```
\ swk4 "swiss knife" 4bit I/O
\ using 4e4th Release0.34 on TI MSP430G2553 MCU LaunchPad
\ started: mka, 2014_10_26
\ Idea: Juergen Pintaske



\ History:
\ added: DEMOLOOP to select examples with S2 and S1 switches.
\ cleaning up examples, added EX12 (3.11.2014)
\ added: 2 servo examples (02.11.2014)
\ added: 4 servo pulses to OUT (30.10.2014 22:47)
\ renamed demos to EXx (29.10.2014 21:19)
\ added: some demos Px (29.10.2014 13:05)
\ added: ADC (29.10.2014 00:02)
\ first test today (26.10.2014 23:49)
\   Compiles in target without errors.
\   blocks with mk0 mark = tested ok on first glance ...
    ... functioning, stack ok.
\   MYBONNIE is playing, no stack error - ok
\   PWM running, SWEEP is ok.
\   I/O and nibbels working on variables.
\   mem u. 6703 (plenty flash left)

\ Issues:
\ Glitches while setting upper nibble of P2 - fixed OUT!
\ port name P2 is in conflict with modul name P2 - renamed.

\ ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
\ Start of Program  SWK4 programmed in Forth

\ This program gives some routines planned to help Learning \ Programming.
\ And we use them here as an example of how to program the
\ MSP430 Controller.
\ Add the 5 resistor LED combinations and the
\ 3 switches to e th430 board and run routines completely
\ in the MSP430, using Forth wich is on this chip as well.



\ Defining the I/O for this application

\ Device Pinout: MSP430G2553 20-Pin PDIP
\                     (TOP VIEW)

\ VCC-----------------[01   20]-------------VSS
\ (LED1 LP) AD0 P1.0--[02   19]--P2.6 OUT2
\ RXD-----------P1.1--[03   18]--P2.7 OUT3
\ TXD-----------P1.2--[04   17]--test
\ (S2 LP)----S2 P1.3--[05   16]--RST      S1-LP
\ ..........AD4 P1.4--[06   15]--P1.7 S1          [SPI]
\ _-_-_-_-_-FRQ P1.5--[07   14]--P1.6 PWM LED2-LP [SPI] TA0.1
\ ----------IN0 P2.0--[08   13]--P2.5 OUT1
\ ----------IN1 P2.1--[09   12]--P2.4 OUT0
\ ----------IN2 P2.2--[10   11]--P2.3 IN3

\ P1.0 used as Input, later on as Analog Input 1
\ P1.1 RX and
\ P1.2 TX are used as the serial interface to the PC
\ P1.3 used as S2 as on the TI Launchpad,
\      internal resistor enabled
\ P1.4 used as Input, later on as Analog Input 2
\ P1.5 used as Output,
\      later on to output a frequency with defined length
\ P1.6 Outout, later on as Pulse Width Modulation output
```

```
\      â€“ quasi D/A Output
\ P1.7 Input S1, internal resistor enabled

\ All P2.x Inputs have internal resistor enabled,
\ so open input means HIGH
\ P2.0 Input 0
\ P2.1 Input 1
\ P2.2 Input 2
\ P2.3 Input 3

\ P2.4 Output 0
\ P2.5 Output 1
\ P2.6 Output 2
\ P2.7 Output 3



\ The MSP 430 will be rather fully used.
\ So to save Program Space, the 4 Bit Nibbles are packaged
\ into 4x4 nibbles to be one Variable (16Bit cell).
\ This will save on RAM used.

\ Some scrach variables    mk0
\ IN  4 bit input, when used reflects status of 4 Input lines
\ F   4 bit value for tones
\ A   4 bit A register
\ B   4 bit B register
variable IFAB     \ define a variable

\ W   4 bit W variable
\ X   4 bit A variable
\ Y   4 bit Y variable
\ Z   4 bit Z variable
variable WXYZ

\ A1  4 bit analog in channel 1
\ A2  4 bit analog in channel 2
\ P   4 bit pulse width modulated output
\ O   4 bit output register
variable 12PO

: INITSV \ -- \ init scrach variables
  0 IFAB !    0 WXYZ !    0 12PO ! ;
  initsv

\ define a Word to show a cell at address unsigned   mk0
: ? ( adr -- )   @ u. ;



\ juggling nibbles    mk0
HEX
: 0! \ n adr -- \ store nibble 0 to variable at address
  >r r@ @ FFF0 and swap F and + r> ! ;
: 1! \ n adr -- \ store nibble 1 to variable at address
  >r r@ @ FF0F and swap F and 4 lshift + r> ! ;
: 2! \ n adr -- \ store nibble 2 to variable at address
  >r r@ @ F0FF and swap F and 8 lshift + r> ! ;
: 3! \ n adr -- \ store nibble 3 to variable at address
  >r r@ @ 0FFF and swap F and C lshift + r> ! ;

: 0@ \ adr - n \ fetch nibble 0 of variable at address
  @ 000F and ;
: 1@ \ adr - n \ fetch nibble 1 of variable at address
  @ 00F0 and 0004 rshift ;
```

```
: 2@ \ adr - n \ fetch nibble 2 of variable at address
  @ 0F00 and 0008 rshift ;
: 3@ \ adr - n \ fetch nibble 3 of variable at address
  @ F000 and 000C rshift ;


\ As  a first step we have to define the relevant IO Bits
\ as Input ( with enabling internal pull-up resistor)
\ or as output.
\ Here you can find the relevant memory mapped addresses
\ to set functionality, read Input Data, write Output Data
\ Some of the functions used are rather complex,
\ so in the beginning we will test as Input and Output only,
\ see separate Forth Code


\ In 4e4th, P1 and P2 are predefined 2constants.
HEX
: INITIO \ -- \ I/O initialisation of JPS4 ports   mk0
\ mask adr op
\       76543210
[ bin ] 10011001 [ hex ] 0022 ( P1DIR ) cclr \ P1 INs
[ bin ] 01100110 [ hex ] 0022 ( P1DIR ) cset \ P1 OUTs
[ bin ] 10001000 [ hex ] 0024 ( P1IES ) cset \ falling edge detect
[ bin ] 00100000 [ hex ] 0026 ( P1SEL ) cset \ P1.5 sec func TA0.0 (FRQ)
[ bin ] 10001000 [ hex ] 0027 ( P1REN ) cset \ pullup selected
[ bin ] 10000000 [ hex ] P1 cset \ P1.7 pullup enabled
[ bin ] 00001111 [ hex ] 002A ( P2DIR ) cclr \ P2 INs
[ bin ] 11110000 [ hex ] 002A ( P2DIR ) cset \ P2 OUTs
[ bin ] 00001111 [ hex ] 002F ( P2REN ) cset \ P2 pullups selected
[ bin ] 00001111 [ hex ] P2 cset \ P2 pullups enabled
[ bin ] 11000000 [ hex ] 002E ( P2SEL ) cclr \ clear these bits to I/O

; INITIO



\ Basic I/O          mk0
HEX
: OUT!  \ n -- \ write OUT to P2 upper nibble (4 LEDS)
  04 lshift    \ move to n upper nibble
  0F or        \ keep input pullups set!
  p2 c! ;      \ write port

: OUT>  \ --   \ copy O to OUT!
  12PO 0@ out! ;

: OUT   \ n --   \ save n to O and do OUT!
  12PO 0! out> ;

: IN@ \ -- n \ read digital input pins (nibble )
  P2 1- c@ 000F and ;

\ helper to output a nibble to terminal
HEX
: 4#  \ n --   \ display 4 digits unsigned with leading zeros
  <#  zero # # # #   #> TYPE SPACE ;
: 2#  \ n --   \ display 2 digits unsigned with leading zeros
  <#  zero # #    #> TYPE SPACE ;

\ test falling edge at switch S1 and S2      mk0
023 constant P1IFG
: S1?  080 P1IFG cget ;   \ -- f   \ set on edge event
: S2?  008 P1IFG cget ;   \ -- f   \ set on edge event
: S1-  080 P1IFG cclr ;   \ --   \ reset edge event flag
```

```
: S2-  008 P1IFG cclr ;   \ --   \ reset edge event flag



\ Delays    mk0
\ 1MS is prefefined in 4e4th. Its an empty loop,
\ predefined for MCU running with 8Mhz DCO = default.
\ Use Forth word MS to make any ms delay.
\ Example:
\ decimal 10 ms   \ delay of 10 miliseconds.

DECIMAL
: 1SEC  \ -- \ delay of one second
  1000 ms ;

: SECS  \ n --   \ delay of n seconds      mk0
  0 DO 1sec LOOP ;



\ ------------------------------------------------------------
\ Utilities for this MCU
\ ------------------------------------------------------------

\ ------------------------------------------------------------
\ Tones
\ Output is for MSP430G2553, 8Mhz DCO and SMCLK /2

HEX
\ Output square wave at P1.5 using timer TA0.0      mk0

\ Use P1.5 as timer TA0.0 ouput.
\ This is done by setting P1.5 to its second function.
\ Then set and start Timer.
\ Note: In second function, pin is no longer
\ a general purpose I/O pin. It is switched to another I/O
\ mode called 'second function' of the pin.

: P15SEC   \ set P1.5 to its second function
  020 dup p1 1+ cset  026 cset ;

: P15IO     \ set back P1.5 to its GPIO
  020 026 cclr  020 041 cclr ;

: TON-  zero 0160 ! p15io ;  \ stop timer

: TON+ \ n --   \ start timer-A with interval n
  p15sec        \ init pin
  0080  0162 !  \ CCTL0  set timer output mode
  ( n ) 0172 !  \ CCR0   set interval
  0254  0160 !  \ CTL    start timer clock, mode and divider
  ;

\ midi tones B3 to B5, 2 octaves; pitch list.   mk0
DECIMAL
IHERE
 0000 i,  \ no tone
 7962 i,  \ B3
 7515 i,  \ C4
 6695 i,  \ D4
 5965 i,  \ E4
 5630 i,  \ F4
 5016 i,  \ G4
 4469 i,  \ A4
 3981 i,  \ B4
```

```
 3758 i,  \ C5
 3348 i,  \ D5
 2983 i,  \ E5
 2815 i,  \ F5
 2508 i,  \ G5
 2235 i,  \ A5
 1991 i,  \ B5
constant TONELIST

: T@   \ i -- n   \ get pitch using index i
  cells tonelist + @ ;

DECIMAL
: NOTE   \ i --   \ play note i
  t@ ton+ 500 ms ;
: LNOTE  \ i --   \ play note i longer
  note 500 ms ;
: PAUSE  \ n --   \ play pause of duration n
  ton- ms ;  \ ( use n = 50,100,200)

\ see example melodie EX14



\ ------------------------------------------------------------
\ PWM of 16KHz                          mk0
\ for 8MHZ DCO using timer TA0

HEX
: P16IO   \ --   \ set P1.6 to GPIO
  0040 0022 cset    \ P1DIR   P1.6 OUT
  0040 0026 cclr    \ P1SEL   P1.6 GPIO
  ;

: P16SEC  \ --   \ set P1.6 to second function
  0040 0022 cset    \ P1DIR   P1.6 OUT
  0040 0026 cset    \ P1SEL   P1.6 select second function
  ;

: PWM-
  zero 160 !  \ TA0CTL   stop timer
  p16io 0040 0021 cclr ; \ set p1.6 I/O and clear p1.6

: PWM+  \ n --  \ init and start PWM at P1.6
  01F4 0172 !   \ TA0CCR0    set period 16KHz at 8MHZ DCO
  00E0 0164 !   \ TA0CCTL1   set output mode
       0174 !   \ TA0CCR1    set pulsewidth
  0210 0160 !   \ TA0CTL     set timer mode and run
  ;

: PWM  \ n --   \ set PWM at P1.6   n = 0...F
  ton- 000F and
  dup 0= IF drop pwm-
  ELSE 001F * pwm+ p16sec THEN ;



\ ------------------------------------------------------------
\ ADC
HEX
\ You may name addresses and bits.

\ address name \ function
\ 01B0 constant ADC10CTL0 \ ADC10 control register 0
\ 01B2 constant ADC10CTL1 \ ADC10 control register 1
```

```
\ 004A constant ADC10AE0  \ ADC10 input enable register 0
\ 01B4 constant ADC10MEM  \ ADC10 Memory, conversion result

\ some named bits
\ 0002 constant ENC       \ ADC10 Enable Conversion bit
\ 4000 constant INCH_4    \ Selects Channel 4
\   10 constant BIT4      \ for P1.4 Analog Input Enable
\ 0001 constant ADC10SC   \ start conversion bit
\ 0004 constant ADC10IFG  \ ADC10 Interrupt Flag

\ option bits are:
\ 2000 constant SREF_1     \ set VR+ = VREF+ and VR- = AVSS
\ 1000 constant ADC10SHT_2 \ 16 x ADC10CLKs
\ 0040 constant REF2_5V    \ ADC10 Ref 0:1.5V / 1:2.5V
\ 0020 constant REFON      \ ADC10 Reference on
\ 0010 constant ADC10ON    \ ADC10 On/Enable
\ ----
\ 3070 <-- sum of option bits

: ADCOFF   \ --  \ stop ADC10 ...
  zero 01B0 ! ;   \ ... can be modified only when ENC = 0

: ADCON    \ --  \ start ADC10
  0002 01B0 set ;  \ Set ENC

: ADC@   \  -- x   \ get ADC value
  03 01B0 SET \ start conversion (ENC+ADC10SC bits)
  BEGIN 04 01B0 cget UNTIL \ test BIT2 (ADC10IFG)
  01B4 @  ; \ get result

: ADC0     \ --   \ select and init ADC channel 0
  adcoff
  3070 01B0 !      \ set options (see MCU user manual)
  0100 01B2 !      \ select input channel (A0 at P1.0)
  01 004A cset ;   \ set pin, analog input enable

: ADC4     \ --   \ select and init ADC channel 4
  adcoff
  3070 01B0 !      \ set options (see MCU user manual)
  4000 01B2 !      \ select input channel (A4 at P1.4)
  10 004A cset ;   \ set pin, analog input enable

\ Example:
\ adc4 adcon adc@ .  \ print single conversion



\ ------------------------------------------------------------
\ 4 servos connected to OUT

\ Use extern DC power supply for servos.
\ Connect servo-GND to MCU-GND.
\ Connect servo control lines to OUTx.
\ Control line needs one 1..2ms puls, 20ms pause, ~$40x,
\ for one position

HEX
variable x0
variable x1
variable x2
variable x3

: LDX    \ x0 x1 x2 x3 -- \ load variables X0 .. X3
  x3 ! x2 ! x1 ! x0 ! ;
```

```
: INITX  \ -- \ load X0 .. X3 wit &500
  500 dup dup dup ldx ;
  initx

: x?     \ -- \  print all X
  x0 ? x1 ? x2 ? x3 ? ;

: DEX    \ x -- \ waste time x
  0 DO LOOP ;

: PULS   \ x n -- \ send puls x to bit n of P2; n=80,40,20,10
  >r r@ p2 cset dex r> p2 cclr 4 ms ;

: SERVE   \ -- \ send one puls to all servos
  x0 @ 10 puls
  x1 @ 20 puls
  x2 @ 40 puls
  x3 @ 80 puls ;

: SERVOS \ -- \ position all servos
  initio
  30 0 DO serve LOOP ;

DECIMAL
\ set x to &480 ... &1700 (Range is ~1200 steps)
: SER0 \ x -- \ reposition servo0
  x0 ! servos ;
: SER1 \ x -- \ reposition servo1
  x1 ! servos ;
: SER2 \ x -- \ reposition servo2
  x2 ! servos ;
: SER3 \ x -- \ reposition servo3
  x3 ! servos ;

\ adjust x-min and x-max to match your servo.

\ test for oscilloscope
: SERTEST \ -- \ position all servos, permanent
  initio BEGIN serve key? UNTIl ;



\ ------------------------------------------------------------
\ Initialisation
DECIMAL
: INIT \ -- \ set pin I/O and variables.
  initio initsv initx ;



\ ------------------------------------------------------------
\ Example Programms
\ ------------------------------------------------------------

\ ------------------------------------------------------------
\ Example 0 - leave demo loop, run forth.
: EX0
  cr .ver
  cr ." forth - command me." abort ;



\ ------------------------------------------------------------
\ Example 1 - display IN (1)
HEX
```

```
: EX1  \ -- \  IN to OUT                                  mk0
  initio
  BEGIN
  in@ out               \ get IN and store it to OUT
  key? UNTIL key drop ; \ leave loop, clean up



\ ------------------------------------------------------------
\ Example 2 - display IN (2)
HEX
: EX2 \ -- \   IN to PWM and OUT                          mk0
  initio
  BEGIN
  in@               \ get IN
  dup out           \ store to OUT
      PWM  1sec     \ do PWM for a second
  key? UNTIL        \ leave loop, clean up
  key drop  pwm- ;



\ ------------------------------------------------------------
\ Example 3 - falling edge detector (1)
HEX
: EX3  \ -- \   faling edge detector S2 to OUT0      mk0
  initio s2-        \ init system and reset edge detection
  BEGIN
  s2? IF
      1 out               \ set on edge detect
      1sec s2- zero out   \ wait, then reset
      THEN
  key? UNTIL        \ leave loop, clean up
  key drop  ;



\ ------------------------------------------------------------
\ Example 4 - falling edge detector (2)
HEX
: EX4  \ -- \   falling edge detector S2 to note      mk0
  initio s2-  \ init system and reset edge detection
  01 note     \ start low note
  BEGIN
  s2? IF
      1 out               \ set on edge detect
      0F note             \ do high note
      1sec s2- zero out   \ wait, then reset out ...
      01 note             \ and low note
      THEN
  key? UNTIL        \ leave loop, clean up
  key drop  ton- ;



\ ------------------------------------------------------------
\ Example 5 - toggle OUT bits
HEX
: tout \ c -- c \
  dup [char] 1 = IF 01 12po ctoggle out> exit THEN
  dup [char] 2 = IF 02 12po ctoggle out> exit  THEN
  dup [char] 3 = IF 04 12po ctoggle out> exit  THEN
  dup [char] 4 = IF 08 12po ctoggle out> exit  THEN
  dup [char] 0 = IF zero out exit THEN
  dup [char] f = IF 0f out exit  THEN  ;
```

```
: EX5 \ -- \  press 1 2 3 4 to toggel OUT bits,
          \ 0 to clear all, F to set all.
  initio zero out
  BEGIN
  key tout
  1B ( esc ) = UNTIL
  ; \ exit on esc-character



\ ------------------------------------------------------------
\ Example 6 - dance of 4 servos
DECIMAL
0480 constant SL
1700 constant SR
1000 constant SM

IHERE        \ create a list of moves
SL i, SL i, SL i, SL i,
SR i, SR i, SM i, SR i,
SM i, SM i, SR i, SM i,
SL i, SL i, SM i, SL i,
SM i, SM i, SL i, SM i,
SR i, SR i, SR i, SR i,
CONSTANT M0  \ -- adr \ get list address

: ROWS   \ i -- n   \ calculate row offset in bytes
  4 cells * ;

: ROW@  \ adr -- x0 x1 x2 x3  \ get 4 values from adr
  dup 4 cells + swap
  DO  i @  cell +LOOP ;

: DANCE  \ adr n -- \ play list at address, n rows.
  rows over + swap    \ calculate end of list
  DO   \ from beginning till end of list
  ( i u. ) \ testing
  i row@ ldx servos   \ drive servos to positions of row
  1 rows +LOOP ;      \ advance one row

: EX6 \ --  \ dance until key is pressed
  initio
  BEGIN
  m0 6 dance   \ set list and rows, then play it once ...
  key? UNTIL   \ ... again until key is pressed.
  key drop  ;   \ clean up.


\ ------------------------------------------------------------
\ Example 7 - servo follows analog input pin (ADC4)

\ ADC@ is 0 .. $3FF
\ SERVO is &480 .. &1700

DECIMAL
: POSITION \ adc -- pos \ scale adc value to servo position
  480 + ;

: EX7 \ --  \ Position of servo2 given by potentiometer
  initio       \ init ports,
  adc4 adcon   \ init ADC
  initx        \ statposition of servos
  BEGIN
  adc@ position x2 ! serve
```

```
  key? UNTIL
  key drop  ;   \ clean up.


\ ------------------------------------------------------------
\ Example 8 - get analog input
HEX
: EX8 \ -- \ ADC to OUT and terminal                  mk0
  initio adc4 adcon  \ init all modules
  BEGIN
  adc@ 44 /          \ scale 10Bit value, 3FF..0 --> F...0
  dup out            \ display scaled value
  dup note 50 pause  \ play apropriate tone
      .              \ print to terminal
  key? UNTIL         \ leave loop, clean up
  key drop  ton- ;



\ ------------------------------------------------------------
\ Example 9 - tbd
: EX9 ;



\ ------------------------------------------------------------
\ Example 10 - tbd
: EX10 ;



\ ------------------------------------------------------------
\ Example 11 - tbd
: EX11 ;



\ ------------------------------------------------------------
\ Example 12 - pressing S2 increments OUT
HEX
: EX12  \ -- \   faling edge detector S2, counting up OUT
  initio s2-  \ init system and reset edge detection
  F OUT       \ initial OUT value
  BEGIN
  s2? IF
      12po 0@ 1+ F and 12po 0! \ increment OUT nibble
      out>                     \ display it
      100 ms s2-     \ wait, then reset S2
      THEN
  key? UNTIL         \ leave loop, clean up
  key drop  ;



\ ------------------------------------------------------------
\ Example 13 - echo any key
: EX13 \ -- \   terminal KEY to OUT with echo          MK=
  initio zero out
  cr ." press any key to start - press Esc-key to exit"
  cr  \ new line
  BEGIN
  key       \ get key
  dup over 20 < IF
    5E emit 40 + emit   \ echo control character
    ELSE emit THEN      \ echo character
```

```
  dup space 2# space    \ print character value
  dup out   \ send lower nibble to OUT
  1B ( esc ) = UNTIL
  ; \ exit on esc-character



\ ------------------------------------------------------------
\ Example 14
HEX
: EX14  \ -- \  display ADC at OUT and play note      mk0
  initio adc4 adcon  \ init all modules
  BEGIN
  adc@ 44 /          \ scale 10Bit value, 3FF..0 --> F...0
  dup out            \ display scaled value
      note 50 pause  \ play apropriate tone
  key? UNTIL         \ leave loop, clean up
  key drop  ton- ;



\ ------------------------------------------------------------
\ Example 15 - play MYBONNIE
\ Connect speaker between P1.5 and GND.
HEX
: EX15 \ -- \ play once
  06 note   50 pause   \ G4
  0C note   50 pause   \ F5
  0A note   50 pause   \ D5
  09 note  100 pause   \ C5
  0A note   50 pause   \ D5
  09 note   50 pause   \ C5
  07 note  100 pause   \ A4
  06 note   50 pause   \ G4
  04 lnote 100 pause   \ E4
  06 note   50 pause   \ G4
  0C note   50 pause   \ F5
  0A note  100 pause   \ D5
  09 note   50 pause   \ C5
  09 note   50 pause   \ C5
  08 note  100 pause   \ B4
  09 note   50 pause   \ C5
  0A lnote 200 pause   \ D5
  ton-  ; \ try to make it sound better ...



\ ------------------------------------------------------------
\ In DEMOLOOP you may select an example programm by its number
\ and run it. EXIT demoloop by selecting EX0.

: DEMOLOOP \ --
  BEGIN
  initio s2- s1- \ init system and reset edge detection
  F OUT       \ initial OUT value
  BEGIN
  s2? IF      \ select demo with S2
      12po 0@ 1+ F and 12po 0! \ increment OUT nibble
      out>                     \ display it
      100 ms s2-     \ wait, then reset S2
      THEN
  s1? IF      \ run demo with S1
  12po 0@
  dup  0 = IF EX0  THEN \ run example ...
  dup  1 = IF EX1  THEN
```

```
  dup  2 = IF EX2  THEN
  dup  3 = IF EX3  THEN
  dup  4 = IF EX4  THEN
  dup  5 = IF EX5  THEN
  dup  6 = IF EX6  THEN
  dup  7 = IF EX7  THEN
  dup  8 = IF EX8  THEN
  dup  9 = IF EX9  THEN
  dup  A = IF EX10 THEN
  dup  B = IF EX11 THEN
  dup  C = IF EX12 THEN
  dup  D = IF EX13 THEN
  dup  E = IF EX14 THEN
  dup  F = IF EX15 THEN
  drop s1-     \ clean up stack, reset S1
  F out        \ back to start value
  THEN
  AGAIN ;



\ --> swk4-0x.txt

\ save

\ ------------------------------------------------------------
\ todo ***
\ P0..P15 loop


decimal
unused u. \ RAM
mem u.    \ FLASH
hex .s
( finis )
```