CET_453_ Microcomputers

Wednesday December 11th, 2024

Final Project

MICROCOMPUTERS

CET_453_ FALL 2024

To :

Professor : **Jonathan Braverman**

CET_453_ Microcomputers

From :

Van Nguyen

CET_453 Fall 2024

Microcomputer Light Show Project

Outline

1. Introduction

Objective:

- Practical application of the knowledge learned throughout the CET - 453 Microcomputers course. Using Embedded software with assembly language code, on TI Launchpad kit hardware and Breadboard with LEDs light displays.

2. System Overview

Components Used:

❖ Hardware:

- A Computer used Windows Operating System 11.
- Microcomputer board - Microcontrollers development Kit - TI Launchpad – MSP-430G2553 MCU
- A Makeronics Breadboard Raspberry Pi.
- Five LEDs from 1.8 V – 2.5 V (Blue, Green, Red, Yellow and White)
- Five resistors 330 Ω,
- Some Dupont Wire Breadboard Jumper Wires Prototype Board 20cm 54mm Pitch and a Micro Momentary Button Switch Tact Assortment.

❖ Software Environment:

- IAR Embedded software workbench MSP-430, with assembly language code.
- References :
  - MSP430 Microcontroller Basics by John Davies.
  - MSP4302xx Family Data Sheet User's Guide by Texas Instruments.
  - Lecture on Blackboard of course by Professor.
  - Final Project Specification

3. Project Requirements :

Final Project Specification

1) Design Properties
    a) The system must use the following addressing modes in their most logical and appropriate uses (see functional requirements for some uses):
        i) register mode
        ii) index mode
            (1) Putting data into an array (see 2.c below) would be a good use
        iii) absolute mode
            (1) Interacting with peripherals and hardware usually accomplishes this
        iv) indirect register mode or indirect autoincrement mode
            (1) Reading data from an array (see 2.c.iii below) would be a good use.
    b) The system must use at least one subroutine
    c) The system must use at least one interrupt service routine
    d) The system must use at least two peripherals (not including General Purpose I/O)
2) Functional Requirements
    a) The system must possess and use a watchdog timer to ensure the program does not get stuck permanently.
    b) The system must output some type of "heartbeat" indication (unless I/O limitations prevent this).
    c) The system must maintain historical data of either a key data input value or output value over a fixed time base in an array.
        i) The amount of data retained in the history must be appropriate and the frequency of storage must be appropriate for the application.
        ii) It should be possible to easily determine the most recently written data
        iii) For example if the system were a temperature monitoring/controlling system; every 30 seconds the current temperature should be placed in the array. It would be possible to determine the temperature 2 minutes ago, by looking at the location 4 records prior than the most recent data location.

❖ Hardware Requirements, detailed specifications of components.

1) - **Microcomputer board** The MSP430G2553 LaunchPad™ development Kit is cost-effective and easy-to-use evaluation module (EVM) . the ultra-low-power (3.5 - 5 V) MSP430™ microcontroller platform, including an on-board debug probe for programming, debugging and energy measurements. The board also features a push button and three LEDs for creating a simple user interface.

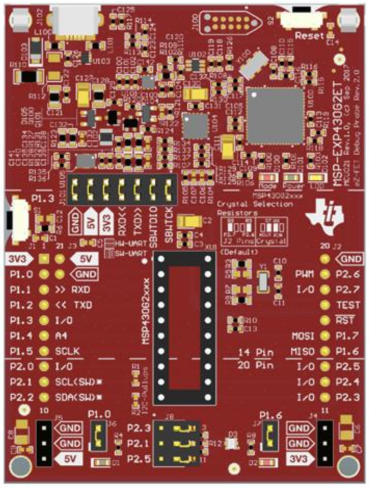
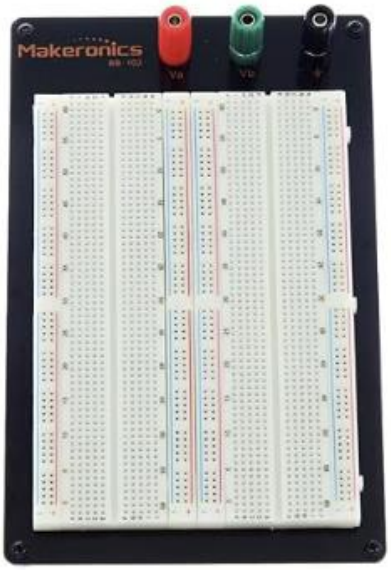


Figure 1. MSP-EXP430G2ET LaunchPad Development Kit

Features

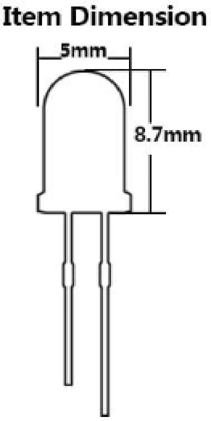
- 14-/20-pin DIP (N) Socket
- 20 pin LaunchPad standard leveraging the BoosterPack ecosystem
- On-Board EZ-FET emulator featuring EnergyTrace™ technology
- Supports devices in PDIP14 or PDIP20 packages
- 1 user buttons and 3 LEDs for user interaction.

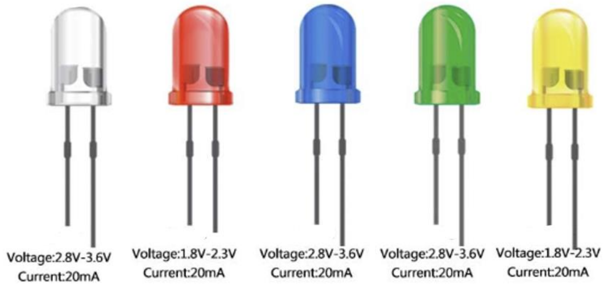
2) - **Breadboard** Makeronics Solderless for Circuit/Arduino/Raspberry Pi Prototyping Powered by Makeronics Technology with :



- ✓ 1660 Test Points In Total(including 2 terminal strips 1260 tie-point and 4 bus strips 400 tie-point) & 3 Binding Posts
- ✓ Will Accommodate Up To (18) 14pin DIP ICs, or (15)16 pin DIP ICs.
- ✓ ABS Plastic Housing, Aluminum Back Plate, Metal Contact Clips; Accept Wire With Diameter 20-29AWG;
- ✓ Binding Posts Coded Black, Red and Green, Colored Coordinates for Easy Component Placement

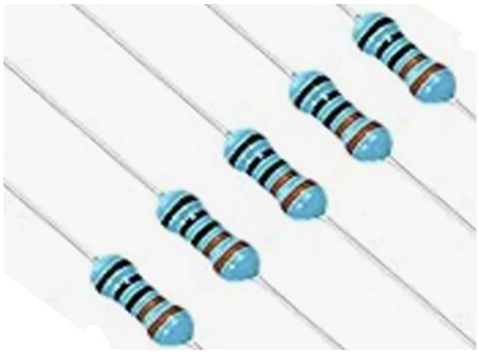
3)- **LEDs** (5) from 1.8 V – 3.6 V (White, Red, Blue, Green, and Yellow)




Voltage:2.8V-3.6V Current:20mA
Voltage:1.8V-2.3V Current:20mA
Voltage:2.8V-3.6V Current:20mA
Voltage:2.8V-3.6V Current:20mA
Voltage:1.8V-2.3V Current:20mA

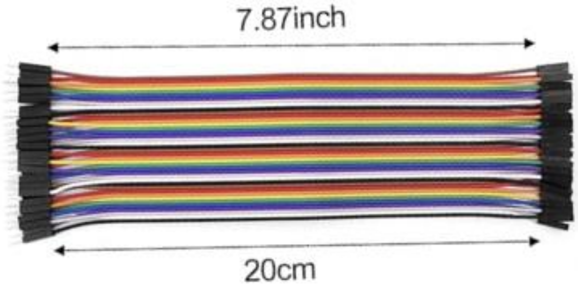
4)- **Resistor** :

- Five Resistors 330 Ω 1/4W (0.25W) Metal Film Fixed Resistor 0.01 ±1% Tolerance 330R MF Through Hole Resistors Current Limiting Rohs Certificated



5)- Wire Breadboard

Wire Breadboard Jumper Wires Prototype Board Male to Female, size 20cm 54mm Pitch



6)- Button Switch

Micro Momentary Tactile Push Button Switch Tact Assortment

❖ Software Requirements:

- Programming languages : Assembly Language code IAR Embedded software workbench MSP-430
- Libraries :
  - ✓ Class Code Examples :

    Link : data transmitter .zip (22.108 KB)

    Link : switch.txt (1.594 KB)s
  - ✓ Code Examples:

    Link : https://github.com/jonmbraverman/AssemblyExampleCode
  - ✓ LCD code and documentation:

    IAR : https://github.com/jonmbraverman/LCD-Simple

CCS: https://github.com/jonmbraverman/AssemblyExampleCode/tree/main/CCS/LCD%20Display

  - ✓ TI Code Examples :

    Link : msp430g2xx3_TI.zip (62.226 KB)
  - ✓ Link : Music_Example
  - ✓ New Code Examples :

    Course Link/Final Project/Code_Examples/Class Code Examples

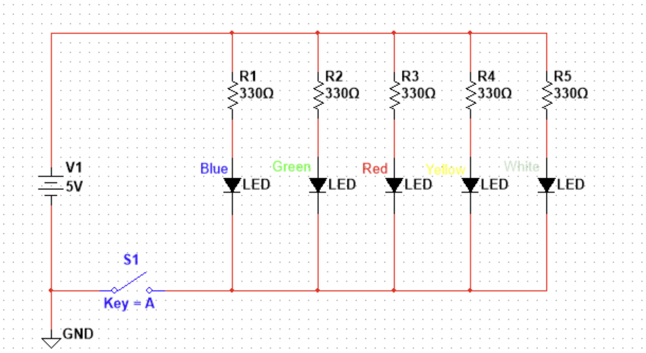
❖ Performance Goals:

- ✓ The five LEDs from 1 to 5 will light up slowly and there will be a delay of about 1 second between the LEDs before they light up.
- ✓ After all 5 LEDs light up, they will turn off and also turn off slowly, and also a delay of about 1 second between the LEDs.
- ✓ When they are completely off. They will go into a flashing mode for all the LEDs. And when they are done. We will go back to the original cycle, light up, Off and flash.

4. Design and Implementation

❖ Circuit Design:

- Block diagram of the hardware setup.



Circuit Desigh for 5 LEDs, Register and Switch

- Circuit schematic for connections.

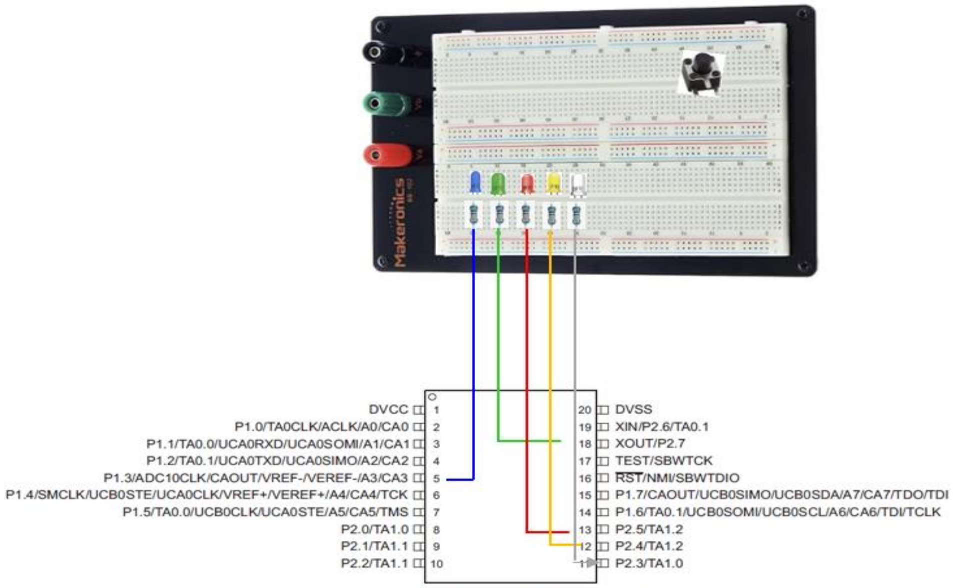

Makeronics
DVCC 1
P1.0/TA0CLK/ACLK/A0/CA0 2
P1.1/TA0.0/UCA0RXD/UCA0SOMI/A1/CA1 3
P1.2/TA0.1/UCA0TXD/UCA0SIMO/A2/CA2 4
P1.3/ADC10CLK/CAOUT/VREF-/VEREF-/A3/CA3 5
P1.4/SMCLK/UCB0STE/UCA0CLK/VREF+/VEREF+/A4/CA4/TCK 6
P1.5/TA0.0/UCB0CLK/UCA0STE/A5/CA5/TMS 7
P2.0/TA1.0 8
P2.1/TA1.1 9
P2.2/TA1.1 10
20 DVSS
19 XIN/P2.6/TA0.1
18 XOUT/P2.7
17 TEST/SBWTCK
16 RST/NMI/SBWTDIO
15 P1.7/CAOUT/UCB0SIMO/UCB0SDA/A7/CA7/TDO/TDI
14 P1.6/TA0.1/UCB0SOMI/UCB0SCL/A6/CA6/TDI/TCLK
13 P2.5/TA1.2
12 P2.4/TA1.2
11 P2.3/TA1.0

Figure 4. MSP430G2553 20-Pin N Package (Top View)

Circuit schematic for connections

❖ Software Design:

```
#include "msp430g2553.h"              ; #define controlled include file
;---------------------------------------------------------------------
        ORG     0F800h
;---------------------------------------------------------------------
init:   MOV.W   #0280h, SP                          ; set up stack
        MOV.W   #WDTPW+WDTHOLD,&WDTCTL              ; Stop watchdog timer
        MOV.W   #0001h, R14
        MOV.W   #0000h, R4
Setup Output:
        BIS.B   #0x08, &P1DIR                       ; P1.3 output
        BIS.B   #0xB8, &P2DIR                       ; P2.3, P2.4, P2.5 and P2.7 as Output
        BIC.B   #BIT6 + BIT7, &P2SEL
        BIC.B   #BIT6 + BIT7, &P2SEL2

SetupC0
        MOV.W   #CCIE,&CCTL0                        ; CCR0 interrupt enabled
        MOV.W   #50000,&CCR0                        ;
SetupTA
        MOV.W   #TASSEL_2+MC_2,&TACTL               ; SMCLK, contmode

        BIS.W   #GIE,SR                             ; enable interrupts
        MOV.W   0(R14), R4              ; copy the contents of memory specified by R14 to R4.

main:   NOP                                         ; main Loop program
        ADD.W   #0x001, R4                          ; Increment loop counter
        MOV.W  @ R4. R14
        CMP R4, R14                     ; copy the contents of memory specified by R4 to R14.
        JEQ     LED_ON

        MOV.W   #0x002, R14
        CMP     R4, R14
        JGE     LED_OFF

        JLO     LED_BLINK

LED_ON:
        BIS.B #BIT3, &P1OUT          ; P1.3 = 1 high and turn on LED
        BIS.B #BIT7, &P2OUT          ; P2.7 = 1 high and turn on LED
        BIS.B #BIT5, &P2OUT          ; P2.5 = 1 high and turn on LED
        BIS.B #BIT4, &P2OUT          ; P2.4 = 1 high and turn on LED
        BIS.B #BIT3, &P2OUT          ; P2.3 = 1 high and turn on LED

        JMP main

LED_OFF:
        BIC.B #BIT3, &P1OUT          ; P1.3 = 0 Low and turn off LED
        BIC.B #BIT7, &P2OUT          ; P2.7 = 0 Low and turn off LED
        BIC.B #BIT5, &P2OUT          ; P2.5 = 0 Low and turn off LED
        BIC.B #BIT4, &P2OUT          ; P2.4 = 0 Low and turn off LED
        BIC.B #BIT3, &P2OUT          ; P2.3 = 0 Low and turn off LED

         JMP main

LED_BLINK:
        XOR.B  #BIT3, &P1OUT         ; P1.3  LED turn Blink
        XOR.B  #BIT7, &P2OUT         ; P2.7  LED turn Blink
        XOR.B  #BIT5, &P2OUT         ; P2.5  LED turn Blink
        XOR.B  #BIT4, &P2OUT         ; P2.4  LED turn Blink
        XOR.B  #BIT3, &P2OUT         ; P2.3  LED turn Blink

        CALL    #THESUB
        NOP
        NOP
        NOP
        NOP
        JMP     init               ; jump to current location
                                   ; end loop

;-------------------------------------------------------------------------------
;           Subroutines
;-------------------------------------------------------------------------------
```

```
THESUB:
            PUSH.W  R4
            MOV.W   #0F000h, R4
            ADD.W   #00F00h, R4
            ADD.W   #000F0h, R4
            POP.W   R4
            RET


;-------------------------------------------------------------------------------
;           Interrupt Service Routines
;-------------------------------------------------------------------------------
TA0_ISR;
             XOR.B   #001h,&P1OUT          ; Toggle P1.0
             MOV.W   #0000h,&TAR           ; Reset Timer A
             RETI

;-------------------------------------------------------------------------------
;           Interrupt Vectors
;-------------------------------------------------------------------------------
            ORG     0FFFEh
            DC16    init                    ; set reset vector to 'init' label
            ORG     0FFF2h                  ; Timer_A0 Vector
            DW      TA0_ISR
            END
```

❖ Coding:

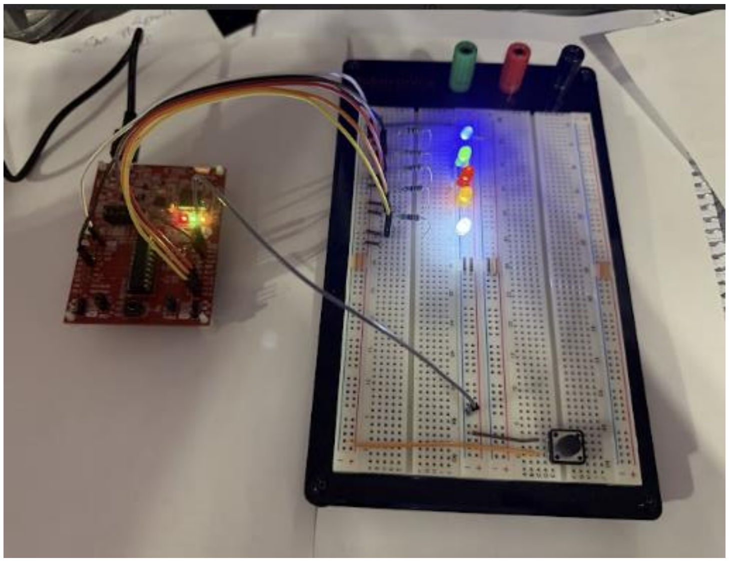
Your final project code submittal must include:

- ✓ Completed table 1 below
- ✓ PDFs of your code with line numbers showing

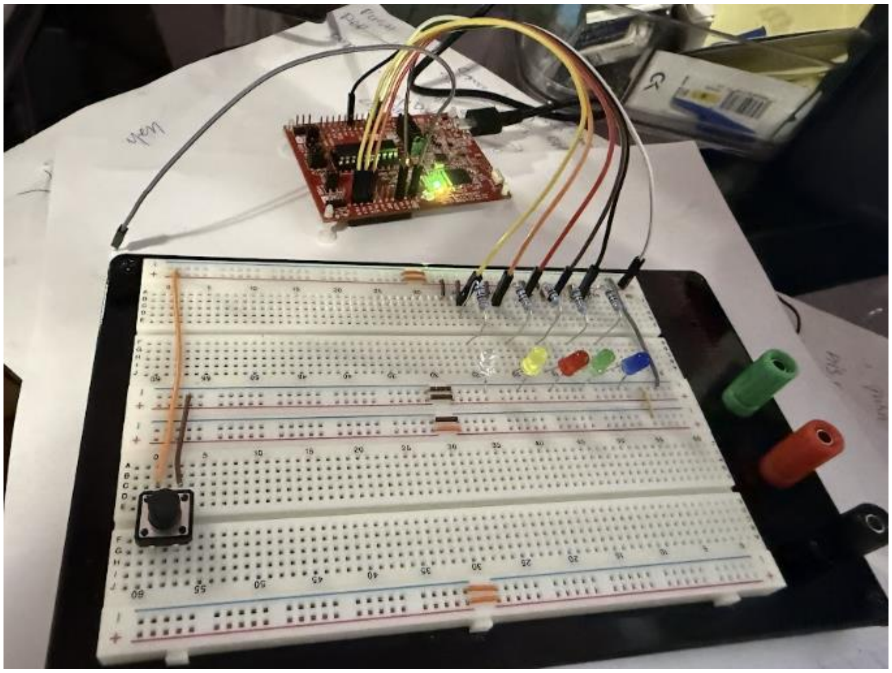
Requirement	File Name	Line Number(s)
1.a.i - Register mode addressing	MOV.W #0280h, SP	5
1.a.ii - Index addressing	MOV.W 0(R14), R4	22
1.a.iii - Absolute addressing	BIS.B #0x08, &P1DIR	10
1.a.iv - Indirect register mode or indirect autoincrement mode	MOV.W @ R4. R14	26
1.b – Subroutine	The SUB	71
1.c – Interrupt service routine	TA0_ISR	83
1.d.i – Peripheral 1	MOV.W #0000h, &TAR	85
1.d.ii – Peripheral 2	MOV.W #CCIE,&CCTL0	16
2.a – Watchdog timer resetting	MOV .W #WDTPW+WDTHOLD,&WDTCTL	6
2.b – Heartbeat indication		
2.b – Historical data storage		

5. Testing and Validation

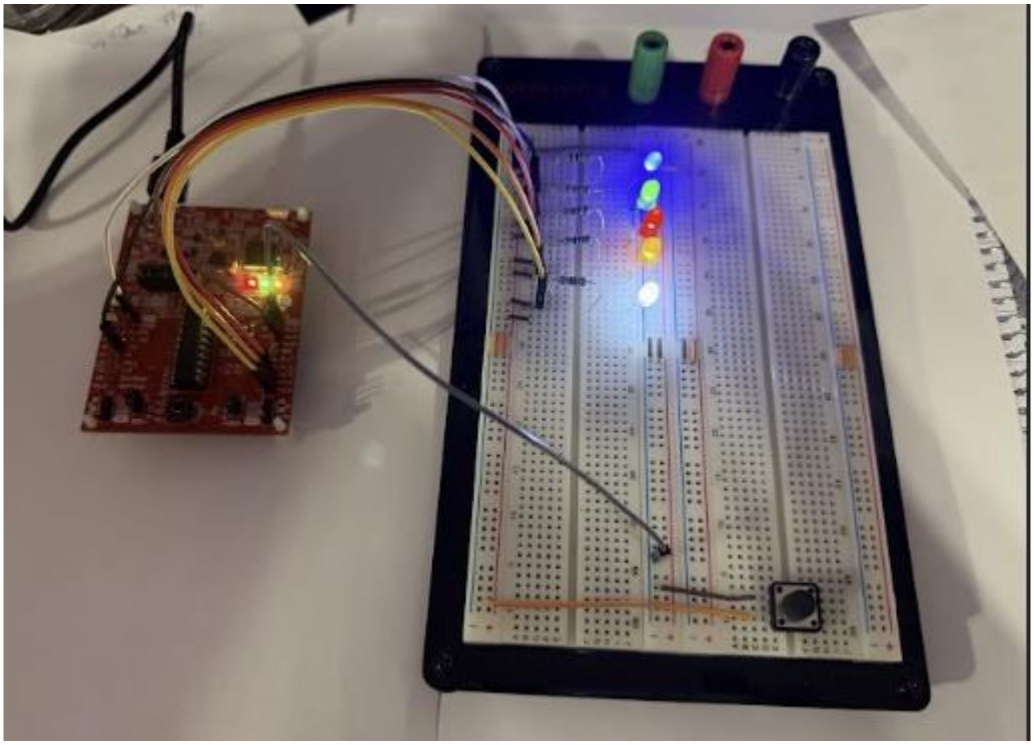
☆ **When The LED all turn on**



☆ **When The LED all turn Off**



- ☆ **When The LED are Blinking:**



Conclusion :

- ☆ When connecting the hardware and software for the Final Project, Light Show everything went as expected.