```
\ VFXTESTAPP -  Set up a folder c:\VFXTESTAPP. Store file VFXTESTAPP.fs there. ( INCLUDE c:\VFXTESTAPP\VFXTESTAPP.fs ) copy text
\ in brackets into VFX window, hit <cr>  to load the demo.  VFX reads in the file, compiles new words where needed. SOS should
\ be heard, then a simple "GUI" of few lines shows and the 4 OUT bits are counted up.  <cr> stops this and answers ok. Now set
\ bits, e.g. with PWL or PWH <cr> or the others; try NAME <cr>, type SOS <cr>, COUNTER <cr>; <cr> gets back into command mode;
\ or 400 SCOUNTER <cr> the 400 setting speed; or test the very basic debugger with ???? <cr>. VFXTESTAPP ExMark 2016_10_02 V3a

HEX \ from now on all of the numbers are understood to be hex. The LINES n print a small GUI, very basic, just uses PAGE and CR
: LINE1 ." \ VFXTESTAPP - bit change ie. PWH / PWL <cr>, try COUNTER, 400 SCOUNTER, SOS , ???? plus <cr> ExMark Oct2016" CR ;
: LINE2 ." \ PWHL T3HL T2HL T1HL O3HL O2HL O1HL O0HL I3HL I2HL I1HL I0HL A3Hl A2HL A1HL A0HL h/l ???? " CR ;
: LINE3 ." \ PWM  T3___T2___T1   O3___O2___O1___O0   I3___I2___I1___I0   A3___A2___A1___A0  "  CR ;
: LINE4 ."    X    t    t    t    o    o    o    o    i    i    i    i    a    a    a    a  "  CR ;
: TEST  PAGE LINE1 LINE2 LINE3  LINE4  ( LINE5 ) CR ;

Variable PSWI   ( PSWI X t t t   4 bits PWM and Switches )
Variable OUTP   ( OUTP 3 2 1 0   7 6 5 4 of the 8 bits   )
Variable IN     ( IN   3 2 1 0   3 2 1 0 of the 8 Bits   )
Variable ANI    ( ANI  3 2 1 0   4 more bits for test    )

: disbit4  DUP $8 AND IF ." 1" ELSE ." 0" THEN  ;
: ds       disbit4 1 LSHIFT ;
: dssp     ds Space Space Space Space ;
: 4dssp    dssp dssp dssp disbit4  drop ;
: DV       3 Spaces PSWI @ 4dssp  4 Spaces OUTP @ 4dssp 4 Spaces IN @ 4dssp  4 Spaces  ANI @ 4dssp  ;
: SPACES   ( u -- ) 0 ?DO SPACE LOOP ;
: MBV2     PAGE LINE1 LINE2 LINE3 LINE4 DV CR ;
: COUNTER  Begin outp @ 1+ outp !     300 ms mbv2 key? until ;
: SCOUNTER Begin dup outp @ 1+ outp !     ms mbv2 key? until ;
: SOS 07 emit 100 ms 07 emit 100 ms 07 emit 600 ms 07  emit 300 ms 07 emit 300 ms 07 emit 600 ms 07 emit 100 ms 07 emit 100 ms 07 emit ;

: LINE5  ." PSWI  OUT  IN   ANI    - Stack contents "  CR ;
: ????   3 Spaces PSWI @ . OUTP @ . IN @ . ANI @ . 4 Spaces >R >R >R >R >R >R >R >R R> DUP . R> DUP . R> DUP . R> DUP . R> DUP . R> DUP .  R> DUP . R>
DUP . ;  \ ???? displays PSWI OUTP IN ANI and DSTACK: 8 levels. Type in FFFF first to mark the bottom of the Data Stack

\ Preset Variables:
9 PSWI !
0 OUTP !
3 IN   !
F ANI  !

\ Control Words  The long version of this sandbox, more explanations you find at http://wiki.forth-ev.de/doku.php/en:projects:a-start-with-forth:start
: PWH   PSWI @  $8 OR   PSWI !  MBV2 ;
: T3H   PSWI @  $4 OR   PSWI !  MBV2 ;
: T2H   PSWI @  $2 OR   PSWI !  MBV2 ;
: T1H   PSWI @  $1 OR   PSWI !  MBV2 ;
: O3H   OUTP @  $8 OR   OUTP !  MBV2 ;
: O2H   OUTP @  $4 OR   OUTP !  MBV2 ;
: O1H   OUTP @  $2 OR   OUTP !  MBV2 ;
: O0H   OUTP @  $1 OR   OUTP !  MBV2 ;
: I3H   IN   @  $8 OR   IN   !  MBV2 ;
: I2H   IN   @  $4 OR   IN   !  MBV2 ;
: I1H   IN   @  $2 OR   IN   !  MBV2 ;
: I0H   IN   @  $1 OR   IN   !  MBV2 ;
: A3H   ANI  @  $8 OR   ANI  !  MBV2 ;
```

```
: A2H   ANI  @  $4 OR   ANI  !  MBV2 ;
: A1H   ANI  @  $2 OR   ANI  !  MBV2 ;
: A0H   ANI  @  $1 OR   ANI  !  MBV2 ;
: PWL   PSWI @  $7 AND  PSWI !  MBV2 ;
: T3L   PSWI @  $B AND  PSWI !  MBV2 ;
: T2L   PSWI @  $D AND  PSWI !  MBV2 ;
: T1L   PSWI @  $E AND  PSWI !  MBV2 ;
: O3L   OUTP @  $7 AND  OUTP !  MBV2 ;
: O2L   OUTP @  $B AND  OUTP !  MBV2 ;
: O1L   OUTP @  $D AND  OUTP !  MBV2 ;
: O0L   OUTP @  $E AND  OUTP !  MBV2 ;
: I3L   IN   @  $7 AND  IN   !  MBV2 ;
: I2L   IN   @  $B AND  IN   !  MBV2 ;
: I1L   IN   @  $D AND  IN   !  MBV2 ;
: I0L   IN   @  $E AND  IN   !  MBV2 ;
: A3L   ANI  @  $7 AND  ANI  !  MBV2 ;
: A2L   ANI  @  $B AND  ANI  !  MBV2 ;
: A1L   ANI  @  $D AND  ANI  !  MBV2 ;
: A0L   ANI  @  $E AND  ANI  !  MBV2 ;

\ Set I1 and/or I0 of the INPUTs, then call AND01, OR01, XOR01, INVERT0
: AND01    IN @  DUP 1 RSHIFT AND 01 AND  OUTP !  MBV2  ;
: OR01     IN @  DUP 1 RSHIFT OR  01 AND  OUTP !  MBV2  ;
: XOR01    IN @  DUP 1 RSHIFT XOR 01 AND  OUTP !  MBV2  ;
: INVERT0  IN @  INVERT           01 AND  OUTP !  MBV2  ;

\ Forth Words used
\  0 INCLUDE   \  1 HEX      \  2 \        \  3 :        \  4 ."       \  5 CR        \  6 ;         \  7 TEST
\  8 Variable  \  9 DUP      \ 10 $n       \ 11 AND      \ 12 IF       \ 13 ELSE      \ 14 THEN      \ 15 LSHIFT
\ 16 SPACE     \ 17 DROP     \ 18 DV       \ 19 @        \ 20 ?DO      \ 21 LOOP      \ 22 PAGE      \ 23 BEGIN
\ 24 1+        \ 25 !        \ 26 MS       \ 27 KEY?     \ 28 UNTIL    \ 29 EMIT      \ 30 .S        \ 31 .
\ 32 >R        \ 33 R>       \ 34 ( and )  \ ------ surprisingly this covers all of the words needed for this sandbox

\ Now start running Forth code
: Name  ." Hello Forth World " ;
Page CR Name  1000 ms
sos
counter
\ This is not programmed optimally  - but this was not the target -  beginner's code for beginners - easy to understand and explain.

\ The Forth Machine:  Data stream coming in, THE STACK, Return Stack, Variables in memory, other memory, route to screen.
\                     Digital_In and Digital_Out are for later, the same applies to Memory
\ The Forth Machine looks very complicated - but people actually use a similar model every day at the desk, work coming in, execute, results out
\
\    DIGITAL_IN                        TOP_OF_STACK   TOP_OF_RS                 x7       DIGITAL_OUT
\   ( for later )                      DS-1           RS-1                      x6      ( for later )
\                                      DS-2           RS-2                      x5
\                                      DS-3           RS-3                      x4
\                                      DS-4           RS-4           PSWI       x3
\                                      DS-5           RS-5           OUTP       x2
\                                      DS-6           RS-6           IN         x1
\    Token8 T7 T6 T5 T4 T3 T2 T1 T0    FFFF (-7)      FFFF           ANI        x0       Hello Forth World
\
\    Terminal_Input_Buffer_Contents    DATA STACK     RETURN_STACK   VARIABLES   MEMORY   TO_SCREEN                                ExMark October 2016
```