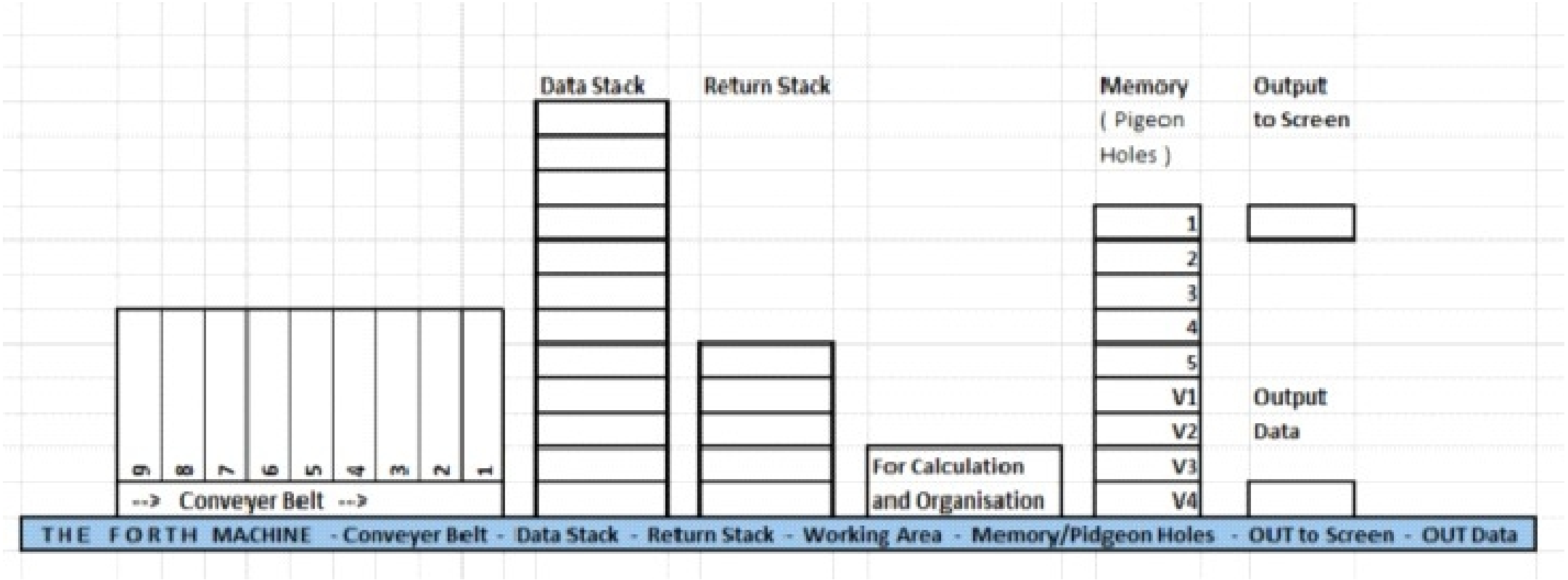
```
\ Write a simple Application using 35 Forth Words - set IO bits and start a counter   --  ExMark -  Juergen Pintaske Sept2016
\ There is a Word for SOS as well, but dependent on the hardware you can hear the SOS as some sounds (or not) just try
\ UMMT Demo running on the PC. Set up a folder c:\UMMT. Store this file UMMTDEMO.f into this folder
\ INCLUDE c:\UMMTDEMO\UMMT3.f - copy this into the VFX window, and hit <cr>  to run the demo, .
\ A few lines "GUI" will be shown. Type in  PWL <cr>, changes 1 to 0, or other bits, sos <cr>, counter <cr> and ???? <cr>

\ Copy file into VFX, see how VFX compiles to learn new words; when finished, type <cr>  and then type COUNTER or 400 SCOUNTER
\ BIN_HEX 0000-0 0001-1 0010-2 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
\ Desription: This is an implementation of the MicroBox,  which consists of a Micro, PWM LED, 3 Switches, 4 OUT LEDs, 4 IN LEDs
\ Here implements some functionality in SW: no chip, no LEDs, no resistors, no PCB, no soldering, execute the same - PC only


\ The "GUI", the Graphic User Interface is implemented using two very simple words - Page to clear screen, ." type text lines
\ MMT7 shows the funtions represented, and below MMT7 the 1bit PWM, 3bits SWITCHES, 4bits OUT, 4bits IN, 4bits ANI - 16 bits
\ Further down find MBV2, it prints the 16 Bits underneath this text. All lines are resent whenever the 16 bits are updated.
\ MMT4 and MMT5 show the words that control the single bits or nibbles.   PWF = flashing, ??? and ???? for DEBUG,  more tbd.
\ 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 interactively.

HEX \ from now on all of the numbers are hexadecimal.  MMT lines print a display window GUI, very basic, just uses Page and CR
: MMT0 ." \ copy INCLUDE c:\UMMTDEMO\UMMT3.f into VFX, hit <cr> ExMark Juergen Pintaske 16_09"  CR ; \
: MMT1 ." PSWI OUT  IN   ANI  - R0  R1   - Stack contents                                    "  CR ; \ description of ??? display
: MMT2 ." psss oooo iiii aaaa   rrrr rrrr                                                    "  CR ; \ Variables - RSTACK - DSTACK
\ : MMT2a ." 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" ;
: MMT2b ."   PWHL S3HL S2HL S1HL O3HL O2HL O1HL O0HL I3HL I2HL I1HL I0HL A3Hl A2HL A1HL A0HL h/l ????" CR ; \ ???? DEBUG
: MMT3 ." \ PWM  T3   T2   T1   O3   O2   O1   O0   I3   I2   I1   I0   A3   A2   A1   A0  " CR ;
: MMT4 ." \ PWH  T3H  T2H  T1H  O3H  O2H  O1H  O0H  I3H  I2H  I1H  I0H  A3H  A2H  A1H  A0H     Commands A \"  CR ; \ taken out?
: MMT5 ." \ PWL  T3L  T2L  T1L  O3L  O2L  O1L  O0L  I3L  I2L  I1L  I0L  A3L  A2L  A1L  A0L     Commands B \"  CR ; \ taken out?
: MMT6 ." \ PWF  ???  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd  tbd     Commands C \"  CR ; \ taken out?
: MMT7 ."    X    s    s    s    o    o    o    o    i    i    i    i    a    a    a    a  "  CR ;  \ display after START?

: MicroBox PAGE MMT0 ( MMT1 ) ( MMT2 ) ( MMT2b ) ( MMT3 ) ( MMT4 ) ( MMT5 ) ( MMT6 ) MMT7 CR ; \ display some lines, others not

Variable PWM    ( 0 or 1                                )      \ not used for now
Variable SWI    ( Switches SW3 SW2 SW1    0 0 0         )      \ not used for now
Variable PSWI   ( Combined PWM and Switches  X s s s    )  \ combines the bit for PWM and the 3 bits for the 3 switches
Variable OUTP   ( OUTP 3 2 1 0 )( same as Bit 7 6 5 4   )  \ the 4 "OUTPUT LEDs" of the MicroBox
Variable IN     ( IN  3 2 1 0  ( 3 2 1 0 of the 8 Bits )  \ Inputs that can be set e.g. for AND OR XOR INVERT
Variable ANI    ( Simulated analog input 0 to F         )  \ either simulated external ANALOG - or free for anything
Variable HOU    ( OUT as hexadecimal number 0 to F      )  \ to be implemented, just showing the 4 OUT bits in hexadecimal
Variable ALL    ( PWM SWI OUT INP ANI                   )  \ To be done, combining it all into a 16 bit Variable

: disbit4  DUP $8 AND IF ." 1" ELSE ." 0" THEN  ;   \ display a bit, here the topmost of the 4
: ds       Disbit4 1 LSHIFT ;                       \ display a bit and shift the 4 bit left by one
: dssp     ds Space Space space space ;             \ combine to a block using spaces per bit
: 4dssp    dssp dssp dssp disbit4  drop ;           \ and the 4 bits of a nibble
: DV       3 spaces PSWI @ 4dssp  4 spaces OUTP @ 4dssp 4 spaces IN @ 4dssp  4 spaces  ANI @ 4dssp  ;  \ display the 16 bits
: SPACES   ( u -- ) 0 ?DO SPACE LOOP ;              \ define SPACES, often included in the wordset already

: MBV2     PAGE MMT0 ( MMT1 ) ( MMT2 )  MMT2b ( MMT3 ) ( MMT4 ) ( MMT5 ) ( MMT6 ) MMT7  DV CR ;   \ MBV2 updates the "SCREEN"

: COUNTER   Begin outp @ 1+ outp !     300 ms mbv2 key? until ; \ run counter program, see the OUT bits change, just type counter <cr>
: SCOUNTER  Begin dup outp @ 1+ outp !     ms mbv2 key? until ; \  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 ;
```

```
: ???    PSWI @ .  OUTP @ .  IN @ .  ANI @ .   base @ >r hex .S r> base ! ; \ Small Debugger - prints 4 variables + Data Stack contents
                                                \ to be modified to a compromise ???? to replace .S, so all data printed out as one line
: ????   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 . ;

\ Preset Variables:
0 PWM  !    \      not used for now
0 SWI  !    \      not used for now
9 PSWI !    \  Set this combined variable to 9 = 1001  this will show the PWM LED ON and SW1 ON
0 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 but not used for now
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 MMT3 and MMT4 )
: PWH   PSWI @  $8 OR   PSWI !  MBV2 ; \ Get the variable onto stack, set bit 3, and store it back, now update the Display
: T3H   PSWI @  $4 OR   PSWI !  MBV2 ; \ Get the variable onto stack, set bit 2, and store it back, see above
: T2H   PSWI @  $2 OR   PSWI !  MBV2 ; \ And the same for setting all of the other bits
: T1H   PSWI @  $1 OR   PSWI !  MBV2 ;
: O3H   OUTP @  $8 OR   OUTP !  MBV2 ;  \ x x x x    1 0 0 0
: O2H   OUTP @  $4 OR   OUTP !  MBV2 ;  \            0 1 0 0
: O1H   OUTP @  $2 OR   OUTP !  MBV2 ;  \            0 0 1 0
: O0H   OUTP @  $1 OR   OUTP !  MBV2 ;  \            0 0 0 1
: 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 ;  \ here end the words that Set Bits HIGH - the next ones set the same bits LOW
: 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 @  $D AND  PSWI !  MBV2 ;  \            1 1 0 1
: T1L   PSWI @  $E AND  PSWI !  MBV2 ;  \            1 1 1 0
: 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 and see the result of the logic result in O0
: AND01    IN @  DUP 1 RSHIFT AND 01 AND  OUTP !  MBV2  ;   \ Do an AND of IN bit0 and bit1, 00=>1 01=>0 10=>0 11=>1
: OR01     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
: 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
: INVERT0  IN @  INVERT           01 AND  OUTP !  MBV2  ;   \ Do an INV of IN bit0,           0=>1  1=>0

\ Forth Words used - remember: Any Forth Word consists of a character sequence plus a space (not allowed in Words are CR, BS, Space )
\  0 INCLUDE   INCLUDE will load and start a file in VFX; set up c:\ummtdemo, store file there and type  INCLUDE c:\UMMTDEMO\UMMTDEMO.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.
\  3 :         : starts the definition of a new WORD, the definition is terminated by ; see 6  : Bell 07 EMIT ; sends a sound.
\  4 ."        ." SPACE starts a string of characters to be sent to the screen, is terminated by "
\  5 CR        CR send a Carriage Return to the screen
\  6 ;         ; ends the definition started by :
\  7 MicroBox  MicroBox is the name of a new word defined
\  8 Variable  Variables are memory locations with a name. VARIABLE xx to define, nn xx ! to preset, xx @ . print
\  9 DUP       DUP takes the value on top of the stack and put 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  not zero do A  ELSE do B THEN continue
\ 13 ELSE
\ 14 THEN
\ 15 LSHIFT    LSHIFT does a bitwise shift of the value on stack, expects the number of shifts to be done on stack - 1 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 will send the combined variables to the screen 16 Bit wide. A DVG could send a general 16 bit
\ 19 @         @ stands for AT, for example with Variables - xx puts the address of xx onto the stack, xx @ replaces with value
\ 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
\ 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; returns to BEGIN or continues
\ 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 a number on stack at a location on stack, e.g. 55 xx ! at Variable
\ 26 MS        MS stands for 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, and all of the items
\ 31 .         . is similar, prints out the top item only and as well consumes it; an easy way to get rid of the top stack item
\ 32 >R        >R takes an item from DSTACK and moves to the RSTACK;
\ 33 R>        R> does the opposite: take item from RSTACK move it to the DSTACK
\ 34 CR        CR defines a word that goes to the beginning of the next line
\  ------ it seems this covers all of the words used in this little application
\
\ We know this application is not programmed optimally  - but this was not the target -  beginner's code for beginners
\ Links  to further information: see word and PDF version of this
\
sos      \ send  SOS      and then
```

```
counter  \ start COUNTER
\ Programming Code again, but with spae included for handwritten comments and explanations

: Name  ." Hello my name is Duncan " ;


: disbit4  DUP $8 AND IF ." 1" ELSE ." 0" THEN  ;   \ display a bit, here the topmost of the 4


: ds       Disbit4 1 LSHIFT ;                       \ display a bit and shift the 4 bit left by one


: dssp     ds Space Space space space ;             \ combine to a block using spaces per bit


: 4dssp    dssp dssp dssp disbit4  drop ;           \ and the 4 bits of a nibble


: DV       3 spaces PSWI @ 4dssp  4 spaces OUTP @ 4dssp 4 spaces IN @ 4dssp  4 spaces  ANI @ 4dssp  ;  \ display the 16 bits


: SPACES   ( u -- ) 0 ?DO SPACE LOOP ;              \ define SPACES, often included in the wordset already


: MBV2     PAGE MMT0 ( MMT1 ) ( MMT2 )  MMT2b ( MMT3 ) ( MMT4 ) ( MMT5 ) ( MMT6 ) MMT7  DV CR ;   \ MBV2 updates the "SCREEN"


: COUNTER   Begin outp @ 1+ outp !     300 ms mbv2 key? until ; \ run counter program, see the OUT bits change, just type counter <cr>
: SCOUNTER  Begin dup outp @ 1+ outp !     ms mbv2 key? until ; \  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 ;
```

```
: Bell  07 emit ;
```


Data Stack
Return Stack
Memory
(Pigeon
Holes)
Output
to Screen
1
2
3
4
5
V1
V2
V3
V4
Output
Data
For Calculation
and Organisation
9 8 7 6 5 4 3 2 1
--> Conveyer Belt -->
THE FORTH MACHINE - Conveyer Belt - Data Stack - Return Stack - Working Area - Memory/Pidgeon Holes - OUT to Screen - OUT Data