EE306 - Microprocessors

Laboratory Exercise 5
Timer

April 29, 2020

Erdal Sidal Dogan
#041702023

1 Task1

In this task a LED had been switched on/off with .25 second intervals, which was accurately measured using *A9 Private Timer*.

Listing 1: Assembly code of Task-I

```
.global _start
_start:
        LDR R12, =0xFF200000 //LED adress
        MOV R11, #1 //LED value
        LDR R1, =0xFFFEC600 //timer adress
        LDR R2, =50000000 // 1/4th of the 200M
        STR R2, [R1] //set load register of the timer
        MOV R3, #0b011 //control configuration
        STR R3, [R1, #8] //set config bits

LOOP:
        STR R11, [R12] //turn on
        BL DELAY
        EOR R11, R11, #1 //switch the state of led
        B LOOP

DELAY:
        LDR R4, [R1, #0xC] // read the interrupt status bit
        CMP R4, #1 // 1 when load register reaches 0
        BNE DELAY
        STR R4, [R1, #0xC] // reset interrupt status bit
        BX LR

.end
```

2 Task2

Click here for video explanation of Task 2

This task is the implementation of a clock using 7-segment displays. *A9 Private Timer* is used for accurate timing. The clock counts up to 59.99 seconds then resets. Since the each digit between 0-9 has a unique representation in 7-segment display, these representations are being held in memory location, which the `SEQUENCE` points the address of.

Listing 2: Assembly code of Task-II

```
.global _start
_start:
    LDR R0, =0xFFFEC600 //timer adress
    LDR R1, =2000000 // 1/100th of the 200M
    STR R1, [R0] //set load register of the timer
    MOV R2, #0b011 //control configuration
    STR R2, [R0, #8] //set config bits

    LDR R1, =0xFF200020 // 7-segment address
    LDR R2, =0xFF200050 // push button address
    LDR R4, =SEQUENCE // sequential digits
    B SET_TO_ZERO // start from zero
```

```
LOOP:
    BL HOLD_ON // checks if it should wait or not
    BL INCREMENT_MS // increment miliseconds
    B LOOP

INCREMENT_MS:
    BL DELAY
    LDRB R6, [R4, R5] //
    STRB R6, [R1] // rightmost digit
    LDRB R7, [R4, R10]
    STRB R7, [R1, #1] // second to right
    ADD R5, R5, #1
    CMP R5, #10
    ADDEQ R10, R10, #1
    MOVEQ R5, #0
    CMP R10, #10
    MOVEQ R10, #0
    BEQ INCREMENT_SEC
    B LOOP

INCREMENT_SEC:
    LDRB R8, [R4, R11]
    STRB R8, [R1, #3] // third to left
    LDRB R9, [R4, R12]
    STRB R9, [R1, #0b10000] // second to left
    ADD R11, R11, #1
    CMP R11, #10
    MOVEQ R11, #0
    ADDEQ R12, R12, #1
    CMP R12, #6
    BLEQ SET_TO_ZERO
    BX LR

DELAY:
    LDR R3, [R0, #0xC]
    CMP R3, #1
    BNE DELAY
    STR R3, [R0, #0xC] // reset status flag.
    BX LR

SET_TO_ZERO:
    MOV R5, #0 // array index
    MOV R10, #0
    MOV R11, #0
    MOV R12, #0
    B LOOP

HOLD_ON:
    LDR R6, [R2]
    CMP R6, #0
    BNE HOLD_ON
    BX LR
```

```
SEQUENCE: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0xFF, 0xEF
// 0 = 0x3F, Array Index: 0
// 1 = 0x06, Array Index: 4
// 2 = 0x5B, Array Index: 8
// 3 = 0x4F, Array Index: C
// 4 = 0x66, Array Index: 10
// 5 = 0x6D, Array Index: 14
// 6 = 0x7D, Array Index: 18
// 7 = 0x07, Array Index: 1C
// 8 = 0xFF, Array Index: 20
// 9 = 0xEF, Array Index: 24
.end
```