A3 Assignment

Group members:

Jacob Rowe U12325180

Jegert Merkaj U27189696

Michael Strickland U03010237

Xzavier McKenzie U05013794

Problem Description:

The objective of this assignment is to utilize the ADC system on the MPS430 board to convert a continuous analog signal to a digital I/O signal that will be able to control the frequency in which an external LED blinks.

Pseudocode:

Initialization-

*Clear Bits

*Initialize used ports

*Configure ADC system

-Read input from POT

-Single channel, repeated

-output LED

Main Loop-

*Read ADC output digital signal

*compare reference to digital signal

*Output proper voltage to LED port

*continuous loop

Assembly Code:

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
        .cdecls C,LIST,"msp430.h"      ; Include device header file

;-------------------------------------------------------------------------------
        .def    RESET                  ; Export program entry-point to
                                       ; make it known to linker.

;-------------------------------------------------------------------------------

        .text                          ; Assemble into program memory.
        .retain                        ; Override ELF conditional linking
                                       ; and retain current section.
        .retainrefs                    ; And retain any sections that have
                                       ; references to current section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP        ; Initialize stackpointer
```

```
StopWDT    mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------
init:
                bis.b    #BIT1, &P1DIR    ;here we set P1.0 as output for the LED2
      bic.b    #BIT1, &P1OUT    ; we make sure the LED2 is OFF


      bis.b #BIT2, &P1SEL0        ;link P1.2 as ADC controller
                bis.b #BIT2, &P1SEL1 ;set P1.1 as analog input


                mov.w #0210h, &ADCCTL0      ;set ADC  sampling time
                mov.w #0220h, &ADCCTL1      ;set ADC conversion mode
                mov.w #0020h, &ADCCTL2      ;set ADC resolution
                mov.w #0002h, &ADCMCTL0  ;set ADC reference


                bic.w #0001h, &PM5CTL0       ;now we unlock the I/O low_power


;---------Set up timer block


timer:
```

```
                ; This sets up the timer. Causing a interrupt to occur every 1
second
                mov.w   #7FFFh, &TB0CCR0 ; We set the max value for compare
register
                bis.w #TBCLR, &TB0CTL          ; we set the clear bit in the control
register
                bis.w #TBSSEL__ACLK, &TB0CTL ; we select ACLK as source
                bis.w #MC__UP, &TB0CTL       ; We select continuos mode
counting
                bis.w #TBIE, &TB0CTL    ; We enable interrupt on overflow
                nop
                bis.w #GIE, SR                 ; we enable maskable interrupts
                nop


convert:
                mov.w #0213h, &ADCCTL0      ;we move digital signal to output pin
here:
                bit.w #BIT0, &ADCIFG    ;we read the digital signal from ADC
                jz here
                mov.w ADCMEM0, &TB0CCR0          ;we write trigger the interrupt
                jmp convert
                nop


ISR_TB0:
```

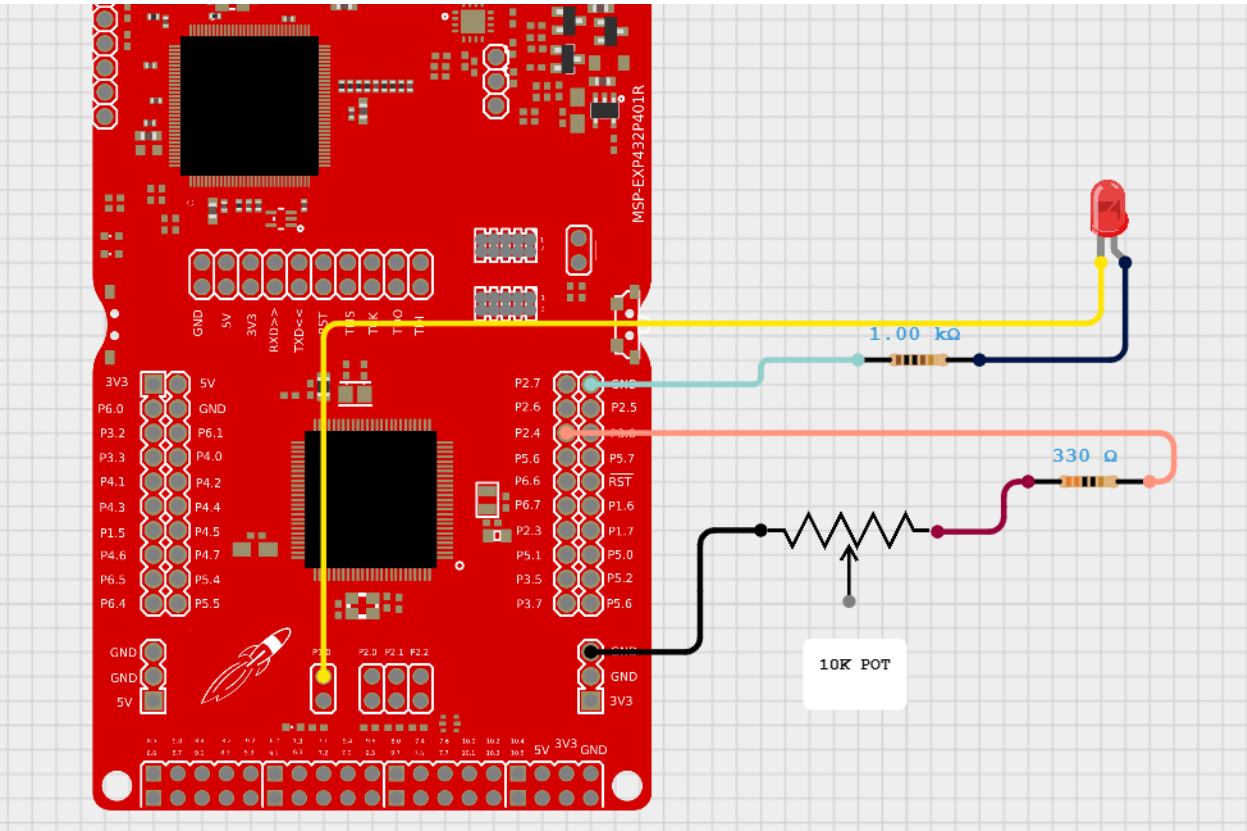
```
                    ;we enter here everytime it interrupts
    xor.b   #BIT1, &P1OUT   ;same xor trick to toggle the LED2
    bic.w   #TBIFG,   &TB0CTL   ;We make sure to clear the interrupt flag
    reti            ;we return from interrupt

;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
        .sect   ".reset"            ; MSP430 RESET Vector
        .short  RESET

        .sect   ".int42"
        .short   ISR_TB0
```

Wiring Diagram:



Video:

Additional Attachment on Canvas Submission