16  b/scr 1- min #col /mod swap at false ;
17
18 ¦ : +tab  ( -- f)
19  0  $a 0 DO  drop currite dup
20            IF LEAVE THEN  LOOP ;
21
22 ¦ : -tab  ( -- f)
23  5 0 DO $9D con!  LOOP  false ;
24
```

27 112

```
0 ( Edi cr, clear/newline      12jun85re)
1
2 : <cr>  ( -- f)
3  row 0 at  unlink  imode off  curdown ;
4
5 : clrline  ( -- ff)
6  rowadr #col bl fill false ;
7
8 : clrright  ( -- ff)
9  curadr #col col - bl fill false ;
10
11 : killine  ( -- f)
12  rowadr dup #col + swap
13  scrstart $3C0 + dup >r
14  over - cmove
15  r> #col bl fill false ;
16
17 : newline  ( -- f)
18  blank.end? not  ?dup ?exit
19  rowadr dup #col + scrstart b/scr +
20  over - cmove>  clrline ;
21
22
23
24
```

28 113

```
0 ( Edi character handling     18jun85re)
1
2 : dchar  ( -- f)
3  currite  dup ?exit $14 con! ;
4
5 : @char  ( -- f)
6  chars 2@ + 1+  lines @ memtop min
7  u>  dup ?exit
8  curadr c@  chars 2@ +  c!
9  1 chars 2+ +! ;
10
11 : copychar  ( -- f)
12  @char  ?dup ?exit  currite ;
13
14 : char>buf  ( -- f)
15  @char  ?dup ?exit  dchar ;
16
17 : buf>char  ( -- f)
18  chars 2+ @ 0=  ?dup ?exit
19  insert        dup ?exit
20  -1 chars 2+ +!
21  chars 2@ +  c@  curadr c! ;
22
23
24
```

29 114

```
0 ( Edi line handling, imode    18jun85re)
1
2 : @line  ( -- f)
3  lines 2@ +  memtop  u>  dup ?exit
4  rowadr  lines 2@ +  #col  cmove
5  #col lines 2+ +! ;
6
7 : copyline  ( -- f)
8  @line  ?dup ?exit  curdown ;
9
10 : line>buf  ( -- f)
11  @line  ?dup ?exit  killine ;
12
13 : !line  ( --)
14  #col negate lines 2+ +!
15  lines 2@ +  rowadr  #col  cmove  ;
16
17 : buf>line  ( -- f)
18  lines 2+ @ 0=  ?dup ?exit
19  newline  dup ?exit  !line ;
20
21 : setimd  ( -- f)   imode on false ;
22
23 : clrimd  ( -- f)   imode off false ;
24
```

30 115

```
 0 ( Edi the stamp              17jun85re)
 1
 2 Forth definitions
 3 : rvson $12 con! ;  : rvsoff $92 con! ;
 4
 5 Code ***ultraFORTH83***
 6      Next here 2- !  end-code
 7 : Forth-Gesellschaft   [compile] \\ ;
 8 immediate
 9
10 Editor definitions
11 Create stamp$ $12 allot stamp$ $12 erase
12
13 : .stamp  ( -- ff)
14  stamp$ 1+ count  scrstart #col +
15  over -   swap >scrmove false ;
16
17 : getstamp  ( --)
18  input push  keyboard  stamp$ on
19  cr ." your stamp: "  rvson $10 spaces
20  row $C at  stamp$ 2+ $10 expect
21  rvsoff  span @ stamp$ 1+ c! ;
22
23 : stamp?  ( --)
24  stamp$ c@ ?exit getstamp ;
```

31 116

```
 0 \ Edi the screen#            clv01aug87
 1
 2 : savetop  ( --)
 3  scrstart pad #col 2* cmove
 4  scrstart #col 2* $A0 fill ;
 5 : resttop  ( --)
 6  pad scrstart #col 2* cmove ;
 7 : updated?  ( scr# -- n)
 8  block 2- @ ;
 9 : special  ( --)
10  curon BEGIN pause key? UNTIL curoff ;
11
12 : drvScr ( --drv scr')
13  scr @ offset @ + blk/drv u/mod swap ;
14
15 : .scr#  ( -- ff) at? savetop  rvson
16  0 0 at drvScr ." Scr # " . " Drv " .
17  scr @ updated? 0=
18  IF ." not " THEN ." updated"  1 1 at
19  [ ' ***ultraFORTH83*** >name ] Literal
20  count type 2 spaces
21  [ ' Forth-Gesellschaft >name ] Literal
22  count $1F and type
23  rvsoff at special resttop false ;
24
```

32 117

```
 0 ( Edi exits                  20may85re)
 1
 2 : at?>r#  ( --)
 3  at? swap #col 1+ * + r# ! ;
 4
 5 : r#>at  ( --)
 6  r# @  dup  #col 1+  mod  #col =  -
 7  b/blk 1- min  #col 1+  /mod  swap at ;
 8
 9 : cancel  ( -- n)
10  unlink  %0001  at?>r# ;
11
12 : eupdate ( -- n)
13  cancel  scr @ block changed?
14  IF .stamp drop  scr @ block sc>b
15     update %0010 or THEN ;
16
17 : esave   ( -- n)   eupdate %0100 or ;
18
19 : eload   ( -- n)   esave   %1000 or ;
20
21
22
23
24
```

33 118

```
0 \ leaf thru Edi                clv01aug87
1
2 : elist  ( -- ff)
3  scr @ block b>sc  imode off  unlink
4  r#>at  false ;
5
6 : next    ( -- ff)
7  eupdate drop  1 scr +!  elist ;
8
9 : back    ( -- ff)
10 eupdate drop -1 scr +!  elist ;
11
12 : >shadow  ( -- ff)
13 eupdate drop  shadow @ dup drvScr nip
14 u> not IF negate THEN  scr +!  elist ;
15
16 : alter  ( -- ff)
17 eupdate drop  ascr @  scr @
18 ascr !  scr !  elist ;
19
20
21
22
23
24
```

34 119

```
0 \ Edi digits                   2oct87re
1
2 Forth definitions
3
4 : digdecode  ( adr cnt1 key -- adr cnt2)
5  #bs case?   IF  dup  IF
6                  del 1- THEN exit THEN
7  #cr case?   IF  dup span !  exit THEN
8  capital dup digit?
9  IF  drop >r 2dup + r@ swap c!
10     r> emit  1+  exit  THEN  drop ;
11
12 Input: digits
13  c64key c64key? digdecode c64expect ;
14
15 Editor definitions
16
17 : replace  ( -- f)
18  fbuf @ 0 DO  #bs con!  LOOP
19  false rbuf @ 0 DO insert or LOOP
20  dup ?exit
21  rbuf 2@ curadr swap >scrmove
22  eupdate drop ;
23
24
```

35 120

```
0 ( Edi >bufs                    20nov85re)
1
2 : .buf  ( adr count --)
3  type Ascii < emit
4  #col 1- col - spaces ;
5
6 : >bufs  ( --)
7  input push
8  unlink savetop at?  rvson
9  1 0 at ." replace with: "
10 at? rbuf 2@ .buf
11 0 0 at ." >     search: "
12 at? fbuf 2@ .buf
13 0 2  2dup at  send @ 3 u.r  2dup at
14 here 1+ 3 digits expect  span @ ?dup
15 IF  here under c!  number drop send !
16     THEN  at  send @ 3 u.r  keyboard
17 2dup at fbuf 2+ @  #col 2- col - expect
18 span @ ?dup IF  fbuf !  THEN
19 at fbuf 2@ .buf
20 2dup at rbuf 2+ @  #col 2- col - expect
21 span @ ?dup IF  rbuf !  THEN
22 at rbuf 2@ .buf
23 rvsoff resttop at ;
24
```

36 121

```
0 \ Edi esearch                clv06aug87
1
2 ¦ : (f      elist drop
3  fbuf 2@  r# @  scr @ block  +
4  b/blk r# @ - (search 0=
5  IF  0  ELSE  scr @ block -  THEN
6  r# !  r#>at ;
7
8 ¦ : esearch  ( -- f)
9  eupdate drop  >bufs
10  BEGIN BEGIN  (f  r# @
11        WHILE  key  dup Ascii r =
12               IF replace ?dup
13                  IF nip exit THEN THEN
14               3 = ?dup ?exit
15        REPEAT  drvScr nip send @ -
16        stop? 0= and ?dup
17  WHILE 0< IF   next drop
18           ELSE back drop THEN
19  REPEAT true ;
20
21
22
23
24
```

37 122

```
0 \ Edi keytable               clv2:jul87
1 ¦ : Ctrl  ( -- 8b)
2  [compile] Ascii $40 - ; immediate
3 ¦ Create keytable
4 Ctrl n c, Ctrl b c, Ctrl w c, Ctrl a c,
5 $1F c, (64 Ctrl ^ C)      (16 $92 C) c,
6 $0D c,   $8D c,
7 Ctrl c c, Ctrl x c, Ctrl f c, Ctrl l c,
8 $85 c,   $89 c,    $86 c,    $8A c,
9 $9F c,   $1C c, (64 00 C) (16 $1e C) c,
10 $88 c,   $87 c,    $88 c,    $8C c,
11 $1D c,   $11 c,    $9D c,    $91 c,
12 $13 c,   $93 c,    $94 c,
13 $14 c,    Ctrl d c, Ctrl e c, Ctrl r c,
14 Ctrl i c, Ctrl o c,
15                            $ff c,
16
17
18
19
20
21
22
23
24
```

38 123

```
0 ( Edi actiontable            clv9.4.87)
1
2
3 ¦ Create actiontable ]
4 next      back      >shadow   alter
5 esearch   copyline
6 <cr>      <cr>
7 cancel    eupdate   esave     eload
8 newline   killline  buf>line  line>buf
9 .stamp    .scr#           copychar
10 char>buf  buf>char  +tab      -tab
11 currite   curdown   curleft   curup
12 home      >"end     insert
13 delete    dchar     clrline   clrright
14 setimd    clrimd
15                             dochar  [
16 ¦ Code findkey  ( key n -- adr)
17  2 # lda  setup jsr  N ldy  dey
18  [[  iny  keytable ,Y lda  $FF # cmp
19      0<> ?[  N 2+ cmp  ]?  0= ?]
20  tya  .A asl  tay
21  actiontable    ,Y lda  pha
22  actiontable 1+ ,Y lda  Push jmp
23 end-code
24
```

39 124

```
0 ( Edi show errors           clv21.3.87)
1
2
3 ' 0   ¦ Alias dark
4
5 ' 1   ¦ Alias light
6
7 ¦ : half  ( n --)
8  border c!  pause $80 0 DO LOOP ;
9
10 ¦ : blink ( --)
11  border push  dark half light half
12               dark half light half ;
13
14 ¦ : ?blink ( f1 -- f2)
15  dup true = IF  blink 0=  THEN ;
16
17
18
19
20
21
22
23
24
```

40 125

```
0 ( Edi init                  18jun85re)
1
2 ' Literal ¦ Alias Li  immediate
3
4 Variable (pad       0 (pad !
5
6 ¦ : clearbuffer  ( --)
7  pad       dup  (pad  !
8  #col 2* + dup  fbuf  2+ !
9  #col    + dup  rbuf  2+ !
10  #col    + dup  chars !
11  #col 2* +      lines !
12  chars 2+ off  lines 2+ off
13  [ ' ***ultraFORTH83*** >name ] Li
14  count >r fbuf 2+ @ r@ cmove r> fbuf !
15  [ ' Forth-Gesellschaft >name ] Li
16  count $1F and >r
17  rbuf 2+ @ r@ cmove r> rbuf ! ;
18
19 ¦ : initptr ( --)
20  pad (pad @ = ?exit clearbuffer ;
21
22
23
24
```

41 126

```
0 \ Edi show                  clv15jul87
1
2 ' name >body 6 +  ¦ Constant 'name
3 (16 \ c16 benutzt standard C)
4
5 (64
6 ¦ Code curon
7  $D3 ldy    $D1 )Y lda  $CE sta
8  $80 # eor  $D1 )Y sta
9  xyNext jmp  end-code
10
11 ¦ Code curoff
12  $CE lda  $D3 ldy  $D1 )Y sta
13  xyNext jmp  end-code
14
15 C)
16
17
18
19
20
21
22
23
24
```

42 127

```
 0 ( Edi show                    17jun85re)
 1
 2 ' ; showoff
 3  ['] exit 'name !  rvsoff  curoff ;
 4
 5 ' ; show  ( --)
 6  blk @ ?dup 0= IF  showoff exit  THEN
 7  >in @ 1-  r# !  rvsoff curoff rvson
 8  scr @  over - IF  scr !  elist
 9  1 0 at .status THEN r#>at curon drop ;
10
11 Forth definitions
12
13 : (load  ( blk pos --)
14  >in push  >in !  ?dup 0= ?exit
15  blk push  blk !  .status interpret ;
16
17 : showload  ( blk pos -)
18  scr push  scr off  r# push
19  ['] show 'name ! (load showoff ;
20
21 Editor definitions
22
23
24
```

43 128

```
 0 \ Edi edit                    clv01aug87
 1 ' : setcol ( 0 / 4 / 8 --)
 2  ink-pot +
 3  dup c@ border c! dup 1+ c@ bkgrnd c!
 4   2+ c@ pen c! ;
 5 ' : (edit  ( -- n)
 6  4 setcol $93 con!
 7  elist drop  scroll off
 8  BEGIN key dup char c!
 9    0 findkey execute ?blink ?dup UNTIL
10  0 0 at killine drop  scroll on
11  0 setcol (16 0 $7ea c! C) \ Append-Mode
12 ;
13 Forth definitions
14 : edit ( scr# -) (16 c64fkeys C)
15  scr !  stamp?  initptr  (edit
16  $18 0 at  drvScr ." Scr " . ." Drv " .
17  dup 2 and 0=  IF ." not "     THEN
18                ." changed"
19  dup 4 and     IF save-buffers THEN
20  dup 6 and 6 = IF ." , saved"  THEN
21      8 and     IF ." , loading" cr
22        scr @  r# @  showload   THEN ;
23
24
```

44 129

```
 0 \ Editor Forth83              clv2:jul87
 1
 2 : l  ( scr -)   r# off  edit ;
 3 : r  ( -)       scr @ edit ;
 4 : +l ( n -)     scr @ + l ;
 5
 6 ' ,v  ( -) ( text)
 7    >name  ?dup IF  4 - @  THEN  ;
 8
 9 : view  ( -) ( text)
10  v ?dup
11  IF  l  ELSE  ." from keyboard"  THEN ;
12
13 Editor definitions
14
15 (16 ' ; curaddr \ --Addr
16      linptr @ curofs c@ + ; C)
17
18 : curlin  ( --curAddr linLen) \ & EOLn
19 (64 linptr @ $D5 c@ -trailing
20      dup $d3 c! C)
21 (16 $1b con! ascii j con! curaddr
22     $1b con! ascii k con! $1d con!
23      curaddr  over - C) ;
24
```

45 130

```
( Edidecode                clv26.3.87)

: edidecode  ( adr cnt1 key -- adr cnt2)
 $8D case? IF  imode off cr exit  THEN
 #cr case? IF  imode off
curlin dup span @ u> IF drop span @ THEN
  bounds ?DO
  2dup +  I c@ scr>cbm  swap c!  1+ LOOP
  dup span !  exit  THEN
 dup char c!
 $12 findkey execute ?blink drop ;


: ediexpect ( addr len1 -- )
 initptr  span !
 0 BEGIN  dup span @  u<
   WHILE  key decode  REPEAT
 2drop space ;

Input: ediboard
 c64key c64key? edidecode ediexpect ;

ediboard
```

46 131

```
( .status                   15jun85re)

' noop Is .status

: .blk  ( -)
 blk @ ?dup IF  ." Blk " u. ?cr  THEN ;

' .blk Is .status
```

47 132

```
\ tracer: loadscreen         clv12oct87

Onlyforth

\needs Code -$28 +load \ Trans Assembler

\needs Tools   Vocabulary Tools

Tools also definitions

   1 6  +thru  \ Tracer
   7 8  +thru  \ Tools for decompiling

Onlyforth

\\

Dieser wundervolle Tracer wurde
von Bernd Pennemann und Co fuer
den Atari entwickelt. Ich liess mir
aufschwatzen, ihn an C64/C16 anzupassen
und muss sagen, es ging erstaunlich
einfach. /clv
```

```
\\ zu tracer:loadscreen      clv12oct87

***Fuer die naechste u4th-Version****

Falls jemand mal die <IP IP>-Sache
ordnet und mit Atari vereinheitlicht,
hier ein paar kritische
Beispiele zum Testen:

: aa dup drop ;
: bb aa ;
\\
debug bb
trace' aa

trace' Forth


Mein Verdacht: Das IP 2inc findet bei
CBM/Atari vorher bzw. nachher statt.
```

48 133

```
0 \ tracer: wcmp variables      clv04aug87  \\ zu tracer:wcmp variables   clv04aug87
1
2 Assembler also definitions
3
4 : wcmp ( adr1 adr2--) \ Assembler-Macro   benutzt in der Form: adr1 adr2 wcmp
5  over lda dup cmp swap  \ compares word   vergleicht das ganze Wort. Danach
6  1+   lda 1+  sbc ;                       ist: Carry=1  : (adr1) >= (adr2)
7                                                Carry=0  : (adr1) <  (adr2)
8                                           mit den andern Flags ist nix anzufangen
9 Only Forth also Tools also definitions
10
11 ¦ Variable (W                           Temporaer Speicher fuer W
12 ¦ Variable <ip      ¦ Variable ip>       Bereich, in dem getraced werden soll
13 ¦ Variable nest?    ¦ Variable trap?     Flag: ins Wort rein  Flag: trace ja/nein
14 ¦ Variable last'     ¦ Variable #spaces   hab ich vergessen    Schachtelungstiefe
15
16
17
18
19
20
21
22
23
24                                                                          @
```

49 134

```
0 \ tracer:cpush oneline        clv12oct87  \ zu tracer:cpush oneline     clv04aug87
1
2 ¦ Create cpull    0  ]
3  rp@ count 2dup + rp! r> swap cmove ;
4
5 : cpush  ( addr len -)                    sichert LEN bytes ab ADDR auf dem
6  r> -rot   over  >r                       Return-Stack. Das naechste UNNEST
7  rp@ over 1+ - dup rp!  place             oder EXIT tut sie wieder zurueck
8  cpull >r  >r ;
9
10 ¦ : oneline  &82 allot keyboard display   die neue Hauptbefehlsschleife.
11  .status  space  query  interpret        Ermoeglicht die Eingabe einer Zeile.
12  -&82 allot  rdrop
13  ( delete quit from tnext )  ;
14
15 : range ( adr--) \ ermittelt <ip ip>     ermittelt den zu tracenden Bereich
16  ip> off  dup <ip !
17  BEGIN 1+ dup @
18    [ Forth ] ['] unnest = UNTIL
19  3+ ip> ! ;
20
21
22
23
24                                                                          @
```

50 135

```
0 \ tracer:step tnext           clv04aug87  \ zu tracer:step tnext        clv04aug87
1
2 ¦ Code step                               wird am Ende von TNEXT aufgerufen,
3  $ff # lda trap? sta trap? 1+ sta          um TRAP? wieder einzuschalten und
4           RP X) lda  IP sta               die angeschlagene NEXT-Routine
5  RP )Y lda  IP 1+ sta  RP 2inc            wieder zu reparieren.
6  (W lda  W sta   (W 1+ lda   W 1+ sta
7 Label W1-  W 1- jmp  end-code
8
9 ¦ Create: nextstep step ;
10                                          Diese Routine wird auf die NEXT-Routine
11 Label  tnext IP 2inc                     gepatched und ist das Kernstueck.
12  trap? lda  W1- beq                       Wenn nicht getraced wird: ab
13  nest? lda 0=  \ low(!)Byte test          Ins aktuelle Wort rein?
14  ?[ IP <ip wcmp W1- bcc                      nein: ist IP im debug-Bereich?
15     IP ip> wcmp W1- bcs                            nein: dann ab
16  ][ nest? stx  \ low(!)Byte clear            ja:   dann halb(!) loeschen
17  ]?
18   trap? dup stx 1+ stx \ disable tracer   trap? ausschalten ( der Tracer soll
19   W lda  (W sta    W 1+ lda   (W 1+ sta   sich schliesslich nicht selbst tracen,
20                                           wo kommen wir denn da hin!)
21
22
23
24                                                                          =
```

51

```
0 \ tracer:..tnext              clv12oct87
1
2  ;c: nest? @
3  IF nest? off r> ip> push <ip push
4     dup 2- range
5     #spaces push 1 #spaces +! >r THEN
6  r@  nextstep >r
7  input push    output push
8  2- dup last' !
9  cr #spaces @ spaces
10 dup 4 u.r @ dup 5 u.r space
11 >name .name  $10 col - 0 max spaces .s
12 state push  blk push  >in push
13 [ ' 'quit      >body ] Literal  push
14 [ ' >interpret >body ] Literal  push
15 #tib push  tib #tib @ cpush  r0 push
16 rp@ r0 !
17 ['] oneline Is 'quit  quit ;
18
19
20
21
22
23
24
```

```
\ tracer:..tnext              clv04aug87

Forth-Teil von TNEXT
 ins aktuelle Wort rein?
    ja: Debug-Bereich pushen, neuen
        Schachtelungstiefe incr.
 STEP soll nachher ausgefuehrt werden
 PUSHed alle wichtige Sachen

 gibt eine Informationszeile aus


 PUSHed nochmehr Zeug

 PUSHed den Return-Stack-Pointer !!
 und tut so, als waer der RStack leer
 Haengt ONELINE in die
 Haupt-Befehls-Schleife und ruft sie auf
```

52

```
0 \ tracer:do-trace traceable   clv12oct87
1
2 ¦ Code do-trace \ installs TNEXT
3  tnext 0 $100 m/mod
4      # lda  Next $c + sta
5      # lda  Next $b + sta
6  $4C # lda  Next $a + sta  Next jmp
7 end-code
8
9 ¦ : traceable ( cfa--<IP ) recursive
10 dup @
11 ['] :    @ case? IF >body     exit THEN
12 ['] key  @ case? IF >body c@ Input  @ +
13                    @ traceable exit THEN
14 ['] type @ case? IF >body c@ Output @ +
15                    @ traceable exit THEN
16 ['] r/w  @ case? IF >body
17                    @ traceable exit THEN
18 @  [ ' Forth @ @ ] Literal =
19                    IF @ 3 + exit THEN
20 \ fuer def.Worte mit does>
21 >name .name ." can't be DEBUGged"
22 quit ;
23
24
```

```
\ zu tracer:do-trace traceableclv12oct87

installiert (patched) TNEXT in NEXT
 (NEXT ist die innerste Routine,
  zu der jedes Wort zurueckkehrt)


guckt, ob Wort getraced werden kann
  und welche adr dazugehoert
 :      -def.  <IP=cfa+2
 INPUT: -def.  <IP aus input-Vektor

 OUTPUT:-def.  <IP aus output-Vektor

 DEFER  -def.  <IP aus [cfa+2]

 DOES>  -def.  <IP=[cfa]+3

 alle anderen Definitionen gehen nicht
```

53

```
0 \ tracer:Benutzer/innen-Worte clv12oct87
1
2 : nest   \ trace into current word
3  last' @ @ traceable drop nest? on ;
4
5 : unnest \ proceeds at calling word
6  <ip on ip> off ; \ clears trap range
7
8 : endloop last' @ 4 + <ip ! ;
9 \ no trace of next word to skip LOOP..
10
11 ' end-trace Alias unbug \ cont. execut.
12
13 : (debug  ( cfa-- )
14  traceable range
15  nest? off trap? on #spaces off
16  Tools do-trace ;
17
18 Forth definitions
19
20 : debug  ' (debug ; \ word follows
21
22 : trace'           \ word follows
23  ' dup (debug execute end-trace ;
24
```

```
\ zu tracer:Benutzer-Worte    clv12oct87

NEST erlaubt das Hineinsteigen in
     ein getracedes Worte

UNNEST fuehrt das Wort zuende und
     traced dann wieder.

ENDLOOP traced erst hinterm naechsten
     Wort wieder (z.B. bei LOOPs)

UNBUG schaltet jegliches getrace ab.


DEBUG <word> setzt den zu tracenden
     Bereich. Wenn <word> anschliessend
     ausgefuehrt wird, meldet sich der
     Tracer.

TRACE' fuehrt <word> gleich noch aus.
```

54

```
0 \ tools for decompiling,      clv12oct87
1
2 ( interactive use                    )
3
4 Onlyforth Tools also definitions
5
6 : ?:  ?cr dup 4 u.r ." :"  ;
7 : @?  dup @ 6 u.r ;
8 : c?  dup c@ 3 .r ;
9 : bl  $24 col - 0 max spaces ;
10
11 : s  ( adr - adr+)
12  ( print literal string)
13  ?:  space c? 4 spaces dup count type
14  dup c@ + 1+ bl  ;  ( count + re)
15
16 : n  ( adr - adr+2)
17  ( print name of next word by its cfa)
18  ?: @? 2 spaces
19  dup @ >name .name 2+ bl ;
20
21 : k  ( adr - adr+2)
22  ( print literal value)
23  ?: @? 2+ bl ;
24
```

139

```
\\ zu tools for decompil  01oct87clv/re)

\ Wenn zum Beispiel das Wort

: test 5  0  DO    ." magst Du mich ?"
    key Ascii j =
    IF   ." selber schuld " leave
    ELSE ." Aber bestimmt " THEN LOOP
  ." !" ;

\\ beguckt werden soll, dann gehts so:

  bitte umblaettern..>
```

55

```
0 ( tools for decompiling, interactive  )
1
2 : d  ( adr n - adr+n) ( dump n bytes)
3  2dup swap ?: 3 spaces  swap 0
4  DO  c? 1+ LOOP
5  4 spaces -rot type bl ;
6
7 : c  ( adr - adr+1)
8  ( print byte as unsigned value)
9  1 d ;
10
11 : b  ( adr - adr+2)
12  ( print branch target location )
13  ?: @? dup @  over + 6 u.r 2+ bl  ;
14
15 ( used for : )
16 ( Name String Literal Dump Clit Branch )
17 (  -    -      -       -    -    -     )
18
19
20
21
22
23
24
```

140

```
\  zu tools for decompil  01occlv10oct87

cr
tools
' test
k      n c n   n b   n s
   n   n     c n
   n b  n   s             n
   n b     n   s                  n b
  n s   n
```

56

```
0 ( debugging utilities      bp 19 02 85 )
1
2
3 : unravel   \  unravel perform (abort"
4  rdrop rdrop rdrop
5  cr ." trace dump is "  cr
6  BEGIN  rp@   r0 @ -
7  WHILE   r>  dup  8 u.r  space
8          2- @  >name .name  cr
9  REPEAT (error ;
10
11 ' unravel errorhandler !
12
13
14
15
16
17
18
19
20
21
22
23
24
```

141

57 142

```
0 \ Multitasker              BP 13.9.84 )
1
2 Onlyforth
3
4 \needs multitask  1 +load  save
5
6   2  4 +thru        \ Tasker
7 \    5 +load        \ Demotask
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

58 143

```
0 \ Multitasker              BP 13.9.84 )
1
2 \needs Code -$36 +load  \ transient Ass
3
4 Code stop
5  SP 2dec  IP    lda  SP X) sta
6           IP 1+ lda  SP )Y sta
7  SP 2dec  RP    lda  SP X) sta
8           RP 1+ lda  SP )Y sta
9  6 # ldy  SP    lda  UP )Y sta
10     iny  SP 1+ lda  UP )Y sta
11  1 # ldy  tya  clc  UP adc  W sta
12  txa  UP 1+ adc  W 1+ sta
13  W 1- jmp   end-code
14
15 ! Create taskpause   Assembler
16  $2C # lda  UP X) sta   ' stop @ jmp
17 end-code
18
19 : singletask
20  [ ' pause @ ] Literal  ['] pause ! ;
21
22 : multitask   taskpause ['] pause ! ;
23
24
```

59 144

```
0 \ pass  activate           ks 8 may 84 )
1
2 : pass  ( n0 .. nr-1 Tadr r -- )
3  BEGIN  [ rot ( Trick ! ) ]
4   swap  $2C over c! \ awake Task
5   r> -rot           \ IP r addr
6   8 + >r            \ s0 of Task
7   r@ 2+ @  swap     \ IP r0 r
8   2+ 2*             \ bytes on Taskstack
9                     \ incl. r0 & IP
10   r@ @ over -       \ new SP
11   dup r> 2- !       \ into ssave
12   swap bounds  ?DO  I !  2 +LOOP  ;
13 restrict
14
15 : activate ( Tadr --)
16  0 [ -rot ( Trick ! ) ]  REPEAT ;
17 -2 allot  restrict
18
19 : sleep  ( Tadr --)
20  $4C swap c! ;       \ JMP-Opcode
21
22 : wake  ( Tadr --)
23  $2C swap c! ;       \ BIT-Opcode
24
```

60

```
0 \ building a Task          BP 13.9.84 )
1
2 ¦ : taskerror  ( string -)
3  standardi/o  singletask
4  ." Task error :"  count type
5  multitask stop ;
6
7 : Task ( rlen  slen -- )
8  allot                \ Stack
9  here $FF and $FE =
10  IF 1 allot THEN     \ 6502-align
11  up@ here $100 cmove \ init user area
12  here  $4C c,        \ JMP opcode
13                      \     to sleep Task
14  up@ 1+ @ ,
15  dup  up@ 1+ !       \ link Task
16  3 allot             \ allot JSR wake
17  dup  6 -  dup , ,   \ ssave and s0
18  2dup +  ,           \ here + rlen = r0
19  under  + here - 2+ allot
20  ['] taskerror  over
21  [  errorhandler >body c@ ] Literal + !
22  Constant ;
23
24
```

145

61

```
0 \ more Tasks           ks/bp  26apr85re)
1
2 : rendezvous  ( semaphoradr -)
3  dup unlock pause lock ;
4
5 ¦ : statesmart
6  state @ IF [compile] Literal THEN ;
7
8 : 's  ( Tadr - adr.of.taskuservar)
9   ' >body c@ + statesmart ; immediate
10
11 \ Syntax:   2  Demotask 's base  !
12 \ makes Demotask working binary
13
14 : tasks  ( -)
15  ." MAIN"  cr up@ dup 1+ @
16  BEGIN  2dup - WHILE
17   dup [ ' r0 >body c@ ] Literal + @
18   6 + name> >name .name
19   dup c@ $4C = IF ." sleeping" THEN cr
20  1+ @ REPEAT  2drop ;
21
22
23
24
```

146

62

```
0 \ Taskdemo                  clv12aug87
1
2 : taskmark ; \needs cbm>scr : cbm>scr ;
3
4 : scrstart  ( -- adr)
5   (64 $288 C) (16 $53e C) c@ $100 * ;
6
7 Variable counter  counter off
8
9 $100 $100 Task Background
10
11 : >count  ( n -)
12  Background 1 pass
13  counter !
14  BEGIN  counter @  -1 counter +! ?dup
15  WHILE  pause 0 <# #s #>
16   0 DO  pause  dup I + c@  cbm>scr
17         scrstart I +  c!  LOOP  drop
18  REPEAT
19  BEGIN stop REPEAT ; \ stop's forever
20 : wait  Background sleep ;
21 : go    Background wake ;
22
23 multitask       $100 >count  page
24
```

147

63 148

```
0 \ printer loadscreen          27jul85re)
1
2 Onlyforth hex                           setzt order auf FORTH FORTH ONLY   FORTH
3
4 Vocabulary Print
5 Print definitions also
6
7 Create Prter 2 allot  ( Semaphor)       fuer multitasking
8 Prter off
9
10   : ) ; immediate
11   : (u ; immediate  \ for user-port    Centronics-Schnittstelle ueber User-Port
12   : (s  [compile] ( ; immediate       (s Text bis ) wird ueberlesen
13 \ : (s ; immediate  \ for serial bus   serielle Schnittstelle (wegkommentiert)
14 \ : (u  [compile] ( ; immediate        (u Text bis ) wird ueberlesen
15
16 (s  1 +load )                          lade den naechsten Screen nur fuer
17                                             seriellen Bus
18  2 $A +thru
19
20 Onlyforth
21
22 clear                                  unbrauchbare Koepfe weg
23
24                                                                      @
```

64 149

```
0 \ Buffer for the ugly SerBus  28jul85re)  Beim seriellen Bus ist die Ausgabe jedes
1                                           einzelnen Zeichens zu langsam
2 $100 ¦ Constant buflen
3
4 ¦ Variable Prbuf  buflen allot Prbuf off  Buffer fuer Zeichen zum Drucker
5
6 ¦ : >buf  ( char --)                      ein Zeichen zum Buffer hinzufuegen
7  Prbuf count + c!  1 Prbuf +! ;
8
9 ¦ : full?  ( -- f)   Prbuf c@ buflen = ;  Buffer voll?
10
11 ¦ : .buf  ( --)                          Buffer ausdrucken und leeren
12  Prbuf count -trailing
13  4 0 busout bustype busoff Prbuf off ;
14
15 : p!  ( char --)                         Hauptausgaberoutine fuer seriellen Bus
16  pause  >r                               Zeichen merken
17  r@ $C ( Formfeed  ) =                   ist es ein Formfeed?
18  IF  r> >buf .buf exit  THEN               ja, Buffer ausdrucken incl. Formfeed
19  r@ $A ( Linefeed  ) =                   ist es ein Linefeed?
20  r@ $D ( CarReturn ) = or  full? or      oder ein CR oder ist der Buffer voll?
21  IF  .buf  THEN  r> >buf ;                 ja, Buffer ausdrucken, CR/LF merken
22
23
24                                                                      @
```

65 150

```
0 \ p! ctrl: ESC esc:           28jul85re)
1
2 (u
3 : p!  \ char --                         Hauptausgaberoutine fuer Centronics
4  $DD01 c!  $DD00 dup c@ 2dup            Zeichen auf Port  , Strobe-Flanke
5  4 or swap c!  $F8 and swap c!                ausgeben
6   BEGIN  pause  $DD0D c@ $10 and        wartet bis Busy-Signal zurueckgenommen
7   UNTIL ;  )                            wird
8
9 ¦ : ctrl:  ( 8b --)   Create c,         gibt Steuerzeichen an Drucker
10   does>  ( --)   c@ p! ;
11
12    7 ctrl: BEL    ¦ $7F ctrl: DEL      Steuerzeichen fuer den Drucker
13 ¦ $d ctrl: CRET   ¦ $18 ctrl: ESC      in hexadezimaler Darstellung
14   $a ctrl: LF       $0C ctrl: FF       gegebenenfalls anpassen !
15
16 ¦ : esc:  ( 8b --)   Create c,         gibt Escape-Sequenzen an Drucker
17   does>  ( --)   ESC c@ p! ;
18
19  $30 esc: 1/8"       $31 esc: 1/10"    Zeilenabstand in Zoll
20  $32 esc: 1/6"
21  $54 esc: suoff                        Superscript und Subscript ausschalten
22  $4E esc: +jump      $4F esc: -jump    Perforation ueberspringen ein/aus
23
24                                                                      @
```

66 151

```
0 \ printer controls          28jul85re)
1
2 ¦ : ESC2   ESC  p! p! ;                  Escape + 2 Zeichen
3
4   : gorlitz  ( 8b --)   BL ESC2 ;         nur fuer Goerlitz-Interface
5
6 ¦ : ESC"!"  ( 8b --)   $21 ESC2 ;        spezieller Epson-Steuermodus
7
8 ¦ Variable Modus  Modus off               Kopie des Drucker-Steuer-Registers
9
10 ¦ : on:  ( 8b --)  Create c,            schaltet Bit in Steuer-Register ein
11   does>  ( --)
12   c@ Modus c@ or dup Modus c! ESC"!" ;
13
14 ¦ : off:  ( 8b --)   Create $FF xor c,  schaltet Bit in Steuer-Register aus
15   does>  ( --)
16   c@ Modus c@ and dup Modus c! ESC"!" ;
17
18  $10 on: +dark    $10 off: -dark        Diese Steuercodes muessen fuer andere
19  $20 on: +wide    $20 off: -wide        Drucker mit Hilfe von ctrl:, esc: und
20  $40 on: +cursiv  $40 off: -cursiv      ESC2 umgeschrieben werden
21  $80 on: +under   $80 off: -under
22 ¦  1 on: (12cpi                         Zeichenbreite in characters per inch
23 ¦  4 on: (17cpi     5 off: 10cpi        eventuell durch Elite, Pica und Compress
24                                         ersetzen                              @
```

67 152

```
0 \ printer controls          28jul85re)
1
2 : 12cpi   10cpi (12cpi ;                 gegebenenfalls aendern
3 : 17cpi   10cpi (17cpi ;
4 : super   0 $53 ESC2 ;
5 : sub     1 $53 ESC2 ;
6 : lines  ( #lines --)  $43 ESC2 ;        Aufruf z.B.mit 66 lines
7 : "long  ( inches --)   0 lines p! ;     Aufruf z.B mit 11 "long
8 : american   0 $52 ESC2 ;                Zeichensaetze, beliebig erweieterbar
9 : german     2 $52 ESC2 ;
10
11 : prinit                                Initialisierung ...
12 (s  Ascii x gorlitz  Ascii b gorlitz      . fuer Goerlitz-Interface
13     Ascii e gorlitz  Ascii t gorlitz
14     Ascii z gorlitz  Ascii l gorlitz )
15 (u  $FF $DD03 c!                          . fuer Centronics:  Port B auf Ausgabe
16     $DD02 dup c@  4 or swap c! ) ;                  PA2 auf Ausgabe fuer Strobe
17
18 ¦ Variable >ascii  >ascii on            Flag fuer Zeichen-Umwandlung
19
20 : normal   >ascii on                    schaltet Drucker mit Standardwerten ein
21   Modus off  10cpi  american  suoff
22   1/6"  $c "long  CRET ;
23
24                                                                             @
```

68 153

```
0 \ Epson printer interface     08sep85re)
1
2 ¦ : c>a  ( 8b0 -- 8b1)                   wandelt Commodore's Special-Ascii in
3  >ascii @ IF                             ordinaeres ASCII
4 dup $41 $5B uwithin IF $20 or  exit THEN
5 dup $C1 $DB uwithin IF $7F and exit THEN
6 dup $DC $E0 uwithin IF $A0 xor THEN
7  THEN ;
8
9 ¦ Variable pcol  pcol off
10 ¦ Variable prow  prow off               Routinen zur Druckerausgabe      Befehl
11
12 ¦ : pemit  c>a p!  1 pcol +! ;          ein Zeichen auf Drucker          emit
13 ¦ : pcr   CRET LF  1 prow +!  0 pcol ! ; CR und LF auf Drucker            cr
14 ¦ : pdel   DEL  -1 pcol +! ;            ein Zeichen loeschen (?!)        del
15 ¦ : ppage  FF  0 prow !  0 pcol ! ;     Formfeed ausgeben                page
16 ¦ : pat    ( zeile spalte -- )          Drucker auf zeile und spalte     at
17   over   prow @ < IF  ppage  THEN       positionieren, wenn noetig,
18   swap prow @ - 0 ?DO pcr LOOP          neue Seite
19   dup  pcol < IF  CRET  pcol off  THEN
20   pcol @ - spaces ;
21 ¦ : pat?   prow @  pcol @ ;             Position feststellen             at?
22 ¦ : ptype  ( adr count --)  dup pcol +!  Zeichenkette ausgeben            type
23  bounds ?DO  I c@ c>a p!  LOOP ;
24                                                                             =
```

69 154

```
 0 \ print  pl                  02oct87re
 1
 2 ¦ Output: >printer                        erzeugt die Ausgabetabelle >printer
 3  pemit pcr ptype pdel ppage pat pat? ;
 4                                           Routinen fuer Drucker
 5                                           und Bildschirm gleichzeitig (both)
 6 : bemit   dup  c64emit  pemit ;
 7 : bcr          c64cr    pcr   ;          Ausgabe erfolgt zuerst auf Bildschirm
 8 : btype   2dup c64type  ptype ;          ( Routinen von DISPLAY )
 9 : bdel         c64del   pdel  ;          dann auf Drucker
10 : bpage        c64page  ppage ;          ( Routinen von >PRINTER )
11 : bat    2dup c64at    pat   ;
12
13 ¦ Output: >both
14  bemit bcr btype bdel bpage bat pat? ;   erzeugt die Ausgabetabelle >both
15
16 Forth definitions
17                                           Worte sind von Forth aus zugaenglich
18 : Printer
19   normal  (u prinit )  >printer ;         legt Ausgabe auf Drucker
20 : Both
21   normal  >both ;                         legt Ausgabe auf Drucker und Bildschirm
22
23
24                                                                          @
```

70 155

```
 0 \ 2scr's nscr's thru      ks  28jul85re)
 1
 2 Forth definitions
 3
 4 ¦ : 2scr's  ( blk1 blk2 --)               gibt 2 Screens nebeneinander aus
 5  cr LF  17cpi  +wide +dark $15 spaces     Screennummer in Fettschrift und 17cpi
 6  over 3 .r $13 spaces dup 3 .r
 7  -dark -wide cr  b/blk 0 DO
 8   cr I c/l / $15 .r  4 spaces             formatierte Ausgabe der beiden Screens
 9   over block I +  C/L 1- type  5 spaces
10   dup  block I +  C/L 1- -trailing type
11  C/L +LOOP  2drop  cr ;
12
13 ¦ : nscr's  ( blk1 n -- blk2)   2dup      gibt die Screens so aus:   1    3
14  bounds DO I  over I + 2scr's LOOP + ;                              2    4
15
16 : pthru  ( from to --)                    gibt die Screens von from bis to aus
17  Prter lock  Output push Printer  1/8"    Ausgabegeraet merken und Printer ein
18  1+ over - 1+ -2 and 6 /mod               errechnet Druckposition der einzelnen
19  ?dup IF swap >r                          Screens und gibt sie nach obigem Muster
20  0 DO 3 nscr's 2+ 1+ page LOOP  r> THEN   aus
21  ?dup IF 1+ 2/ nscr's page THEN drop
22  Prter unlock ;
23
24                                                                          @
```

71 156

```
 0 \ Printing with shadows       28jul85re)
 1
 2 Forth definitions
 3
 4 ¦ : 2scr's  ( blk1 blk2 --)               wie 2scr's (mit Shadow)
 5  cr LF  17cpi  +wide +dark $15 spaces
 6  dup  3 .r
 7  -dark -wide cr  b/blk 0 DO
 8   cr I c/l / $15 .r  4 spaces
 9   dup  block I +  C/L 1- type  5 spaces
10   over block I +  C/L 1- -trailing type
11  C/L +LOOP  2drop  cr ;
12
13 ¦ : nscr's  ( blk1 n -- blk2)             wie nscr's (mit Shadow)
14  0 DO dup [ Editor ]  shadow @   2dup                    screen Shadow
15  u> IF negate THEN                                       scr+1  Sh+1
16  + over 2scr's 1+ LOOP ;
17
18 : dokument  ( from to --)                 wie pthru  (mit Shadow)
19  Prter lock  Output push  Printer
20  1/8"  1+ over - 3 /mod
21  ?dup IF swap >r
22  0 DO 3 nscr's page LOOP  r> THEN
23  ?dup IF nscr's page THEN drop
24  Prter unlock ;                                                          =
```

72 157

```
0 \ 2scr's nscr's thru      ks  28jul85re)
1
2 Forth definitions  $40 ¦ Constant C/L
3
4 ¦ : 2scr's  ( blk1 blk2 --)              Dasselbe nochmal fuer Standard-Forth
5  pcr LF LF 10cpi +dark  $12 spaces       Screens mit 16 Zeilen zu 64 Zeichen
6  over 3 .r  $20 spaces dup 3 .r
7  cr 17cpi -dark
8  $10 C/L * 0 DO cr over block I + C/L
9  6 spaces type 2 spaces
10 dup block I + C/L -trailing type
11 C/L  +LOOP  2drop cr ;
12
13 ¦ : nscr's ( blk1 n -- blk2)   under 0   Siehe oben
14 DO 2dup dup rot + 2scr's 1+ LOOP nip ;
15
16 : 64pthru  ( from to --)                Wie pthru fuer Standard-Screens
17  Prter lock  >ascii push  >ascii off
18  Output push  Printer
19  1/6" 1+ over - 1+ -2  and 6 /mod
20  ?dup IF swap >r
21  0 DO 3 nscr's 2+ 1+ page LOOP  r> THEN
22  ?dup IF 1+ 2/ nscr's page THEN drop
23  Prter unlock  ;
24
```

73 158

```
0 \ pfindex                    02oct87re
1
2 Onlyforth Print also
3
4 : pfindex  ( from to --)                 Ein schnelles Index auf den Drucker
5  Prter lock  Printer  &12 "long                          12" Papierlaenge
6  +jump  findex  cr page  -jump            Perforation ueberspringen
7  Prter unlock  display  ;
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

74 159

```
0 \ Printspool                 02oct87re  Drucken im Untergrund
1
2 \needs tasks  .( Tasker?!) \\            Der Tasker wird gebraucht
3
4 $100 $100 Task Printspool                Der Arbeitsbereich der Task wird erzeugt
5
6 : spool  ( from to --)                   Hintergrund-Druck ein
7  Printspool 2 pass                        von/bis werden an die Task gegeben
8                                           beim naechsten PAUSE fuehrt die
9  pthru                                    Task pthru aus und legt sich dann
10 stop ;                                   schlafen.
11
12 : endspool  ( --)                       Hintergrund-Druck abbrechen
13  Printspool activate                      die Task wird nur aktiviert,
14  stop ;                                   damit sie sich sofort wieder schlafen
15                                           legt.
16
17
18
19
20
21
22
23
24
```

75

```
\ Printer Routinen 1526       clv14oct87
( Nicht geeignet fuer Printspool!!   re)

Onlyforth  Vocabulary Print

Print also Definitions

: prinit   4 7 busout ( drop ) ;
\needs FF  : FF noop ;
: CRET     $d bus! ;

: pspaces  ( n -)
  0 ?DO  BL bus! LOOP ;

1 2 +thru

Only Forth also Definitions

( save )
```

160

```
\\ zu Printer interface 1526  clv14oct87
Dieser Treiber laueft auch mit:

 C16 & CITIZEN-100DM \ s.Handbuch

<--dieses DROP war doch wohl falsch /clv

FF : Die Formfeed-Definition fehlt
     hier. Wer's kann schreibe sie
     sich selber, wer's nicht kann,
     arbeite halt ohne Seiten-Vorschuebe
```

76

```
\ Printer interface 1526       02oct87re

Variable Pcol   Variable Prow

: pemit  bus!    1 Pcol +! ;
: pcr    CRET    1 Prow +!  0 Pcol ! ;
: pdel   ;
: ppage  FF  0 Prow !  0 Pcol ! ;
: pat    ( zeile spalte -- )
  over   Prow @ < IF  ppage  THEN
  0 rot  Prow @ - bounds ?DO pcr LOOP
  dup  Pcol @ - pspaces  Pcol ! ;
: pat?   Prow @  Pcol @ ;
: ptype  ( adr count -)  dup Pcol +!
 bounds ?DO  I c@ bus! LOOP ;

\ Output: >printer
 pemit pcr ptype pdel ppage pat pat? ;

Forth definitions

: printer   prinit >printer ;

: display   cr busoff display ;
```

161

```
clv14oct87
```

77

```
\ printer routinen             20oct87re

Only Forth also definitions

4 Constant B/scr

: .line  ( line# scr# --)
  block swap c/l * + c/l 1- type ;

: .===
 c/l 1- 0 DO  Ascii = emit  LOOP ;

: prlist ( scr# --)
 dup block drop    printer
 $E emit ." Screen Nr. " dup . $14 emit
 cr .===
 l/s 0 DO I over .line cr LOOP drop
 .=== cr cr cr  display ;
```

162

78 163

```
0 \ CP-80 Printer loadscreen    clv14oct87
1
2 Onlyforth hex
3
4 Vocabulary Print  Print definitions also
5
6 Create Prter 2 allot  ( Semaphor)
7
8 0 Prter !   \ Prter unlock /clv
9
10 1 6 +thru
11
12 Only Forth also definitions
13
14 (  clear   )
15
16
17
18
19
20
21
22
23
24
```

79 164

```
0 \ p! ctrl: ESC esc:          07may85mawe)
1
2 Print definitions
3
4 : p!  ( 8b -)
5  BEGIN  pause  $DD0D c@  $10 and  UNTIL
6  $DD01 c! ;
7
8 ¦ : ctrl:  ( B -)   Create c,
9   does>  ( -)   c@ p! ;
10
11    07 ctrl: BEL     ¦ $7F ctrl: DEL
12 ¦ $0D ctrl: CRET    ¦ $1B ctrl: ESC
13   $0A ctrl: LF        $0C ctrl: FF
14
15 ¦ : esc:   ( B -)   Create c,
16   does>  ( -)   ESC c@ p! ;
17
18  $30 esc: 1/8"        $31 esc: 1/10"
19  $32 esc: 1/6"        $20 esc: gorlitz
20
21 ¦ : ESC2   ESC  p! p! ;
22
23
24
```

80 165

```
0 ( printer controls           07may85mawe)
1
2  $0e esc: +wide  $14 esc: -wide
3  $45 esc: +dark  $46 esc: -dark
4  $47 esc: +dub   $48 esc: -dub
5  $0f esc: +comp  $12 esc: -comp
6
7 : +under 1 $2D esc2 ;
8 : -under 0 $2D esc2 ;
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

81 166

```
 0 ( printer controls          07may85maHe)
 1
 2   $54 esc: suoff
 3
 4 : super   0 $53 ESC2 ;
 5
 6 : sub     1 $53 ESC2 ;
 7
 8 : lines  ( lines -)   $43 ESC2 ;
 9
10 : "long  ( inches -)   0 lines p! ;
11
12 : american   0 $52 ESC2 ;
13
14 : german     2 $52 ESC2 ;
15
16 : pspaces  ( n -)
17   0 swap bounds ?DO  BL p!  LOOP ;
18
19 : initport  0 $DD01 c! $FF $DD03 c! ;
20
21 : prinit   initport
22   american  suoff  1/6"
23   &12 "long  CRET ;
24
```

82 167

```
 0 ( CP80  printer interface     26mar85re)
 1
 2 Variable unchanged?  unchanged? off
 3
 4 : c>a  ( 8b0 - 8b1)
 5  unchanged? @ ?exit
 6  dup $41 $5B uwithin
 7                 IF  $20 or  exit THEN
 8  dup $C1 $DB uwithin
 9                 IF  $7F and exit THEN
10  dup $DC $E0 uwithin
11                 IF  $A0 xor      THEN ;
12
13
14
15
16
17
18
19
20
21
22
23
24
```

83 168

```
 0 ( print  pl                   06may85we)
 1
 2 Variable Pcol   Variable Prow
 3
 4 : pemit  c>a p!  1 Pcol +! ;
 5 : pcr    CRET    1 Prow +!  0 Pcol ! ;
 6 : pdel   DEL  -1 Pcol +! ;
 7 : ppage  FF  0 Prow !  0 Pcol ! ;
 8 : pat    ( zeile spalte -- )
 9   over   Prow @ < IF  ppage  THEN
10   0 rot  Prow @ - bounds ?DO pcr LOOP
11   dup  Pcol @ - pspaces  Pcol ! ;
12 : pat?   Prow @  Pcol @ ;
13 : ptype  ( adr count -)  dup Pcol +!
14  bounds ?DO  I c@ c>a p!  LOOP ;
15
16 Output: >printer
17  pemit pcr ptype pdel ppage pat pat? ;
18
19 Forth definitions
20
21 : Printer   prinit  >printer ;
22
23
24
```

84 169

```
 0 ( 3scr's nscr's thru      ks07may85maWe)
 1 Forth definitions
 2
 3 ¦ : 3scr's  ( blk  -)
 4  cr  -comp +dark
 5   $B spaces dup    3 .r
 6  $19 spaces dup 1+ 3 .r
 7  $19 spaces dup 2+ 3 .r
 8  cr  +comp  -dark  L/S C/L *  0 DO
 9   cr 5 spaces dup  block I + C/L 1- type
10    8 spaces dup 1+ block I + C/L 1- type
11    8 spaces dup 2+ block I + C/L 1- type
12  C/L +LOOP  drop  cr LF ;
13
14 ¦ : nscr's ( blk1 n - blk2)  under 0
15  DO dup 3scr's over + LOOP nip ;
16
17 : pthru  ( from to -)
18  Output @ -rot  Printer Prter lock 1/8"
19  1+ over - 1+ 9 /mod
20  ?dup IF swap >r
21  0 DO 3 nscr's  page LOOP  r> THEN
22  ?dup IF 1- 3 * 1+ 0
23    DO dup 3scr's 3 + LOOP  THEN drop
24  Prter unlock  Output ! ;                                          @
```

o o

```
 0 \\ Directory ultraFORTH 3of4   26oct87re  \\ Directory ultraFORTH 3of4   26oct87re
 1
 2 .                    &0                   .                    &0
 3 ..                   &0                   ..                   &0
 4 rom-ram-sys          &2                   rom-ram-sys          &2
 5 Transient-Assembler  &4                   Transient-Assembler  &4
 6 Assembler-6502       &5                   Assembler-6502       &5
 7 2words              &14                   2words              &14
 8 unlink              &15                   unlink              &15
 9 scr<>cbm            &16                   scr<>cbm            &16
10 (search             &17                   (search             &17
11 Editor              &19                   Editor              &19
12 .blk                &46                   .blk                &46
13 Tracer/Tools        &47                   Tracer/Tools        &47
14 Multi-Tasker        &57                   Multi-Tasker        &57
15 EpsonRX80           &63                   EpsonRX80           &63
16 VC1526              &75                   VC1526              &75
17 CP-80               &78                   CP-80               &78
18
19
20
21
22
23
24                                  @                                  @
```

o o

```
 0 \\ Directory ultraFORTH 3of4   26oct87re  \\ Directory ultraFORTH 3of4   26oct87re
 1
 2 .                    &0                   .                    &0
 3 ..                   &0                   ..                   &0
 4 rom-ram-sys          &2                   rom-ram-sys          &2
 5 Transient-Assembler  &4                   Transient-Assembler  &4
 6 Assembler-6502       &5                   Assembler-6502       &5
 7 2words              &14                   2words              &14
 8 unlink              &15                   unlink              &15
 9 scr<>cbm            &16                   scr<>cbm            &16
10 (search             &17                   (search             &17
11 Editor              &19                   Editor              &19
12 .blk                &46                   .blk                &46
13 Tracer/Tools        &47                   Tracer/Tools        &47
14 Multi-Tasker        &57                   Multi-Tasker        &57
15 EpsonRX80           &63                   EpsonRX80           &63
16 VC1526              &75                   VC1526              &75
17 CP-80               &78                   CP-80               &78
18
19
20
21
22
23
24                                  @                                  @
```