# Report: Reflex Game with MSP430G2553 Microcontroller

---


### Group Members

- Deniz Arda ÇINARER – 202211019
- Ege Arca TUNÇ – 202211067
- Yusuf ÇAĞLAYAN – 202211016


### YouTube Link

Watch the working project

https://www.youtube.com/shorts/YgvDWZVRXS8

## Overview

This project implements a two-player reflex game using the MSP430G2553 microcontroller. The game tests players' reaction times based on visual cues displayed on a 7-segment display. Players interact with the system using buttons, and the results are indicated through LEDs.

The game starts with a countdown displayed on the 7-segment display (3 → 2 → 1 → "-"). Players must press their respective buttons only when the dash ("-") symbol is displayed. If a button is pressed prematurely (during 3, 2, or 1), the other player wins, and their LED lights up. If both players wait until the dash, the first player to press their button wins. Each session resets automatically after 3 seconds, or players can manually reset the game using a designated reset button.

## Hardware Components

- **Microcontroller**: MSP430G2553
- **7-Segment Display**: Displays the countdown (3 → 2 → 1 → "-").
- **Player Buttons**:
  - Player 1: Connected to P1.2 (interrupt-enabled).
  - Player 2: Connected to P2.5 (interrupt-enabled).
- **LEDs**:
  - Player 1's Win Indicator: Connected to P2.7.
  - Player 2's Win Indicator: Connected to P2.4.
- **Reset Button**: Connected to P1.3 (non-interrupt).
- **Resistors and Jumpers**

## System Behavior

1. **Countdown Mechanism**:
    - The 7-segment display sequentially shows 3, 2, 1, and "-".
    - Each number is displayed for 1 second using a delay loop implemented in assembly (`oneSecondDelay`).
    - At the end of the countdown, the dash ("-") is displayed, signaling players to press their buttons.
2. **Player Button Inputs**:
    - Player 1 Button (P1.2) and Player 2 Button (P2.5) are interrupt-driven.
    - If a button press is detected:
        - **During "3", "2", or "1"**: The other player automatically wins.
            - For example:
                - If Player 1 presses the button during "3", Player 2's LED (P2.4) lights up.
                - Similarly, if Player 2 presses during "1", Player 1's LED (P2.7) lights up.
        - **During "-"**: The first player to press wins.
3. **LED Indicators**:
    - P2.7 lights up for Player 1's victory.
    - P2.4 lights up for Player 2's victory.
    - LEDs turn off at the start of each game session.
4. **Reset Mechanism**:
    - **Automatic Reset**: After a game session ends, the system waits for 3 seconds before starting a new session.
    - **Manual Reset**: Players can press the reset button (P1.3) to restart the game immediately.
5. **Interrupt Handling**:
    - **PLR_1 (P1.2)** and **PLR_2 (P2.5)** are interrupt service routines (ISRs) that:
        - Check if the button press occurred at the correct time (during "-").
        - Light the appropriate LED if conditions are met.
        - Clear interrupt flags to allow subsequent interrupts.

## Code Explanation

1. **Initialization**:
    - **Stop Watchdog Timer**: Prevents unintentional resets.
    - **Pin Configuration**:
        - Pins for LEDs and 7-segment segments are set as outputs.
        - Pins for buttons are set as inputs with pull-up resistors enabled.
2. **Countdown Sequence**:
    - **GameMaster**: Main routine that orchestrates the countdown and game flow.
    - The countdown logic lights up specific segments for each number:
        - **3, 2, 1**: Displays the respective countdown numbers for 1 second each.
        - **Dash ("-")**: Indicates the reaction phase where players can press their buttons.
3. **Delay Implementation**:

    - The `oneSecondDelay` subroutine creates an approximate 1-second delay using nested loops.
    - During the delay, the reset button (P1.3) is periodically checked to allow an immediate manual reset.
4. **Interrupt Service Routines**:
    - **PLR_1 (P1.2)**: Handles Player 1's button press:
        - Checks if the press occurred during the countdown or during "-".
        - Lights Player 2's LED (P2.4) if the button was pressed prematurely.
        - Lights Player 1's LED (P2.7) if the button was pressed at the correct time.
    - **PLR_2 (P2.5)**: Same functionality as PLR_1 but for Player 2.
5. **Game Reset**:
    - At the end of each game:
        - LEDs and display segments are turned off.
        - Counters and flags are reset.
        - A new countdown begins after 3 seconds or immediately if manually reset.

## Conclusion

This reflex game effectively demonstrates interrupt-driven programming and multi-pin control using the MSP430G2553 microcontroller. The system is robust, providing reliable gameplay with clear visual feedback through the 7-segment display and LEDs. The combination of automatic and manual reset mechanisms ensures smooth operation across sessions.

## Attachments

- **Source Code**: Included in the document.
- **YouTube Video**: Link.

## 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.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stack pointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer

;-------------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------------
        mov.w #0, r4

        bic.b #11111110b, &P1SEL  ; make P1.1, 2, 3, 4, 5, 6 and 7 Digital I/O
        bic.b #11111110b, &P1SEL2 ; make P1.1, 2, 3, 4, 5,  6 and 7 Digital I/O
        bic.b #11110101b, &P2SEL  ; make P2.0 and 2.2 , 2.4 , 2.5 , 2.6 and 2.7 Digital I/O
        bic.b #11110101b, &P2SEL2 ; make P2.0 , 2.2 , 2.4 , 2.5 and , 2.6 , 2.7 Digital I/O

        bis.b #10110010b, &P1DIR ; make P1.1, 1.4, 1.5, and 1.7 output
        bis.b #11010101b, &P2DIR ; make P2.0 , 2.2 , 2.4 , 2.6, 2.7 output

        bis.b #10110010b, &P1OUT ; All segments OFF
        bis.b #01000101b, &P2OUT ; All segments OFF

        bic.b #00001000b, &P1DIR ; make P1.3 input
        bis.b #00001000b, &P1REN ; enable pull-up resistor for P1.3
        bis.b #00001000b, &P1OUT ; use pull up resistor for manuel reset button

        bic.b #00100000b, &P2DIR ; Make P2.5 Input
        bis.b #00100000b, &P2REN ; enable pull-up resistor for P2.5
        bis.b #00100000b, &P2OUT ; use pull up resistor for player2 button

        bic.b #00000100b, &P1DIR ; make P1.2 input
        bis.b #00000100b, &P1REN ; enable pull-up resistor for P1.2
        bis.b #00000100b, &P1OUT ; use pull up resistor for player1 button

        bis.w #GIE, SR                  ; enable interrupts

        bic.b #00000100b, &P1IFG
        bic.b #00100000b, &P2IFG ; clear flags

        bis.b #00000100b, &P1IES ; p1.2 interrupts from H to L
        bis.b #00000100b, &P1IE  ; enable p1.2 interrupt

        bis.b #00100000b, &P2IES ; p2.5 interrupts from H to L
        bis.b #00100000b, &P2IE  ; enable p2.5 interrupt


        bic.b #BIT4|BIT7, &P2OUT ; Turn off LEDs on 2.4 and 2.7
```

**GameMaster:**

```
        mov.w #0, r4                    ; r6 will be used to see which number is game currently
on
        mov.w #0, r6            ; r4 will be used to check if any player wins the game

        CALL #oneSecondDelay
        cmp.w #1, r8                ; comparison mechanism for manuel game reset
        jeq gameOn
        CALL #oneSecondDelay
        cmp.w #1, r8
        jeq gameOn
        CALL #oneSecondDelay    ; 3 seconds delay after each game session.
        cmp.w #1, r8
        jeq gameOn

gameOn:                                         ; starting the game manually or
automatically
        mov.w #0, r8
        bic.b #BIT4|BIT7, &P2OUT ; Turn off LEDs on 2.4 and 2.7

        bis.b #10110010b, &P1OUT ; turn off all lights. - for numbers
        bis.b #01000101b, &P2OUT ; turn off all lights. - for numbers

number3:
        mov.w #3, r6                    ; currently number 3

        cmp.w #1, r4
        jeq GameMaster          ; if any of the players win go to the main menu
        cmp.w #2, r4
        jeq GameMaster

        bis.b #10110010b, &P1OUT ; turn off all lights. for numbers
        bis.b #01000101b, &P2OUT ; turn off all lights. for numbers

        bic.b #BIT1|BIT4|BIT5, &P1OUT
        bic.b #BIT2|BIT6, &P2OUT ; turn on lights for number 3

        CALL #oneSecondDelay        ; one second delay for each number
        cmp.w #1, r8
        jeq gameOn

        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        jmp number2

number2:
        mov.w #2, r6                    ; currently number 2
```

```
        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        bis.b #10110010b, &P1OUT ; turn off all lights. for numbers
        bis.b #01000101b, &P2OUT ; turn off all lights. for numbers

        bic.b #BIT1|BIT5|BIT7,&P1OUT
        bic.b #BIT2|BIT6,&P2OUT  ; turn on lights for number 2

        CALL #oneSecondDelay
        cmp.w #1, r8
        jeq gameOn

        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        jmp number1

number1:
        mov.w #1, r6                    ; currently number 1

        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        bis.b #10110010b, &P1OUT ; turn off all lights. for numbers
        bis.b #01000101b, &P2OUT ; turn off all lights. for numbers

        bic.b #BIT1|BIT4,&P1OUT  ; turn on lights for number 1

        CALL #oneSecondDelay
        cmp.w #1, r8
        jeq gameOn

        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        jmp dash


dash:
        mov.w #0, r6                    ; currently on dash
```

```
        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        bis.b #10110010b, &P1OUT ; turn off all lights. for numbers
        bis.b #01000101b, &P2OUT ; turn off all lights. for numbers

        bic.b #BIT2,&P2OUT              ; turn on lights for a dash

        CALL #oneSecondDelay
        cmp.w #1, r8
        jeq gameOn
        CALL #oneSecondDelay
        cmp.w #1, r8
        jeq gameOn
        CALL #oneSecondDelay    ; 3 seconds delay for dash
        cmp.w #1, r8
        jeq gameOn

        cmp.w #1, r4
        jeq GameMaster
        cmp.w #2, r4
        jeq GameMaster

        jmp GameMaster                  ; go to the main menu for a new session if no one
hit the buttons

oneSecondDelay:                 ; 1 second delay
        mov.w #0, r8
        mov.w #0xFFFF, r5
Delay:
        mov.w #1, r9
        mov.w #3, r9                    ; to arrange this delay to a second..
        bit.b #00001000b , &P1IN  ; always check here if anyone hit the reset button
        jeq go
        sub.w #1,r5
        jne Delay
        jmp return

go:
        mov.w #1,r8

return: ; return back
        ret

PLR_1:                                          ; first player button interrupt p1.2 port
        cmp.w #2, r4                    ; if player2 already won
        jeq ifPlayer2Won
```

```
        cmp.w #3, r6
        jeq player2Winsbynotpressing

        cmp.w #2, r6
        jeq player2Winsbynotpressing

        cmp.w #1, r6
        jeq player2Winsbynotpressing  ; if player 1 presses the buton in the wrong number

        cmp.w #0, r6                        ; if player 1 wins by pressing it on dash
        jeq player1Winsbypressing

player1Winsbypressing:
        mov.w #1, r4
        bis.b #BIT7, &P2OUT     ; Turn LED on 2.7 ON ( player1 win.)
        jmp finish1

ifPlayer2Won:                           ; do nothing
        jmp finish1

player2Winsbynotpressing:
        mov.w #2, r4
        bis.b #BIT4, &P2OUT         ; Turn LED on 2.4 ON ( player2 win.)
        jmp finish1

finish1:
        bic.b #00000100b, &P1IFG   ; clear the flag and return
        reti



PLR_2:                          ; second player button interrupt p2.5 port
        cmp.w #1, r4 ; if player1 already won
        jeq ifPlayer1Won

        cmp.w #3, r6
        jeq player1Winsbynotpressing

        cmp.w #2, r6
        jeq player1Winsbynotpressing

        cmp.w #1, r6
        jeq player1Winsbynotpressing ; if player 2 presses the buton in the wrong number

        cmp.w #0, r6                        ; if player 2 wins by pressing it on dash
        jeq player2Winsbypressing

player1Winsbynotpressing:
        mov.w #1, r4
        bis.b #BIT7, &P2OUT       ; Turn LED on 2.7 ON ( player1 win.)
```

```
        jmp finish2

ifPlayer1Won: ; do nothing
        jmp finish2

player2Winsbypressing:
        mov.w #2, r4
        bis.b #BIT4, &P2OUT ; Turn LED on 2.4 ON ( player2 win.)
        jmp finish2

finish2:
        bic.b #00100000b, &P2IFG ; clear flag and return
        reti
;-------------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack

;-------------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------------
        .sect ".int02"
        .short PLR_1

        .sect ".int03"
        .short PLR_2

        .sect ".reset"
        .short RESET
```