Internes

Sourcecode

Der rekonstruierte Code des Z80 FORTH 83 der IG Forth. Vieles ist 1:1 aus Laxen/Perry F83 übernommen. Ich bin mir nicht sicher, welche Version zur Verfügung stand. Es scheinen teilweise Codeschnipsel aus der V1.0 (Sep83) zu stammen, vieles aber auch aus V2.10 (Apr84). Einige Primitive-Implementierungen scheinen auch FIG-Forth entlehnt.

Im Unterschied zu Laxen/Perry wird Register IY als Returnstack genutzt. Ein Z80-Prozessor ist damit Voraussetzung. Laxen/Perry arbeitet mit einem 8080-Prozessor. Mehrere Worte nutzen auch die Vorteile des Z80 aus und sind dadurch intern anders programmiert.

Die Register IY und BC (W) dürfen in eigenen Maschinencodeworten nicht verändert werden.

Im Folgenden sind Primitive nicht aufgeführt. diese stehen komplett im Assemblerquellcode forth83.asm bzw. forth83.lst. @@adr bedeutet, dass hier die Adresse aus dem Maschinencodeteil genommen wird. Daher ist der folgende Forth-Code kein kompletter Code zum Meta-Compilieren, sondern eher zum Nachschlagen gedacht.

```
( Z80 FORTH 83 der IG Forth)
( based on Laxen/Perry CP/M-F83 V2.0)
( reconstructed VPohlers Jan 2020)
( complete assembler source see forth83.asm)

VOCABULARY FORTH
FORTH DEFINITIONS
VOCABULARY HIDDEN

\ Run Time Code for Defining Words
Exec-Code f. Colon-Definitionen @@DOCOL
CODE EXIT
HIDDEN DEFINITIONS
CODE UNNEST
Exec-Code f. DOES @@DODOE
Exec-Code f. Variablen @@DOVAR
FORTH DEFINITIONS
CODE UP
Exec-Code f. Konstanten @@DOCON
Exec-Code f. Uservariablen @@DOUSE
HIDDEN DEFINITIONS
CODE (LIT)

\ Run Time Code for Control Structures
CODE BRANCH
CODE ?BRANCH
CODE (LOOP)
CODE (+LOOP)
```

```
CODE (DO)
CODE (?DO)

\ Execution Control
FORTH DEFINITIONS
CODE >NEXT
CODE EXECUTE
Exec-Code f. Vektoren @@DOVEC
Exec-Code f. User-Vektoren @@DOUVEC
CODE NOOP
CODE PAUSE
CODE I
CODE J
HIDDEN DEFINITIONS
CODE (LEAVE)
CODE (?LEAVE)

\ 16 and 8 bit Memory Operations
FORTH DEFINITIONS
CODE @
CODE !
CODE C@
CODE C!

\ Block Move Memory Operations
CODE CMOVE
CODE CMOVE>

\ 16 bit Stack Operations
CODE SP@
CODE SP!
CODE RP@
CODE RP!
CODE DROP
CODE DUP
CODE SWAP
CODE OVER
CODE TUCK
CODE NIP
CODE ROT
CODE -ROT
CODE ?DUP
CODE >R
CODE R>
CODE R@
CODE PICK
CODE ROLL

\ 16 bit Logical Operations
CODE AND
CODE OR
```

```
CODE XOR
CODE NOT
0 CONSTANT FALSE
-1 CONSTANT TRUE

\ Logical Operations
CODE CSET
CODE CRESET
CODE CTOGGLE
CODE OFF
CODE ON

\ 16 bit Arithmetic Operations
CODE +
CODE -
CODE NEGATE
CODE ABS
CODE +!
0 CONSTANT 0
1 CONSTANT 1
2 CONSTANT 2
3 CONSTANT 3
CODE 2*
CODE 2/
CODE U2/
CODE 1+
CODE 2+
CODE 1-
CODE 2-
CODE UM*
CODE UM/MOD

\ 16 bit Comparison Operations
CODE 0=
CODE 0<
CODE 0>
CODE =
CODE U<
CODE <
CODE >
: ?NEGATE   0< IF NEGATE THEN ;
: MIN   ( n1 n2 -- n3 )   OVER OVER > IF SWAP THEN DROP ;
: MAX    ( n1 n2 -- n3 )   OVER OVER < IF SWAP THEN DROP ;
: BETWEEN   >R OVER > SWAP R> > OR NOT ;

\ 32 bit Memory Operations
CODE 2@
CODE 2!

\ 32 bit Memory and Stack Operations
CODE 2DROP
```

```
CODE 2DUP
CODE 2SWAP
CODE 2OVER
: 2ROT   5 ROLL 5 ROLL ;

\ 32 bit Arithmetic Operations
CODE D+
CODE DNEGATE
CODE S>D
CODE DABS
CODE D2/
: D-   DNEGATE D+ ;
: ?DNEGATE   0< IF DNEGATE THEN ;

\ 32 bit Comparison Operations
: D0=   OR 0= ;
: D=   D- D0= ;
: DU<   ROT SWAP 2DUP U< IF 2DROP 2DROP TRUE
   ELSE = IF U< ELSE 2DROP FALSE THEN THEN ;
: D<   2 PICK OVER = IF DU< ELSE NIP ROT DROP < THEN ;
: D>   2SWAP D< ;
: DMIN   2OVER 2OVER D> IF 2SWAP THEN 2DROP ;
: DMAX   2OVER 2OVER D< IF 2SWAP THEN 2DROP ;

\ Mixed Mode Arithmetic
: *D   2DUP XOR >R ABS SWAP ABS UM* R> ?DNEGATE ;
: M/MOD    DUP >R 2DUP XOR >R >R DABS R@ ABS UM/MOD SWAP R> ?NEGATE SWAP R>
   0< IF NEGATE OVER IF 1- R@ ROT - SWAP THEN THEN R> DROP ;
: MU/MOD   >R 0 R@ UM/MOD R> SWAP >R UM/MOD R> ;

\ 16 bit multiply and divide
: *   UM* DROP ;
: /MOD   >R S>D R> M/MOD ;
: /   /MOD NIP ;
: MOD   /MOD DROP ;
: */MOD   >R *D R> M/MOD ;
: */   */MOD NIP ;

\ Task Dependant USER Variables
USER VARIABLE TOS
USER VARIABLE ENTRY
USER VARIABLE LINK
USER VARIABLE SP0
USER VARIABLE RP0
USER VARIABLE DP
USER VARIABLE #OUT
USER VARIABLE #LINE
USER VARIABLE OFFSET
USER VARIABLE BASE
USER VARIABLE HLD
USER VARIABLE PRINTING
```

```
\ System Variables
USER DEFER EMIT   ' (EMIT) IS EMIT
USER DEFER CR   ' (CR) IS CR
VARIABLE FENCE
VARIABLE SCR
VARIABLE PRIOR
VARIABLE STATE
VARIABLE DPL
VARIABLE R#
VARIABLE LAST
VARIABLE CSP
VARIABLE CURRENT
8 CONSTANT #VOCS
VARIABLE  CONTEXT   HERE #VOCS 2* DUP ALLOT ERASE
VARIABLE 'TIB
VARIABLE WIDTH
VARIABLE WARNING
VARIABLE VOC-LINK
VARIABLE BLK
VARIABLE >IN
VARIABLE SPAN
VARIABLE #TIB

\ Strings
32 CONSTANT BL
8 CONSTANT BS
7 CONSTANT BELL
: HERE   DP @ ;
: PAD   HERE 80 + ;
CODE FILL
: ERASE   0 FILL ;
: BLANK   BL FILL ;
: MOVE   -ROT 2DUP U< IF ROT CMOVE> ELSE ROT CMOVE THEN ;
CODE COUNT
: TYPE   0 ?DO COUNT EMIT LOOP DROP ;
: SPACE   BL EMIT ;
: SPACES   0 MAX 0 ?DO SPACE LOOP ;
: -TRAILING   DUP 0 DO 2DUP + 1- C@ BL = NOT ?LEAVE 1- LOOP ;
CODE COMPARE
DEFER KEY   ' (KEY) IS KEY
DEFER KEY?   ' (KEY?) IS KEY?

\ Devices Terminal IO
VOCABULARY I/O
I/O DEFINITIONS
CODE OS
: (KEY?)   6 OS 0= NOT ;
: (KEY)   BEGIN PAUSE 0 OS ?DUP UNTIL ;
: (CONSOLE)   PAUSE 3 OS 1 #OUT +! ;
: (PRINT)   PAUSE 9 OS 1 #OUT +! ;
: (EMIT)   PRINTING @ IF DUP (PRINT) -1 #OUT +! THEN (CONSOLE) ;
```

```
HEX : CRLF   0D EMIT 0A EMIT #OUT OFF 1 #LINE +! ; DECIMAL
FORTH DEFINITIONS
DEFER EXPECT   ' (EXPECT) IS EXPECT
HIDDEN DEFINITIONS
: DEL-IN   DROP DUP IF 1- BS EMIT SPACE BS ELSE BELL THEN EMIT ;
: CR-IN   DROP SPAN ! OVER BL EMIT ;
: CHAR   >R R@ EMIT 2DUP + R> SWAP C! 1+ ;
: (EXPECT)   DUP SPAN ! SWAP 0 BEGIN 2 PICK OVER -
   WHILE KEY DUP 0D = IF CR-IN ELSE DUP 7F = OVER BS = OR
   IF DEL-IN ELSE CHAR THEN THEN REPEAT 2DROP DROP ;
FORTH DEFINITIONS
: TIB   'TIB @ ;
: QUERY   TIB 80 EXPECT SPAN @ #TIB ! BLK OFF >IN OFF ;

\ BLOCK I/O
512 CONSTANT B/BUF
0 CONSTANT FIRST    ( patched on COLD. FIRST := @@init_top - B/BUF)

HIDDEN DEFINITIONS
VARIABLE UPD
VARIABLE BLK#

FORTH DEFINITIONS
DEFER READ-BLOCK   ' READ IS READ-BLOCK
DEFER WRITE-BLOCK   ' WRITE IS WRITE-BLOCK
: EMPTY-BUFFERS   FIRST B/BUF ERASE UPD OFF BLK# ON ;
: SAVE-BUFFERS   UPD @ IF FIRST BLK# @ WRITE-BLOCK UPD OFF THEN ;
: FLUSH   SAVE-BUFFERS EMPTY-BUFFERS ;

HIDDEN DEFINITIONS
: ABSENT?   DUP BLK# @ = NOT ;

FORTH DEFINITIONS
: UPDATE UPD ON ;
: BUFFER   PAUSE OFFSET @ + ABSENT? IF SAVE-BUFFERS THEN BLK# ! FIRST ;
: BLOCK   PAUSE OFFSET @ + ABSENT?
   IF SAVE-BUFFERS FIRST OVER READ-BLOCK THEN BLK# ! FIRST ;

I/O DEFINITIONS
@@init_sod CONSTANT SOD
@@init_eod CONSTANT EOD
: MAX#   EOD @ SOD @ - B/BUF / ;
: 'RAM   1- MAX# 1- OVER U< ABORT" out of Disk " B/BUF * SOD @ + ;
: READ   'RAM SWAP B/BUF CMOVE ;
: WRITE   'RAM B/BUF CMOVE ;

\ Number Input
FORTH DEFINITIONS
: ?MISSING   IF HERE COUNT TYPE TRUE ABORT" ?" THEN ;
CODE DIGIT
: DOUBLE?   DPL @ 1+ 0= NOT ;
```

```
: CONVERT   BEGIN 1+ DUP >R C@ BASE @ DIGIT
   WHILE SWAP BASE @ UM* DROP ROT BASE @ UM* D+
   DOUBLE? IF 1 DPL +! THEN R> REPEAT DROP R> ;
HIDDEN DEFINITIONS
: (NUMBER?)   0 0 ROT DUP 1+ C@ ASCII - = DUP >R - -1 DPL !
  BEGIN CONVERT DUP C@ ASCII , ASCII / BETWEEN WHILE 0 DPL ! REPEAT
  -ROT R> IF DNEGATE THEN ROT C@ BL = ;
: INUMBER?   FALSE OVER COUNT OVER + SWAP
   DO I C@ BASE @ DIGIT NIP IF DROP TRUE LEAVE THEN LOOP
   IF (NUMBER?) ELSE DROP 0 0 FALSE THEN ;
FORTH DEFINITIONS
DEFER NUMBER?  ' INUMBER? IS NUMBER?

\ Number Output
: HOLD   -1 HLD +! HLD @ C! ;
: SIGN   0< IF ASCII - HOLD THEN ;
: <#   PAD HLD ! ;
: #>   2DROP HLD @ PAD OVER - ;
: #   BASE @ MU/MOD ROT 9 OVER < IF 7 + THEN ASCII 0 + HOLD ;
: #S   BEGIN # 2DUP OR 0= UNTIL ;
: HEX   16 BASE ! ;
: DECIMAL   10 BASE ! ;
: (U.)   0 <# #S #> ;
: U.   (U.) TYPE SPACE ;
: U.R   >R (U.) R> OVER - SPACES TYPE ;
: (.)   DUP ABS 0 <# #S ROT SIGN #> ;
: .   (.) TYPE SPACE ;
: .R   >R (.) R> OVER - SPACES TYPE ;
: (D.)   TUCK DABS <# #S ROT SIGN #> ;
: D.   (D.) TYPE SPACE ;
: D.R   >R (D.) R> OVER - SPACES TYPE ;

\ Parsing
CODE SKIP
CODE SCAN
: /STRING   OVER MIN ROT OVER + -ROT - ;
: PLACE   2DUP >R >R 1+ SWAP MOVE R> R> C! ;
: SOURCE   BLK @ ?DUP IF BLOCK B/BUF ELSE TIB #TIB @ THEN ;
: PARSE-WORD   >R SOURCE >IN @ /STRING OVER SWAP R@ SKIP
   OVER SWAP R> SCAN DROP 2DUP SWAP - >R ROT - 1+ >IN +! R> ;
: PARSE   >R SOURCE >IN @ /STRING OVER SWAP R> SCAN DROP OVER - DUP 1+ >IN
+! ;
: WORD   PARSE-WORD HERE PLACE HERE BL OVER COUNT + C! ;
: >TYPE   TUCK PAD SWAP CMOVE PAD SWAP TYPE ;
: .(   ASCII ) PARSE >TYPE ; IMMEDIATE
: (   ASCII ) PARSE 2DROP ; IMMEDIATE

\ Dictionary
CODE HASH
HIDDEN DEFINITIONS
CODE (FIND)
```

```
FORTH DEFINITIONS
: FIND   PRIOR OFF FALSE #VOCS 0 DO
   DROP CONTEXT I 2* + @ DUP
   IF DUP PRIOR @ OVER PRIOR ! =
   IF DROP FALSE ELSE OVER SWAP HASH @ (FIND) DUP ?LEAVE THEN THEN LOOP ;
: DEFINED   BL WORD FIND ;

\ Interpreter
DEFER STATUS   ' CR IS STATUS
: ?STACK   SP@ SP0 @ SWAP U< ABORT" stack underflow"
   SP@ PAD U< ABORT" stack overflow" ;
HIDDEN DEFINITIONS
DEFER HANDLE    ' (INTERPRET) IS HANDLE
: (INTERPRET)   FIND IF EXECUTE ELSE NUMBER? NOT ?MISSING DOUBLE? NOT IF
DROP THEN THEN ;
FORTH DEFINITIONS
: INTERPRET   BEGIN ?STACK BL WORD DUP C@ WHILE HANDLE REPEAT DROP ;

\ Compiler
: ALLOT   DP +! ;
: ,   HERE ! 2 ALLOT ;
: C,   HERE C! 1 ALLOT ;
: COMPILE   R> DUP 2+ >R @ , ;
HEX : IMMEDIATE   40 ( Precedence bit) LAST @ CSET ; DECIMAL
: LITERAL   COMPILE (LIT) , ; IMMEDIATE
: DLITERAL   SWAP [COMPILE] LITERAL [COMPILE] LITERAL ; IMMEDIATE
: '   DEFINED 0= ?MISSING ;
: [']   ' [COMPILE] LITERAL ; IMMEDIATE
: [COMPILE]   ' , ; IMMEDIATE
HIDDEN DEFINITIONS
: (")   R> COUNT 2DUP + >R ;
: (.")   R> COUNT 2DUP + >R TYPE ;
: ,"   ASCII " PARSE TUCK HERE PLACE 1+ ALLOT ;
FORTH DEFINITIONS
: ."   COMPILE (.") ," ; IMMEDIATE
: "   COMPILE (") ," ; IMMEDIATE

\ Compiler
DEFER WHERE   ' NOOP IS WHERE
DEFER ?ERROR   ' (?ERROR) IS ?ERROR
HIDDEN DEFINITIONS
: (ABORT")   R@ COUNT ROT ?ERROR R> COUNT + >R ;
FORTH DEFINITIONS
: ABORT"   COMPILE (ABORT") ," ; IMMEDIATE

\ Defining Words
: !CSP   SP@ CSP ! ;
: ?CSP   SP@ CSP @ = NOT (ABORT") compilation uncorrect" ;

\ Dictionary
: N>LINK   2- ;
```

```
: L>NAME   2+ ;
: BODY>   2- ;
CODE NAME>
: LINK>   L>NAME NAME> ;
: >BODY   2+ ;
CODE >NAME
: >LINK   >NAME N>LINK ;

\ Defining Words
: HIDE   LAST @ DUP N>LINK @ SWAP CURRENT @ HASH ! ;
: REVEAL LAST @ DUP N>LINK SWAP CURRENT @ HASH ! ;
HIDDEN DEFINITIONS
HEX : HEADER   HERE 0 , HERE LAST ! DEFINED WARNING @ AND
  IF HERE COUNT TYPE (.")  is redefined " THEN
  DROP HERE CURRENT @ HASH DUP @ HERE 2- ROT ! SWAP !
  HERE DUP C@ WIDTH @ MIN 1+ ALLOT 80 SWAP CSET 80 HERE 1- CSET ; DECIMAL
FORTH DEFINITIONS
: CREATE HEADER @@dovar , ;
HIDDEN DEFINITIONS
: (COMPILE)   FIND ?DUP IF 0> IF EXECUTE ELSE , THEN
   ELSE NUMBER? NOT ?MISSING DOUBLE? IF DLITERAL ELSE DROP LITERAL THEN THEN
;
FORTH DEFINITIONS
: [   STATE OFF ['] (INTERPRET) (IS) HANDLE ; IMMEDIATE
: ]   STATE ON ['] (COMPILE) (IS) HANDLE ;
VOCABULARY ASSEMBLER
HIDDEN DEFINITIONS
: (;USES)   R> @ LAST @ NAME> ! ;
: (;CODE)   R> LAST @ NAME> ! ;
FORTH DEFINITIONS
: ;USES   ?CSP COMPILE (;USES) [COMPILE] [ REVEAL ASSEMBLER ; IMMEDIATE
: ;CODE   ?CSP COMPILE (;CODE) [COMPILE] [ REVEAL ASSEMBLER ; IMMEDIATE
@@dodoe CONSTANT >DOES
HEX : DOES> COMPILE (;CODE) 0CD ( call) C, >DOES , ; IMMEDIATE DECIMAL
: !CSP   CURRENT @ CONTEXT ! CREATE HIDE ] ;USES @@docol ,
: ;   ?CSP COMPILE UNNEST REVEAL [COMPILE] [ ; IMMEDIATE
: RECURSE   LAST @ NAME> , ; IMMEDIATE
: CONSTANT CREATE , ;USES @@docon ,
: VARIABLE CREATE 0 , ;USES @@dovar ,
: 2CONSTANT   CREATE , , DOES> 2@ ; DROP
: 2VARIABLE   0 0 2CONSTANT DOES> ; DROP
: CRASH TRUE ABORT" undefined execution vector" ;
: DEFER CREATE ['] CRASH , ;USES @@dovec ,

\ Dictionary
4 CONSTANT #THREADS
HIDDEN DEFINITIONS
: TRIM   #THREADS 0 DO 2DUP @ BEGIN 2DUP SWAP U< NOT WHILE @ REPEAT
   NIP OVER ! 2+ LOOP 2DROP ;
: (FORGET)   DUP FENCE @ U< ABORT" protected "
   DUP VOC-LINK @ BEGIN 2DUP U< WHILE @ REPEAT
```

```
   DUP VOC-LINK ! NIP
   BEGIN DUP WHILE 2DUP #THREADS 2* - TRIM @ REPEAT DROP DP ! ;
FORTH DEFINITIONS
: FORGET   BL WORD DUP CURRENT @ HASH @ (FIND) 0= ?MISSING >LINK (FORGET) ;
VARIABLE AVOC
: CODE   CREATE HIDE HERE DUP 2- ! CONTEXT @ AVOC ! ASSEMBLER ;
: LABEL   CREATE HIDE CONTEXT @ AVOC ! ASSEMBLER ;
ASSEMBLER DEFINITIONS
: END-CODE   AVOC @ CONTEXT ! REVEAL ;
FORTH DEFINITIONS
: VOCABULARY   CREATE #THREADS 0 DO 0 , LOOP
   HERE VOC-LINK @ , VOC-LINK ! DOES> CONTEXT ! ;
: DEFINITIONS CONTEXT @ CURRENT ! ;

\ USER Defining Words
VARIABLE #USER
VOCABULARY USER
USER DEFINITIONS
: ALLOT   DP +! ;
: CREATE   CREATE #USER @ , ;USES @@douse ,
: VARIABLE   CREATE 2 ALLOT ;
: DEFER   VARIABLE ;USES @@douvec ,

\ Initialization
FORTH DEFINITIONS
: QUIT  BLK OFF [COMPILE] [
    BEGIN RP0 @ RP! STATUS QUERY INTERPRET
    STATE @ NOT IF ." ok" THEN AGAIN ;

\ ReDefining Words
: >IS   DUP @ DUP @@douse = SWAP DUP @@douvec = SWAP DROP OR
    IF >BODY @ UP @ + ELSE >BODY THEN ;
: (IS)   R@ @ >IS ! R> 2+ >R ;
: IS   STATE @ IF COMPILE (IS) ELSE ' >IS ! THEN ; IMMEDIATE

\ Compiler
HIDDEN DEFINITIONS
: (?ERROR)   IF >R >R SP0 @ SP! PRINTING OFF
    BLK @ IF >IN @ BLK @ WHERE THEN
    R> R> SPACE TYPE SPACE QUIT ELSE 2DROP THEN ;
FORTH DEFINITIONS
: ABORT   SP0 @ SP! QUIT ;
: FORTH-83 ;

\ Initialization
: .VERSION   CR ." Z80 FORTH 83  V1.3" CR ." im Auftrag der AIG Forth" CR
    ." Th. Beierlein  (Juli 88)" CR ;
DEFER BOOT  ' HELLO IS BOOT
: COLD   BOOT ABORT ;
: WARM   TRUE ABORT" Warmstart" ;
warm start
```

```
cold start
: BYE   SAVE-BUFFERS 12 OS ;

\ Structures
: ?CONDITION   = NOT ABORT" Conditionals wrong" ;
: >MARK   HERE 0 , ;
: <MARK   HERE ;
: >RESOLVE   HERE SWAP ! ;
: <RESOLVE   , ;
: ?>RESOLVE   ?CONDITION >RESOLVE ;
: ?<RESOLVE   ?CONDITION <RESOLVE ;
: LEAVE   COMPILE (LEAVE) ; IMMEDIATE
: ?LEAVE   COMPILE (?LEAVE) ; IMMEDIATE
: BEGIN   <MARK TRUE ; IMMEDIATE
: THEN   FALSE ?>RESOLVE ; IMMEDIATE
: DO   COMPILE (DO) >MARK 1 ; IMMEDIATE
: ?DO   COMPILE (?DO) >MARK 1 ; IMMEDIATE
: LOOP   COMPILE (LOOP) OVER 2+ OVER 1 ?<RESOLVE 1 ?>RESOLVE ; IMMEDIATE
: +LOOP   COMPILE (+LOOP) OVER 2+ OVER 1 ?<RESOLVE 1 ?>RESOLVE ; IMMEDIATE
: UNTIL   COMPILE ?BRANCH TRUE ?<RESOLVE ; IMMEDIATE
: AGAIN   COMPILE BRANCH TRUE ?<RESOLVE ; IMMEDIATE
: REPEAT   2SWAP [COMPILE] AGAIN [COMPILE] THEN ; IMMEDIATE
: IF   COMPILE ?BRANCH >MARK FALSE ; IMMEDIATE
: ELSE   COMPILE BRANCH >MARK FALSE SWAP FALSE ?>RESOLVE ; IMMEDIATE
: WHILE   [COMPILE] IF ; IMMEDIATE

\ Resident Tools
: ?   @ . ;
: ASCII   BL WORD 1+ C@ STATE @ IF [COMPILE] LITERAL THEN ; IMMEDIATE
: CONTROL   BL WORD 1+ C@ 31 AND STATE @ IF [COMPILE] LITERAL THEN ;
IMMEDIATE
: DEPTH   SP@ SP0 @ SWAP - 2/ ;
: .S   CR DEPTH ?DUP IF 0 DO DEPTH I - 1- PICK 7 U.R SPACE LOOP
   ELSE ." empty " THEN ;
: ?ENOUGH   DEPTH 1- > (ABORT") not enough parameters" ;

\ Output Formatting
VARIABLE RMARGIN
( Z9001) 36 RMARGIN !
: ?LINE   #OUT @ + RMARGIN @ > IF CR THEN ;

HEX : .ID   DUP 1+ DUP C@ ROT C@ 1F AND 0
  ?DO DUP 7F AND EMIT 80 AND IF 0DFh ( ASCII _ 80 OR) ELSE 1+ DUP C@ THEN
  LOOP 2DROP SPACE ; DECIMAL

\ Display the WORDS in the Context Vocabulary
: LARGEST   OVER 0 SWAP ROT 0
   DO 2DUP @ U< IF -ROT 2DROP DUP @ OVER THEN 2+ LOOP DROP ;
: WORDS   CR CONTEXT @ HERE #THREADS 2* CMOVE
  BEGIN HERE #THREADS LARGEST DUP
  WHILE DUP L>NAME DUP C@ 31 AND ?LINE .ID SPACE SPACE @ SWAP !
```

```
  KEY? IF EXIT THEN REPEAT 2DROP ;

\ Commenting and Loading Words
32 CONSTANT C/L
16 CONSTANT L/SCR
: -->   >IN OFF 1 BLK +! ; IMMEDIATE
: LOAD   BLK @ >R >IN @ >R >IN OFF BLK ! INTERPRET R> >IN ! R> BLK ! ;
: THRU   2 ?ENOUGH 1+ SWAP DO I LOAD LOOP ;
: LIST   1 ?ENOUGH CR DUP SCR ! ." Scr # " DUP . L/SCR 0
   DO CR I 3 .R SPACE DUP BLOCK I C/L * + C/L -TRAILING >TYPE
   KEY? ?LEAVE LOOP DROP CR ;
: INDEX   2 ?ENOUGH 1+ SWAP
   DO CR I 3 .R SPACE I BLOCK C/L -TRAILING >TYPE
   KEY? ?LEAVE LOOP CR ;
: \   >IN @ NEGATE C/L MOD >IN +! ; IMMEDIATE
: \S   B/BUF >IN ! ; IMMEDIATE
: FH   BLK @ + ;
: COPY   2 ?ENOUGH SWAP BLOCK DROP BLK# ! UPDATE SAVE-BUFFERS ;

\ Port Access
CODE P@
CODE P!

\ The ALSO and ONLY Concept
CONTEXT DUP @ SWAP 2+ !   ( Make FORTH also )
VOCABULARY ROOT
ROOT DEFINITIONS
: ALSO   CONTEXT DUP 2+ #VOCS 2- 2* CMOVE> ;
: ONLY   ['] ROOT >BODY CONTEXT #VOCS 1- 2* 2DUP ERASE + ! ROOT ;
: SEAL   ' >BODY CONTEXT #VOCS 2* ERASE CONTEXT ! ;
: PREVIOUS   CONTEXT DUP 2+ SWAP #VOCS 2- 2* CMOVE
   CONTEXT #VOCS 2- 2* + OFF ;
: FORTH   FORTH ;
: DEFINITIONS   DEFINITIONS ;
: ORDER   CR ." Context: " CONTEXT #VOCS
   0 DO DUP @ ?DUP IF BODY> >NAME .ID THEN 2+ LOOP DROP
   CR ." Current: " CURRENT @ BODY> >NAME .ID ;
: VOCS   ." : " VOC-LINK @ BEGIN DUP #THREADS 2* - BODY> >NAME .ID
   @ DUP 0= UNTIL DROP ;
: WORDS   WORDS ;

\ Load up the system
FORTH DEFINITIONS
: HELLO   EMPTY-BUFFERS ONLY FORTH ALSO DEFINITIONS DECIMAL .PROCLAIM ;

\ hex dump
VARIABLE /LINE
( z1013) 4 /LINE !
: .2   0 <# # # #> TYPE SPACE ;
HEX : DUMP   BASE @ >R HEX CR CR
   5 SPACES /LINE @ 0 DO I 3 .R LOOP
```

```
   2 SPACES /LINE @ 0 DO I 1 .R LOOP CR
   OVER + SWAP /LINE @ NEGATE AND
   DO CR I 4 U.R SPACE SPACE I /LINE @ + I 2DUP
     DO I C@ .2 LOOP SPACE
     DO I C@ DUP BL 7E BETWEEN NOT IF DROP ASCII . THEN EMIT LOOP
     KEY? ?LEAVE /LINE @ +LOOP CR R> BASE ! ; DECIMAL
HEX : DU   DUP 3F DUMP 40 + ; DECIMAL

\ placeholder file i/o
DEFER PUT  ' CRASH IS PUT
DEFER GET  ' CRASH IS GET

: MARK   CREATE DOES> (FORGET) FORTH DEFINITIONS ;


( ab hier individuelle Anpassung an Z9001 )

VARIABLE LMARGIN   8 LMARGIN !
I/O DEFINITIONS
: (CR)   CRLF PRINTING @ IF LMARGIN @ 0 DO BL (PRINT) LOOP THEN ;
' (CR) IS CR
FORTH DEFINITIONS
: .PROCLAIM   12 EMIT  ." Z80 FORTH 1.3 (Z9001)" CR  ." AG FORTH" CR
." Th. Beierlein / "  ." V. Pohlers" CR ;
MARK EMPTY  HERE FENCE !
```