# CENG 329 PROJECT

## GROUP MEMBERS:

* Pınar Süzen 202111043
* Şevval Kaymazlı 202211039

Microprocessor Number: I31

## YOUTUBE LINK TO THE PROJECT VIDEO:

<https://youtu.be/jumWhcVw1Zk?si=jS4iXix0Wa6Zqh2f>

## SOURCE 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.

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

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

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

; Main loop here

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

**RESET:**

**bic.b** #00110000b, &P2IFG ; clear IF for next interrupt for p2.4 and p2.5

**bic.b** #00001000b, &P1IFG ; clear IF for next interrupt for p1.3

**mov.w** #\_\_STACK\_END,SP ; Initialize stackpointer

**mov.w** #0x5A80,&WDTCTL ; Stop watchdog timer

**bic.b** #10001000b, &P2OUT ; Turn all the lights off

**bic.b** #10111110b, &P1SEL ; make P1.1, 2, 3, 4, 5, and 7 Digital I/O

**bic.b** #10111110b, &P1SEL2 ; make P1.1, 2, 3, 4, 5, and 7 Digital I/O

**bic.b** #00000101b, &P2SEL ; make P2.0 and 2 Digital I/O

**bic.b** #00000101b, &P2SEL2 ; make P2.0 and 2 Digital I/O

**bis.b** #10110110b, &P1DIR ; make P1.1, 2, 4, 5, and 7 output

**bis.b** #00000101b, &P2DIR ; make P2.0 and 2 output

**bic.b** #00001000b, &P1DIR ; Make P1.3 input

**bis.b** #BIT3, &P1REN ; Enable pull-up resistor for P1.3

**bis.b** #BIT3, &P1OUT ; Enable pull-up resistor for P1.3

**bis.w** #GIE, SR ; Enable interrupts

**bis.b** #00001000b, &P1IES ; p1.3 interrupt from H to L

**bis.b** #00001000b, &P1IE ; Enable p1.3 interrupt

**bis.b** #00110000b, &P2IES ; P2.4 and p2.5 interrupts from H to L

**bis.b** #00110000b, &P2IE ; Enable p2.4 and p2.5 interrupt

;Make p2.4 and p2.5 digital I/O (buttons) p2.3 and p2.7 digital I/O (lights)

**bic.b** #10111000b, &P2SEL

**bic.b** #10111000b, &P2SEL2

;Enable pull-up resistor for the button and pull-down resistor for the light

**bic.b** #10111000b, &P2DIR

**bis.b** #10111000b, &P2REN

**bis.b** #00110000b, &P2OUT

**bic.b** #10001000b, &P2OUT

**bis.b** #10110110b, &P1OUT ; All segments OFF

**bis.b** #00000101b, &P2OUT ; All segments OFF

**mov.w** #0,r6

**Three:**

**bis.b** #10110110b, &P1OUT

**bis.b** #00000101b, &P2OUT

**bic.b** #00110110b, &P1OUT ; Turn on a,b,c,d

**bic.b** #00000100b, &P2OUT ; Turn on g

**mov.w** #4, r5 ; Approximately 1 second delay

**call** #Delay

**Two:**

**bis.b** #10110110b, &P1OUT

**bis.b** #00000101b, &P2OUT

**bic.b** #10100110b, &P1OUT ; Turn on a,b,d,e

**bic.b** #00000100b, &P2OUT ; Turn on g

**mov.w** #4, r5 ; Approximately 1 second delay

**call** #Delay

**One:**

**bis.b** #10110110b, &P1OUT

**bis.b** #00000101b, &P2OUT

**bic.b** #00010100b, &P1OUT ; Turn on b,c

**mov.w** #4, r5 ; Approximately 1 second delay

**call** #Delay

**Zero:**

**bis.b** #10110110b, &P1OUT

**bis.b** #00000101b, &P2OUT

**bic.b** #10110110b, &P1OUT ; Turn on a,b,c,d,e

**bic.b** #00000001b, &P2OUT ; Turn on f

**mov.w** #1, r5

**call** #Delay

**Dash:**

**bis.b** #10110110b, &P1OUT

**bis.b** #00000101b, &P2OUT

**bic.b** #00000100b, &P2OUT ; Turn on g

**mov.w** #5, r6 ; In order to identify dash

**call** #Check

**jmp** Dash

**Check:**

**mov.w** #12, r5 ; Approximately 3 seconds delay

**call** #Delay

**bit.b** #00001000b, &P2OUT ; Check if the green led on or not

**jne** RESET

**bit.b** #10000000b, &P2OUT ; Check if the yellow led on or not

**jne** RESET

**ret**

**Reset\_button:** ; Reset Interrupt

**jmp** RESET

**But\_ISR:** ; Interrupt for led buttons

**cmp.w** #5, r6

**jne** Fail

**Success:**

**bit.b** #00100000b, &P2IN ;read switch at p2.5

**jeq** Yellow\_on ;If p2.5 closed

**bit.b** #00010000b, &P2IN ;Read switch at p2.4

**jeq** Green\_on ;If p2.4 closed

**Fail:**

**bit.b** #00100000b, &P2IN ;Read switch at p2.5

**jeq** Green\_on ;If p2.5 closed

**bit.b** #00010000b, &P2IN ;Read switch at p2.4

**jeq** Yellow\_on ;If p2.4 closed

**Green\_on:**

**bis.b** #00001000b, &P2OUT ; Turn on the green led

**jmp** Dash

**Yellow\_on:**

**bis.b** #10000000b, &P2OUT ; Turn on the yellow led

**jmp** Dash

**Delay:**

**mov.w** #0xFFFF, r4

**Loop:**

**sub.w** #1, r4

**cmp.w** #0, r4

**jne** Loop

**Out\_loop:**

**sub.w** #1, r5

**cmp.w** #0, r5

**jne** Delay

**ret**

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

; Stack Pointer definition

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

**.global** \_\_STACK\_END

**.sect** .stack

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

; Interrupt Vectors

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

**.sect** ".int03" ; Port 2 interrupt vector

**.short** But\_ISR

**.sect** ".int02" ; Port 1 interrupt vector

**.short** Reset\_button

**.sect** ".reset" ; MSP430 RESET Vector

**.short** RESET ; Actually int15

.end

## EXPLANATION

### Main Loop:

Reset: Interrupt flags are cleared, the ports to be utilized in the program are initialized, the stack pointer is reset to the starting position, the watchdog timer is being stopped, and all lights are turned off. Ports P1.1, P1.2, P1.3, P1.4, P1.5, P1.7, P2.0, and P2.2 are configured as digital I/O, while P1.1, P1.2, P1.4, P1.5, P1.7, P2.0, and P2.2 are set as output ports and P1.3 is configured as an input. Additionally, P2.4 and P2.5 are enabled in order to be used as buttons, whereas P2.3 and P2.7 are enabled in order to be used as lights. Pull-up resistors are utilized for the button ports, whereas pull-down resistors are used for the light ports. Finally, all segments are turned off, and the register R6, used to identify the dash, is initialized to zero.

Three: Firstly, all segments are turned off. Then the necessary segments (a, b, c, d, g) are turned on so that the number "3" is shown on the 7-segment display. After that, “Delay” subroutine is called to create an approximately 1 second delay between numbers.

Two: Firstly, all segments are turned off. Then the necessary segments (a, b, d, e, g) are turned on so that the number "2" is shown on the 7-segment display. After that, “Delay” subroutine is called to create an approximately 1 second delay between numbers.

One: Firstly, all segments are turned off. Then the necessary segments (b, c) are turned on so that the number "1" is shown on the 7-segment display. After that, “Delay” subroutine is called to create an approximately 1 second delay between numbers.

Zero: Firstly, all segments are turned off. Then the necessary segments (a, b, c, d, e, f) are turned on so that the number "0" is shown on the 7-segment display. After that, “Delay” subroutine is called to create an approximately 1 second delay between numbers.

Dash: Firstly, all segments are turned off. Then the necessary segment (g) is turned on so that the “-” symbol is shown on the 7-segment display. After that, a number is assigned to the register R6 to identify the game state later in the “but\_ISR” interrupt. Then the “Check” subroutine is called to determine whether one of the players pressed their buttons designated to light up the leds or not. “-” symbol will be displayed until one of the players presses their button.

Check: First of all, the program waits for 3 seconds. After the three-second delay, if neither player presses any button or the button has not been pressed, the program remains in a standby state while displaying the symbol “-”, awaiting a press signal to start an another round by going back to the “RESET” label.

Reset\_button: The program transitions to this label through an interrupt triggered by the designated port P1.3, which is the button on the MSP430 microprocessor, within the Port 1.

But\_ISR: The program transitions to this label through an interrupt triggered by the designated ports P2.5 or P2.4, which is for the buttons designated to control the leds, within the Port 2. After that, if the register R6 is not equal to 5, indicating that the program is not at the “Dash” and the game is not finished yet, the program transitions to the label “Fail.” Conversely, if R6 equals 5, the program proceeds to execute and transitions to the label “Success.”

Success: The program checks the button ports to determine which button is pressed (the program considers only the first pressed button), then turns on the led connected to the pressed button.

Fail: The program checks the button ports to determine which button is pressed (the program considers only the first pressed button), then turns on the opponent of the defeated player’s led.

Green\_on: Green led is turned on. Then the program jumps to the “Dash” label.

Yellow\_on: Yellow led is turned on. Then the program jumps to the “Dash” label.

Delay: This subroutine enables the program to wait for a specified duration, determined by a pre-initialized value stored in the register R5.

Interrupt Vectors:

**.sect** ".int03"

**.short** But\_ISR : Specifies the address of the ISR for Port 2 interrupts. When a Port 2 interrupt is triggered, the microcontroller jumps to the “But\_ISR” label.

**.sect** ".int02"

**.short** Reset\_button : Specifies the address of the ISR for Port 1 interrupts. When a Port 1 interrupt is triggered, the microcontroller jumps to the “Reset\_button” label.

**.sect** ".reset"

**.short** RESET : Specifies the address of the program's main entry point labeled “RESET”.