```
\ Write a simple Application just using 35 Forth Words - set IO bits, start a counter -- ExMark - Juergen Pintaske Oct. 2016
\ There is a Word for SOS as well, but dependent on the hardware you can hear the SOS as some sounds (or not) just try
\ VFXTESTAPP - A Demo running on the PC. Set up a folder c:\VFXTESTAPP. Store this file VFXTESTAPP.f into this folder
\ (INCLUDE c:\VFXTESTAPP\VFXTESTAPP.f) copy the text in brackets into the VFX window, and hit <cr> to run the demo.
\ VFX reads in the file, VFX 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 PWL <cr> or the others.
\ or try NAME <cr>, type SOS <cr>, or COUNTER <cr>; <cr> to get back into command mode; or 400 SCOUNTER <cr> where 400 sets speed
\ Try the very basic debugger with ???? <cr>.
\ Description: The implementation of a virtual MicroBox, which consists of a Micro, PWM LED, 3 Switches, 4 OUT LEDs, 4 IN LEDs
\ Here some functionality implemented in SW: no chip, no LEDs, no resistors, no PCB, no soldering - an Application on PC only
\ The "GUI", the Graphic User Interface implemented using two very simple words - Page to clear screen, <cr> go to next line.
\ LINE1 shows the funtions represented, and below LINE3 the 1bit PWM, 3bits SWITCHES, 4bits OUT, 4bits IN, 4bits A - 16 bits
\ Further down find MBV2, it prints the 16 Bits underneath this GUI. All lines are resent whenever the 16 bits are updated.
\ LINE1 to LINE4 define words to explain the names of the single bits and the control words. LINE4 not used. Modify to show.
\ This has been intentionally written just using simple words and constructs, so anybody can use it as startig point.
\ And as with all Forth programs, this application can be easily modified and extended - and all of this interactively.
\BIN/HEX 0000-0 0001-1 0011-2 0011-3 0100-4 0101-5 0110-6 0111-7 1000-8 1001-9 1010-A 1011-B 1100-C 1101-D 1110-E 1111-F
HEX \ From now on all of the numbers are understood as hexadecimal. The LINES n print a small GUI, very basic, with 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 A3H1 A2HL A1HL A0HL h/1 ???? " CR ;
: LINE3 ." \ PWM T3 T2 T1 O3 O2 O1 O0 I3 I2 I1 I0 A3 A2 A1 A0 "CR;
                             t t
                                                    0 0 0
                                                                     i
                                                                           i i i
                                                                                                                a a " CR; \ display after START?
                                             0
                                                                                                   а
                                                                                                         a
: TEST PAGE LINE1 LINE2 LINE3 LINE4 (LINE5) CR; \ display lines of our GUI Graphic User Interface, LINE5 not used
Variable PWM
                   ( 0 or 1
                                                                              ) \ not used for now, as combined in Variable PSWI
                                                                                        \ not used for now, as combined in Variable PSWI
Variable SWI
                     ( Switches SW3 SW2 SW1
                                                       0 0 0
Variable PSWI ( PSWI X t t t 4 bits PWM and 3 Switches ) \ combines the bit for PWM and the 3 bits for the 3 switches
Variable OUTP ( OUTP 3 2 1 0 7 6 5 4 of the 8 bits
                                                                            ) \ the 4 "OUTPUT LEDs" of the MicroBox
Variable IN (IN 3 2 1 0 3 2 1 0 of the 8 Bits
                                                                               ) \ 4 Input bits that can be set e.g. for AND OR XOR INVERT
                                                                              ) \ either as simulated external ANALOG - or free for anything
Variable ANI
                     (ANI 3 2 1 0 4 more bits for test
                                                                                       \ to be implemented, just showing the 4 OUT bits in hexadecimal
Variable HOU
                     ( OUT as hexadecimal number 0 to F
Variable ALL
                   ( PWM SWI OUT INP ANI
                                                                                       \ To be done, combining it all into one 16 bit Variable
: disbit4 DUP $8 AND IF ." 1" ELSE ." 0" THEN ; \ test a bit, here the topmost bit3 of the 4. If 1 then display 1, else 0
: ds
             Disbit4 1 LSHIFT ;
                                                                     \ display a bit, shift the 4 bits left by one, so bit 2 is the topmost and so on
: dssp
          ds Space Space space ;
                                                                     \ combine a bit with 4 Spaces, to be a block for display
: 4dssp
           dssp dssp dssp disbit4 drop;
                                                                     \ now combine all 4 bits of the Variable nibble, DROP cleans up the stack
              3 spaces PSWI @ 4dssp 4 spaces OUTP @ 4dssp 4 spaces IN @ 4dssp 4 spaces ANI @ 4dssp ; \ combine 4x 4 = 16 bits
: DV
            ( u -- ) 0 ?DO SPACE LOOP ;
                                                                  \ define SPACES, often included in the Forth word set already
: SPACES
: MBV2
              PAGE LINE1 LINE2 LINE3 LINE4 DV CR; \ MBV2 updates the "SCREEN", first the explaining lines, then the 16 Bits
: COUNTER Begin outp @ 1+ outp ! 300 ms mbv2 key? until ; \ run a counter program, see OUT bits change, type counter <cr>
: SCOUNTER Begin dup outp @ 1+ outp ! ms mbv2 key? until ; \ the same with programmable speed, for example 400 scounter <cr>
: 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 - RO R1 - Stack contents
                                                                                                                      " CR; \ description of ???? display
            3 spaces PSWI @ . OUTP @ . IN @ . ANI @ . 4 spaces >R >R >R >R >R >R >R DUP . R> DUP
. R> DUP . R> DUP .; \ ???? displays PSWI OUTP IN ANI and DSTACK 8 levels. Enter FFFF, sets lowest one visible as bottom, so 7 left
: ??? PSWI @ . OUTP @ . IN @ . ANI @ . base @ >r hex .S r> base ! ; \ Debug 2 - prints 4 variables + Data + Stack differently
```

```
\ Preset Variables:
O PWM !
                      not used for now
O SWI !
                      not used for now
           \setminus Set this combined variable to 9 = 1001 this will show the PWM LED ON and T1 ON
9 PSWI!
O OUTP !
           \ All bits LOW 0000
3 IN
           \ The IN variable set to 3 0011 - prepared to try AND OR XOR INVERT
F ANI !
           \ Set to F, so all bits HIGH, free for any use, e.g. program an extra counter, or for other tests
F HOU!
                     not used for now
FFFF ALL !
                     ALL will be the combined 16 Bit Vector PWSI OUT IN ANI later , was commented out here for now
\ Control Words ( from left to right as shown in LINE2 )
: PWH
       PSWI @ $8 OR
                      PSWI ! MBV2 ; \ Get the variable onto stack, set bit 3 using OR, and store it back, now update the Display
       PSWI @ $4 OR
                      PSWI ! MBV2; \ Get the variable onto stack, set bit 2, and store it back, see above
: T3H
: T2H
       PSWI @ $2 OR
                      PSWI ! MBV2; \ And the same for setting all of the other bits
: T1H
       PSWI @ $1 OR
                      PSWI! MBV2;
                      OUTP ! MBV2 ; \ x x x x
: O3H
       OUTP @ $8 OR
                                                 1 0 0 0
                                                 0 1 0 0
: 02H
       OUTP @ $4 OR
                      OUTP! MBV2; \
: 01H
       OUTP @ $2 OR
                      OUTP ! MBV2 ; \
                                                 0 0 1 0
: 00H
       OUTP @ $1 OR
                      OUTP ! MBV2 ; \
                                                 0 0 0 1
: I3H
       IN @ $8 OR
                      IN! MBV2;
: I2H
           @ $4 OR
                          ! MBV2 ;
      IN
                      ΙN
           @ $2 OR
: I1H
      IN
                      ΙN
                          ! MBV2 ;
           @ $1 OR
                          ! MBV2 ;
: IOH
                      ΙN
      IN
: A3H
           @ $8 OR
                      ANI! MBV2;
       ANI
: A2H
       ANI
           @ $4 OR
                      ANI! MBV2;
: A1H
       ANI @ $2 OR
                      ANI! MBV2;
                      ANI ! MBV2; \ here end the words that Set Bits HIGH - the next ones set the same bits LOW using AND
: A0H
       ANI @ $1 OR
: PWL
       PSWI @ $7 AND
                      PSWI! MBV2; \ x x x x
                                                 0 1 1 1
: T3L
       PSWI @
               $B AND
                      PSWI! MBV2;
                                                 1 0 1 1
: T2L
                      PSWI! MBV2;
       PSWI @
              $D AND
                                                 1 1 0 1
: T1L
                      PSWI! MBV2;
                                                 1 1 1 0
       PSWI @
              $E AND
: 03L
       OUTP @ $7 AND
                      OUTP! MBV2;
: 02L
       OUTP @
              $B AND
                      OUTP! MBV2;
                      OUTP! MBV2;
: 01L
       OUTP @ $D AND
: 00L
       OUTP @ $E AND
                      OUTP! MBV2;
: I3L
       IN @ $7 AND
                      IN! MBV2;
: I2L
      IN @ $B AND
                      IN
                          ! MBV2 ;
: I1L
      IN @ $D AND IN
                          ! MBV2 ;
: IOL
       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, INVERTO and see the result of the logic result in OUTO
: AND01
         IN @ DUP 1 RSHIFT AND 01 AND OUTP! MBV2; \ Do an AND of IN bit0 and bit1, 00=10=01=01=01
```

IN @ DUP 1 RSHIFT OR 01 AND OUTP ! MBV2 ; \ Do an OR of IN bit0 and bit1, 00=>0 01=>1 10=>1 11=>1

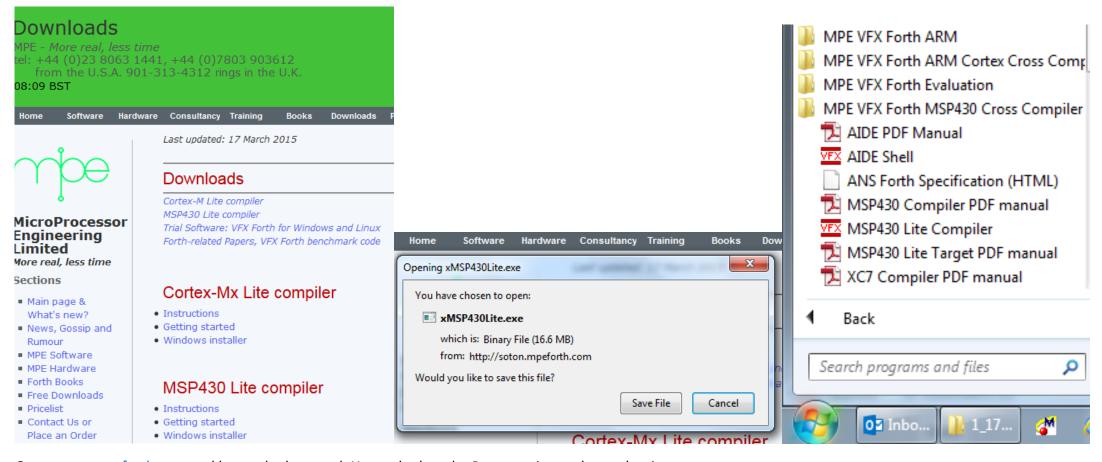
: OR01

```
: XOR01
          IN @ DUP 1 RSHIFT XOR 01 AND OUTP ! MBV2 ; \ Do an XOR of IN bit0 and bit1, 00=>0 01=>1 10=>1 11=>0
: INVERTO IN @ INVERT
                                  01 AND OUTP! MBV2; \ Do an INV of IN bit0,
\ Forth Words used - remember: Any Forth Word consists of a character sequence plus a space (not allowed in Words is Space )
\ 0 INCLUDE
              INCLUDE loads and starts a file in VFX; set up c:\VFXTESTAPP, store file there, type INCLUDE c:\VFXTESTAPP\VFXTESTAPP.f
\ 1 HEX
              Mostly VFX recognizes numbers as decimal or HEX. If in DECIMAL mode any Hex Number will give an error. Careful!
\ 2 \
              \ and a Space after it tells Forth, that the rest of the line is for documentation, so is ignored by VFX.
               : starts new Forth Word definition, is terminated by ; see Number 6. Example : Bell 07 EMIT ; sends a sound.
\ 3:
\ 4 ."
               ." and a space starts a string of characters to be sent to the screen, the string is terminated by "
\ 5 CR
               CR defines a word that places the cursor to the beginning of the next line
\ 6;
               ; ends the definition started by : see number 3
\ 7 TEST
               TEST is the name of a new word defined just to display the GUI - the Graphical User Interface
\ 8 Variable Variables = named memory locations. VARIABLE xx defines, nn xx ! sets, xx @ . prints. Remember: NAME and contents
\ 9 DUP
               DUP takes the value on top of the stack and puts the same value on top as NEW TOS - Top of Stack
\ 10 $n
               $n - the $ ensures that VFX takes the number following as hexadecimal number
\ 11 AND
               AND is a logical operator. works on the two top numbers on the stack, on a bitwise basis, only leaves result
\ 12 IF
               IF is not zero do A - ELSE do B - THEN continue
\ 13 ELSE
\ 14 THEN
              LSHIFT does a bitwise shift of the value on stack, expects the number of shifts to be done on stack - 1 LSHIFT
\ 15 LSHIFT
\ 16 SPACE
               SPACE sends a space to the screen
\ 17 DROP
               DROP does the opposite to DUP; DROP takes the TOS and deletes what was there, all values move up one position
\ 18 DV
               DV is a word that combines the 4x 4 bits of the variables for the screen, 16 Bit wide.
\ 19 @
               @ stands for me for AT, for example with Variables - xx puts address of xx onto the stack, xx @ gets the value of it
\ 20 ?DO
               ?DO will start a DO word word LOOP, expects the number of loops on stack. ?DO checks if number on stack is 0.
\ 21 LOOP
               Goes back to ?DO until 0, then continues
\ 22 PAGE
               PAGE clears the screen, takes the cursor to the top left position and prints ok
\ 23 BEGIN
               BEGIN starts a loop of words which ends with UNTIL. UNTIL expects a flag on stack; not 0 then to BEGIN else continue
\ 24 1+
               1+ is put on stack, adds one to the number below and replaces it. Same function as 1 + as 2 words.
\ 25 !
               ! is the opposite function to @, and stores the number on stack at a location on stack, e.g. 55 xx ! at Variable xx
\ 26 MS
              MS stands 1 Millisecond delay in software. Careful which base you are in - 256 and 100 the same Dec / HEX
\ 27 KEY?
               Key? Checks, if a key has been pushed as this generates a flag. BEGIN xxxxxxxx KEY? UNTIL - flag stops the loop
\ 28 UNTIL
               UNTIL end the BEGIN ... UNTIL loop, exit if flag true.
\ 29 EMIT
              (07 for Bell ) to get audible output from the PC - (:) BELL 07 BEL (;) sends the bell signal to the speaker
\ 30 .s
               .S is a non-destructive display of the stack, all of the items there
\ 31 .
               . is similar to .S, prints out the top item only and as well consumes it; an easy way to get rid of the top stack item
              >R takes an item from DSTACK and moves to the RSTACK;
\ 32 >R
\ 33 R>
               R> does the opposite: take item from RSTACK move it to the top of DSTACK
\ 34 (and) These 2 brackets are used to put explanations within the code, and this is ignored by Forth, similar to \
\ ----- it seems this covers all of the words needed for this little sandbox application
\ This application VFXTESTAPP has not been programmed optimally - but this was not the target - beginner's code for beginners
\ Links to further information: see WORD file and PDF version of this text
: Name ." Hello Forth World ";
Page CR Name 1000 ms
         \ send SOS
                         and then start the counter
sos
counter \ start COUNTER
```

\ The Forth Machine: Data stream coming in on the left, 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)	$DS-\overline{1}$	$RS-\overline{1}$		x6	(for later)	
	DS-2	RS-2		x5		
	DS-3	RS-3		x4		
	DS-4	RS-4	PSWI	жЗ		
	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

MPE_VFX_LITE – download, installation and test



Go to www.mpeforth.com and have a look around. Have a look at the Press section and see what is new.

This website is in the state of transition, it has been updated recently.

Open the tab downloads http://www.mpeforth.com/arena.htm. The link we want is the MSP430 Lite compiler and the link Windows installer.

I suggest you create a new folder MPEFORTH first.

Download into this folder and execute xMSP430lite.exe.

From the files there there are look at: AIDF Manual and start AIDF Shell

In the Utilities window you start ForthEd2 and Forth Console.

Resort the windows and you will have the same display as further down:

The AIDE Window with the 4 other windows:

Power Term – not neeed for now, so switch it off in Utilities.

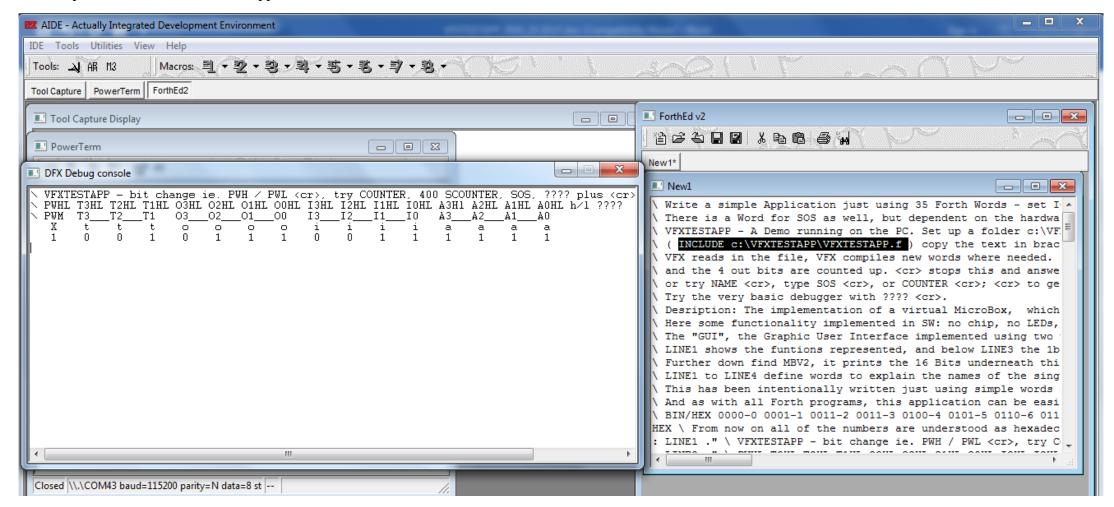
Tool Capture Display: this for later so can be disabled as well.

ForthEd2 our very basic ASCII editor.

And most importantly the Forth Console, this shows the VFX Forth software, here running VFXTESTAPP; and you can see part of the file in ForthEd2.

There will be a download location to be defined. For now it is http://wiki.forth-ev.de/doku.php/en:projects:a-start-with-forth:start

More explanations in the text of the application.



You can as well download a smaller size version of the AIDE Manual from the Forth-eV location if you wish - enjoy A Start in Forth