GAZI UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING

Experiment #5

Ahmet Emin Karakaya
191110046

Code description

In Figure 1, I defined the base addresses of ports A and D.

```
1  ; GPIO base addresses of port A and D
2  gpioDbase    EQU      0x4005B000 ; inp
3  gpioAbase    EQU      0x40058000 ; out
4
```

Figure 1

In Figure 2, I have defined different frequency values. In fact, in the experiment, we were asked to activate the buzzer with different waiting times. That's why these values are actually waiting times. I will also change these values to 3 and 5 to be able to observe more easily in the simulation. Then I defined an address to keep the delay times at an address.

```
50
51 high_frequency_value EQU 0X4C0000  ; high_frequency_value defined(When I observe in the simulation I am making the value 3 of it. To be able to observe.)
52 low_frequency_value  EQU 0X0C0000  ; low_frequency_value defined(When I observe in the simulation I am making the value 5 of it. To be able to observe.)
53 delay_address  EQU 0x20000400      ; Delay_address defined
54
```

Figure 2

In Figure 3, I controlled the button on bit 0 of port D. If the button is not pressed, it will go to the low_frequency branch. If pressed, it will go to the high_frequency branch. Here, I also wrote the 71st line to test it in the simulation and accepted that the button was pressed. In normal code this is not valid.

```
66
67 checkSwitch
68                LDR R1, =gpioDbase
69                ADD R1, #0x04
70                LDR R0, [R1]          ; data read in R0
71                ;MOV R0, #1
72
73                CMP R0, #0            ;Button status checked
74                BEQ low_frequency     ; If the button state is logic 0, low_frequency branch has been moved
75                BNE high_frequency    ; If the button state is not logic 0, high_frequency branch has been moved
76
77
```

Figure 3

In Figure 4, wrote the high_frequency value to delay_address and went to the buzzer_ringing branch to ring the buzzer. The purpose here is actually to write the value given in the middle to the registers to make the buzzer sound and extinguish.

```
78
79  high_frequency
80                  LDR R2, =high_frequency_value  ; High_frequency_value written to register R2
81                  LDR R3, =delay_address         ; Delay_address written to R3 register
82                  STR R2, [R3]                   ; High_frequency_value is written to delay address
83                  B buzzer_ringing               ; Going to buzzer_ringing branch
84
```

Figure 4

In Figure 5, wrote the low_frequency value to delay_address and went to the buzzer_ringing branch to ring the buzzer. The purpose here is actually to write the value given in the middle to the registers to make the buzzer sound and extinguish.

```
85
86  low_frequency
87                  LDR R2, =low_frequency_value   ; Low_frequency_value written to register R2
88                  LDR R3, =delay_address         ; Delay_address written to R3 register
89                  STR R2, [R3]                   ; Low_frequency_value is written to delay address
90                  B buzzer_ringing               ; Going to buzzer_ringing branch
91
```

Figure 5

In Figure 6, the code changes bit 6 of port A to logic 1, and after waiting a certain time, to logic 0. It first goes to the address on bit 6 of the A pot and takes the value in pr. Then it performs ORR operation and activates it by transferring it to the address. It waits for a certain amount of time in this state. After it comes without waiting, it makes logic 0 with the BIC directive and the sound is cut off. Likewise, it goes to standby and after coming from there, it goes to the checkSwitch branch to check the button on port D.

```
91
92  buzzer_ringing
93                  LDR R1, =gpioAbase
94                  ADD R1, #0x100              ; R1 stores address of data reg port A (make sure not to change in any branch)
95                  LDR R0, [R1]
96
97                  ORR R0, R0, #0x01
98                  STR R0, [R1]                ; Makes Buzzer's state logic 1. So it makes it ring.
99                  BL delay
100
101                 LDR R0, [R1]
102
103                 BIC R0, R0, #0x01
104                 STR R0, [R1]                ; Makes Buzzer's state logic 0. So it makes it doesn't ring.
105                 BL delay
106
107                 B checkSwitch               ; Button status went to checkSwitch branch to recheck
108
```

Figure 6

In Figure 7, the delay time transferred to the R3 register is read and written to the R4 register.

```
109
110 delay
111             LDR R4, [R3]                ; The value at address R3, which holds the delaying time, is written to register R4
112
```

Figure 7

In Figure 8, it waits until the waiting time in the R4 register becomes 0 and the waiting process is done.

```
112
113 wait
114             SUB R4, R4, #1              ; Delay time reduced by 1
115             CMP R4, #0                  ; Comparing whether the delay time is 0
116             BNE wait                    ; If not 0 it went back to the beginning of the loop
117             BX  LR
118
```

Figure 8

Simulation outputs

In Figure 9, the code checked the button status in the checkSwitch branch and saw the button status as 1 (R0 register). It then went to the high_frequency branch.

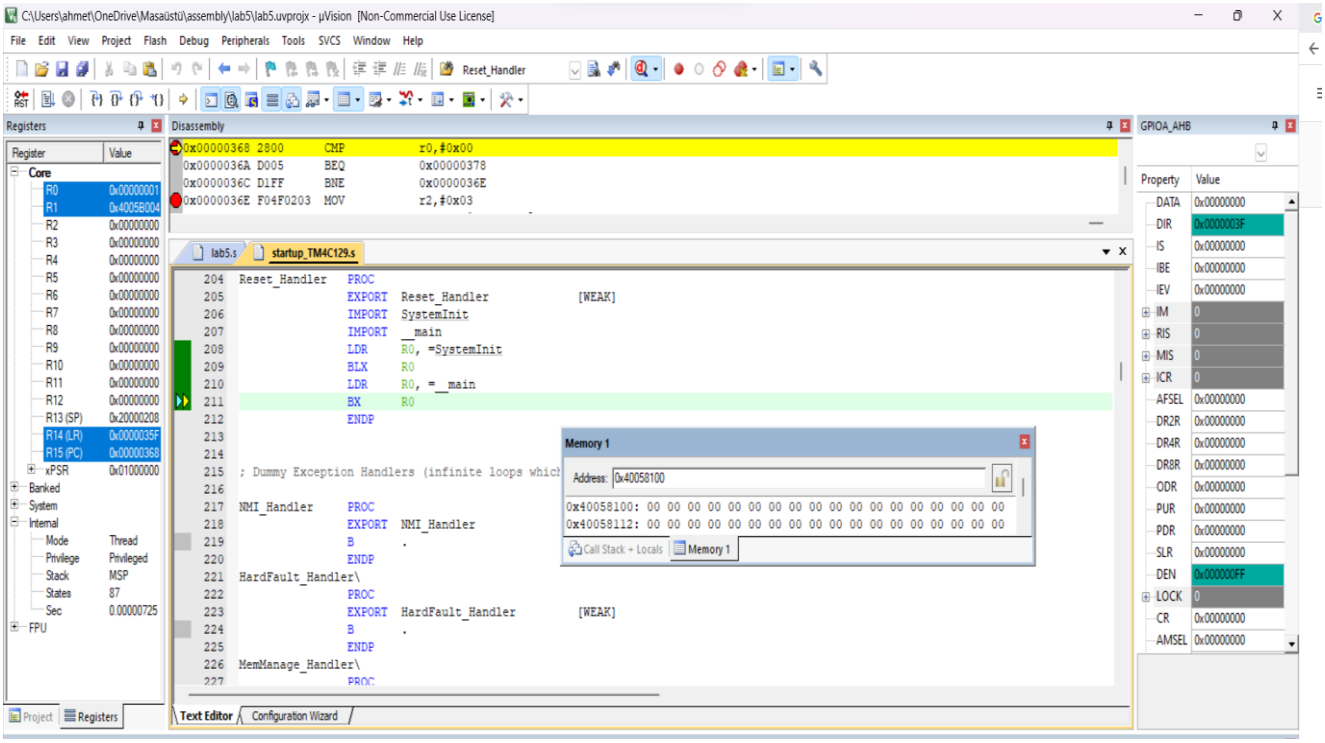

Figure 9

In Figure 10, the code went to the high_frequency branch and wrote the high_frequency_value to the R2 register. Observe in the simulation I made this value 3.

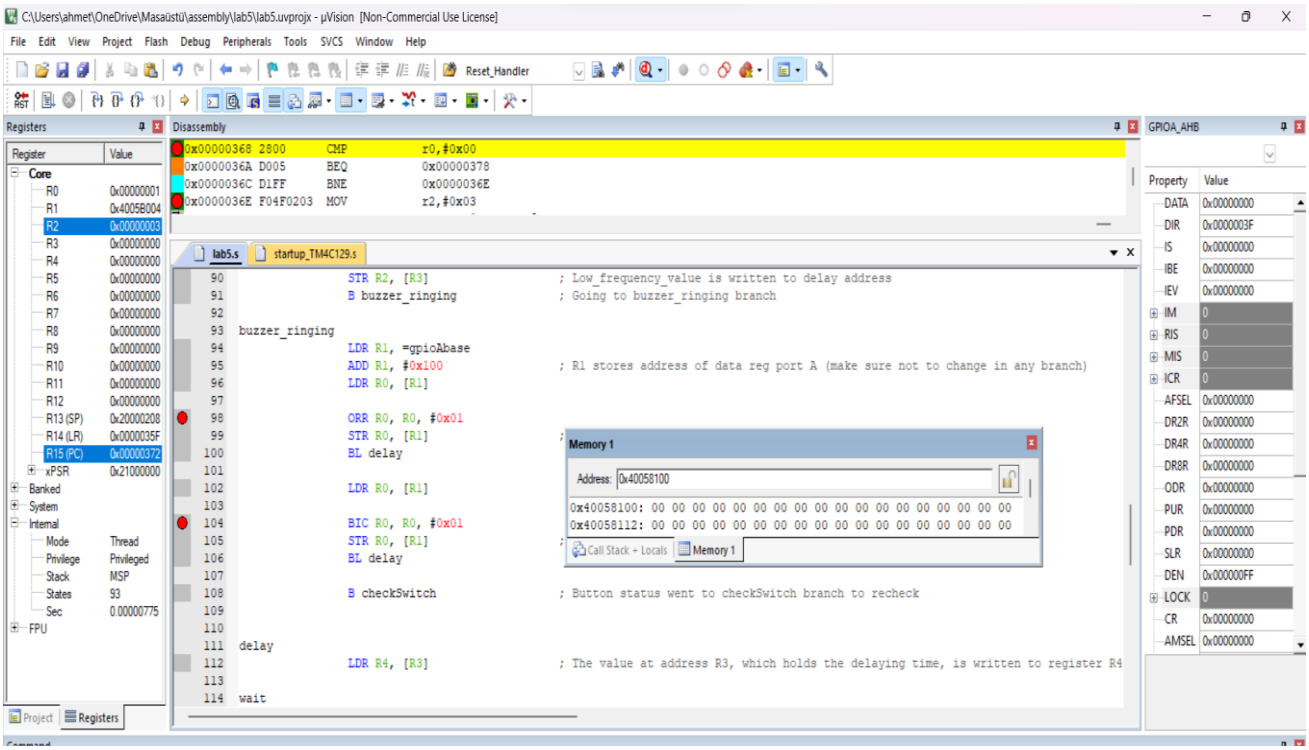

Figure 10

In Figure 11, the code went to the buzzer_ringing branch after registering and writing to the address. There he wrote a 1 to the 6th bit of the A pot. It actually activated the buzzer.

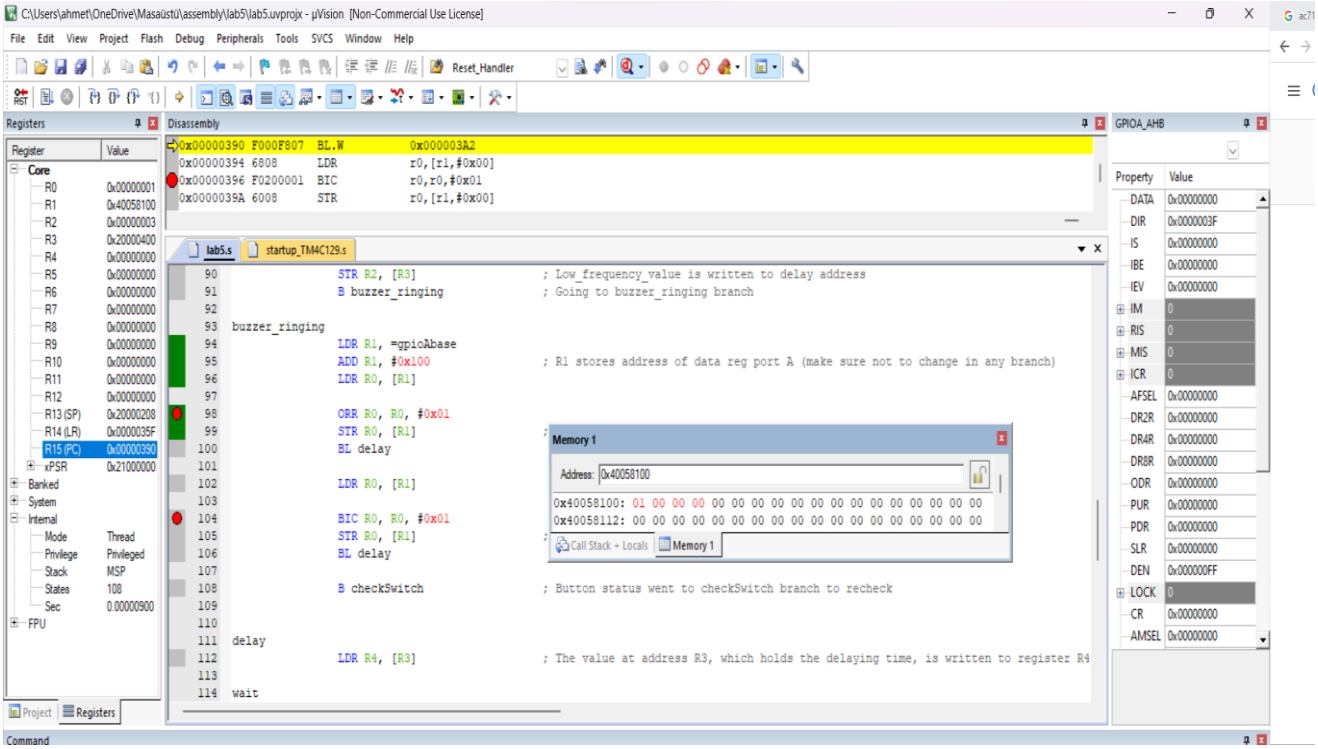

Figure 11

In figure 12, the code went to the delay branch and wrote 3 to register R4 and then went to the wait branch. It waited in the wait branch until the value of 3 became 0.

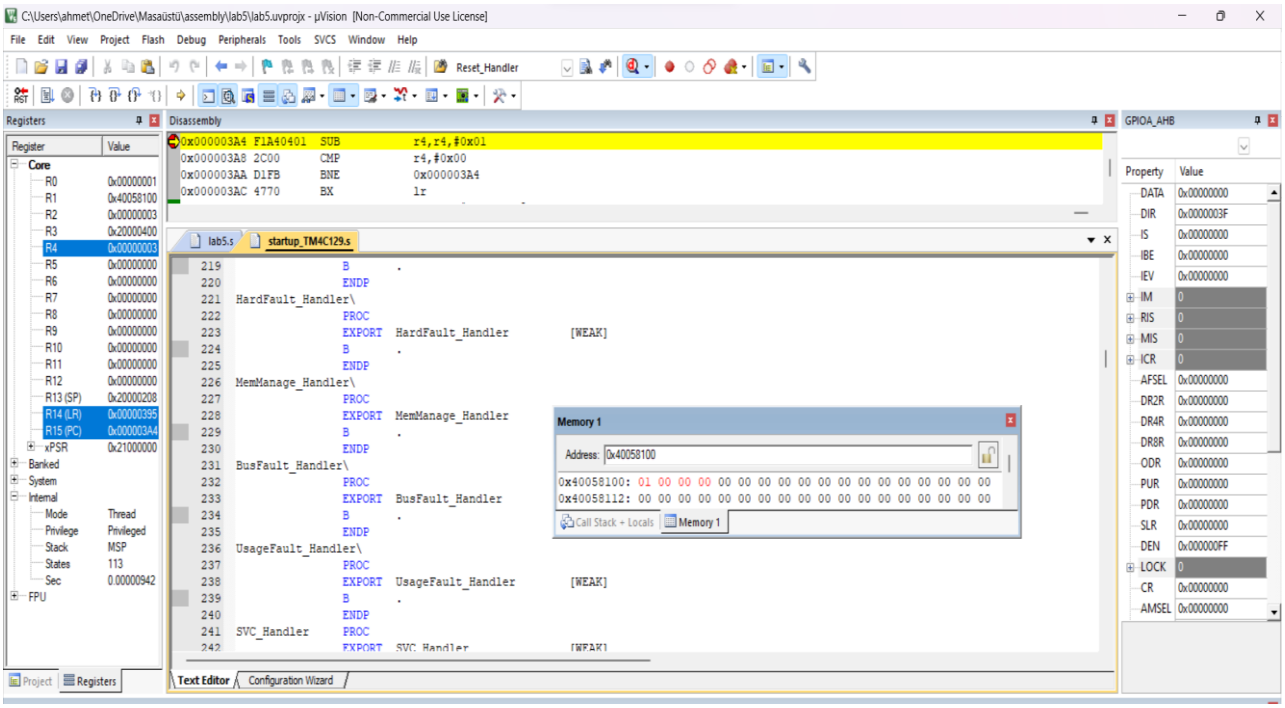

Figure 12

In Figure 13, register R4 became 2.

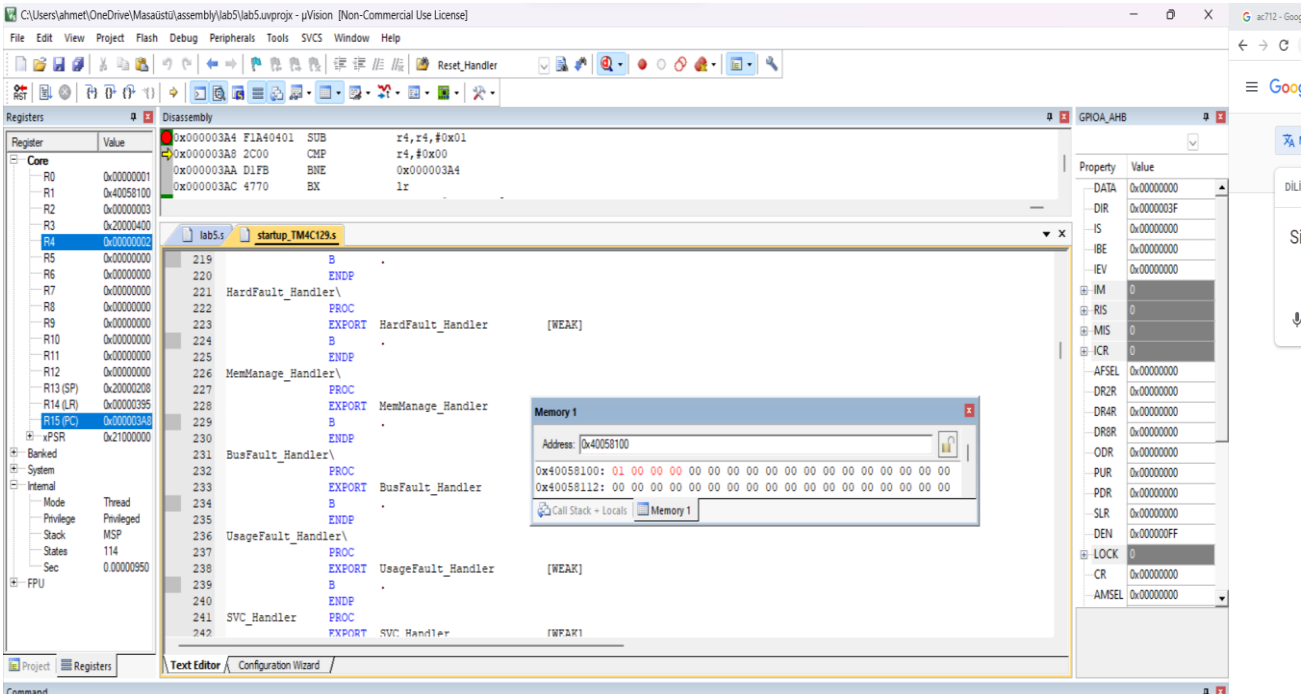

Figure 13

In Figure 14, register R4 became 1.

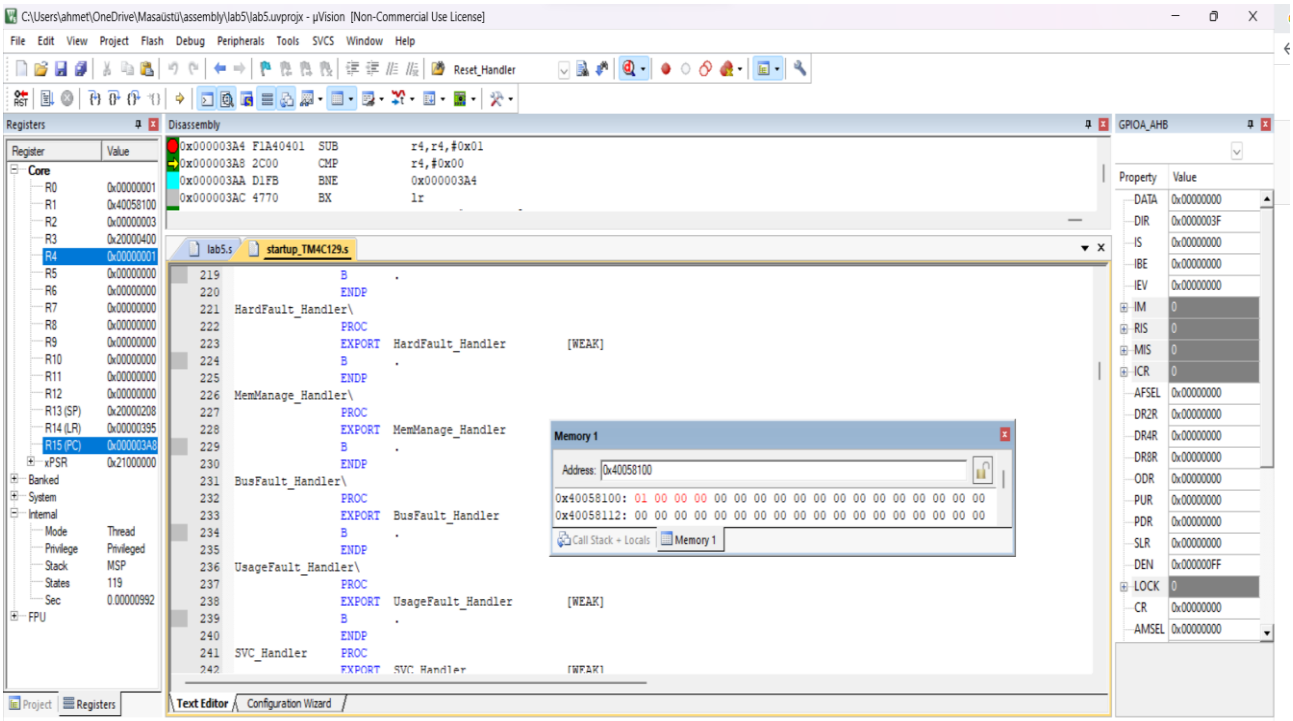

Figure 14

In Figure 15, register R4 became 0.

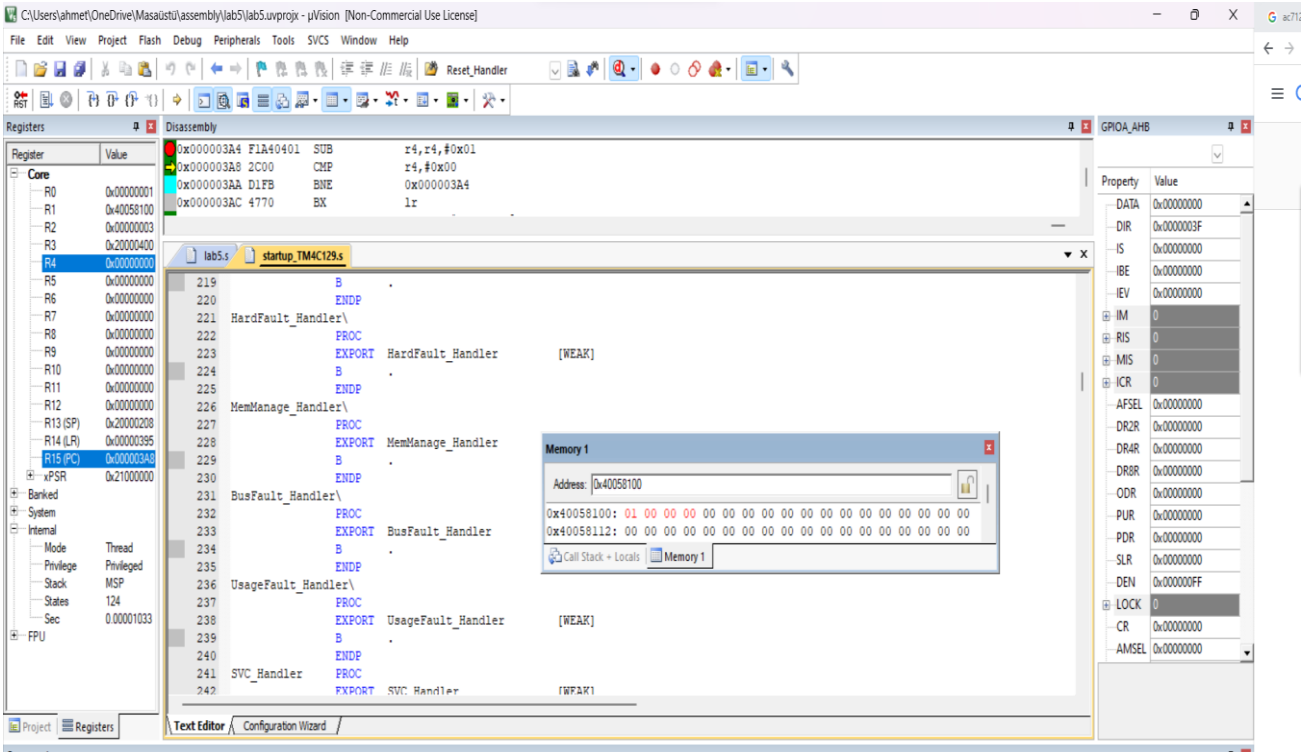

Figure 15

In Figure 16, the code arrived at the buzzer_ring branch and resumed. So it deactivated the buzzer.

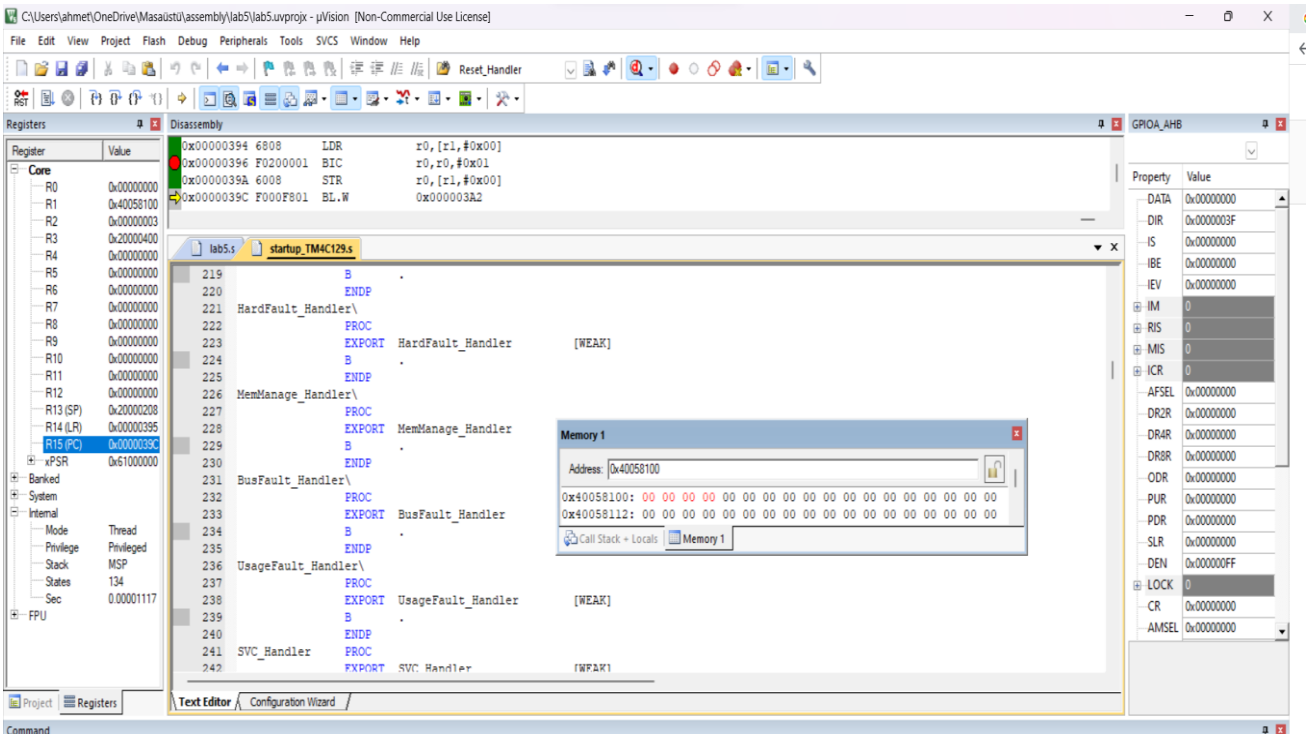

Figure 16

In figure 17, the code went to the delay branch and wrote 3 to register R4 and then went to the wait branch. It waited in the wait branch until the value of 3 became 0.

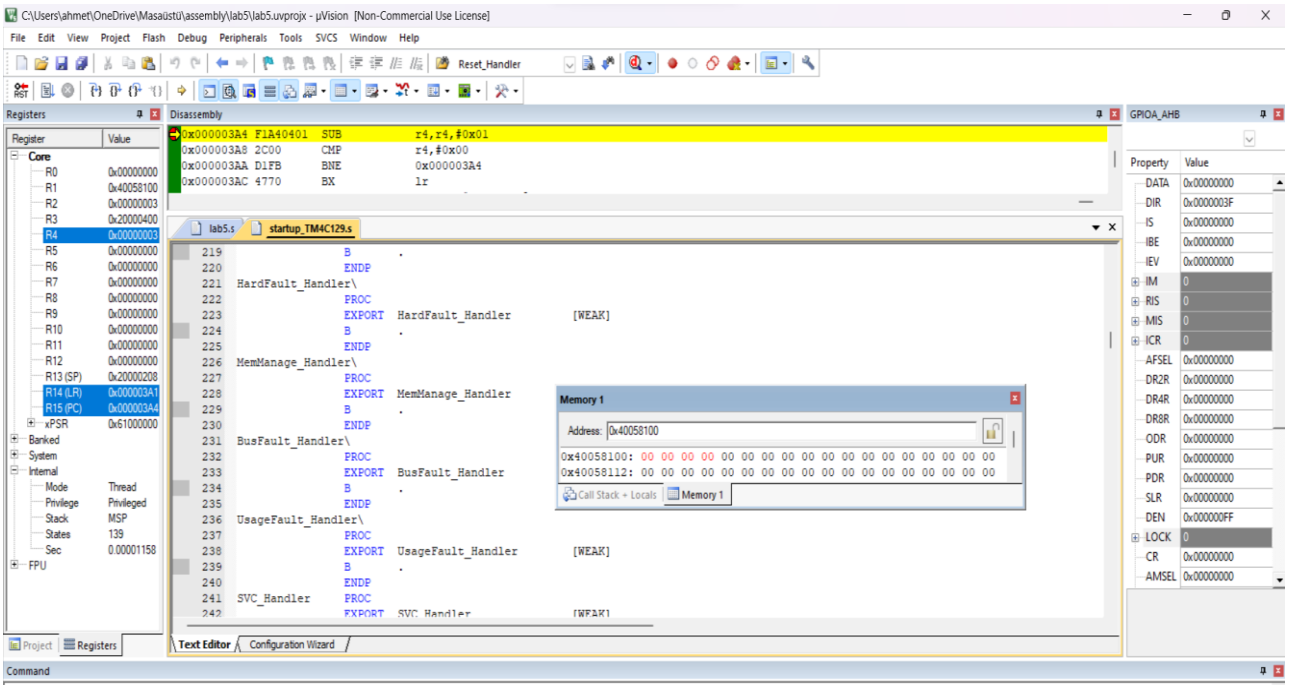

Figure 17

In Figure 18, register R4 became 2.

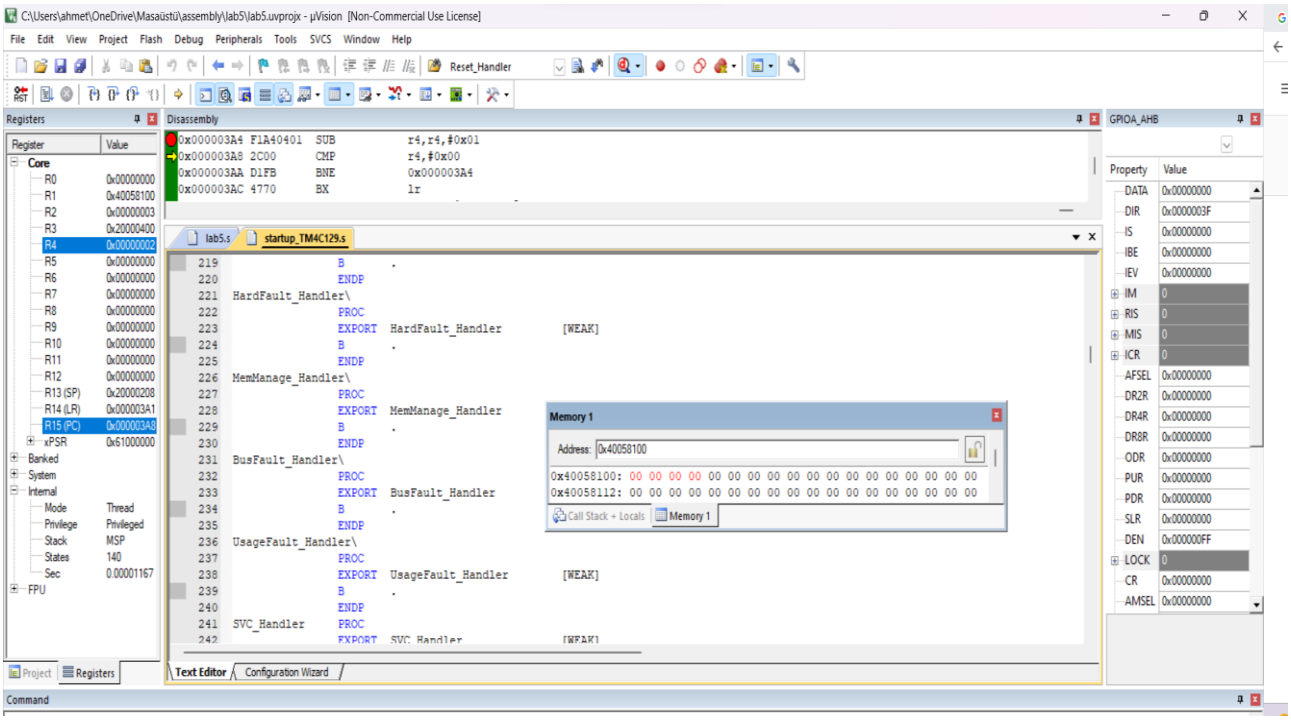

Figure 18

In Figure 19, register R4 became 1.

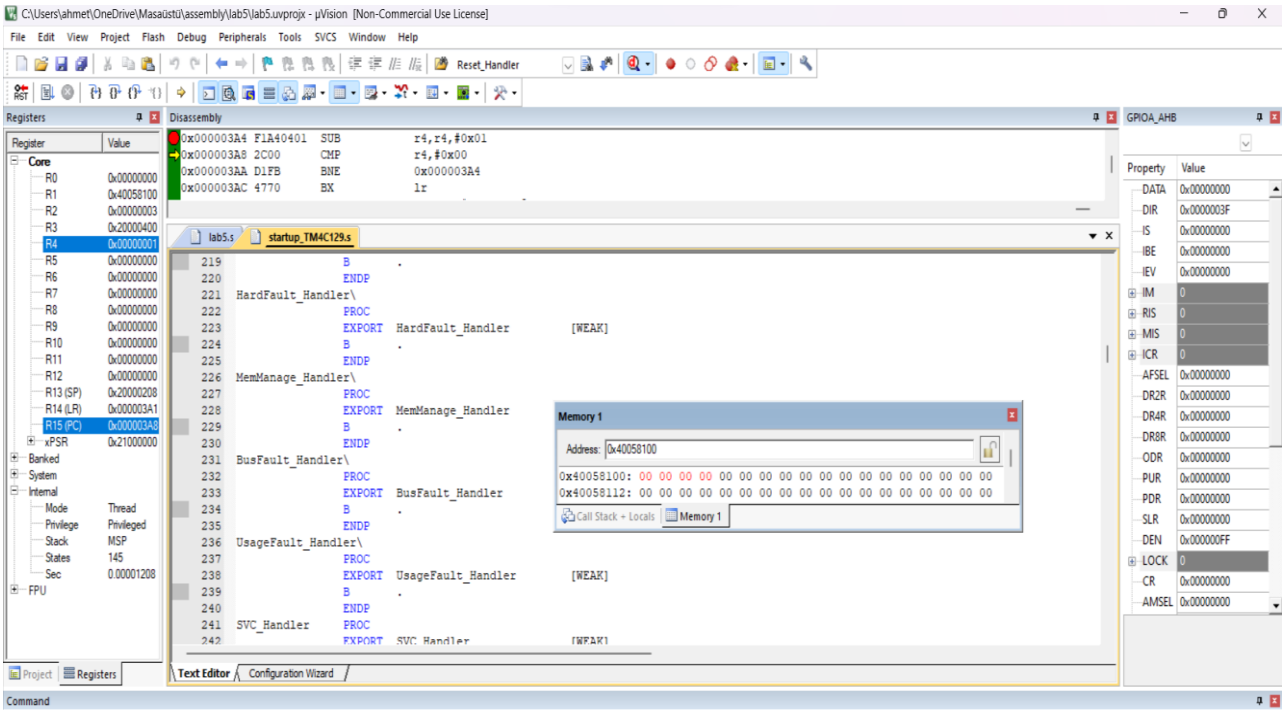

Figure 19

In Figure 20, register R4 became 0.

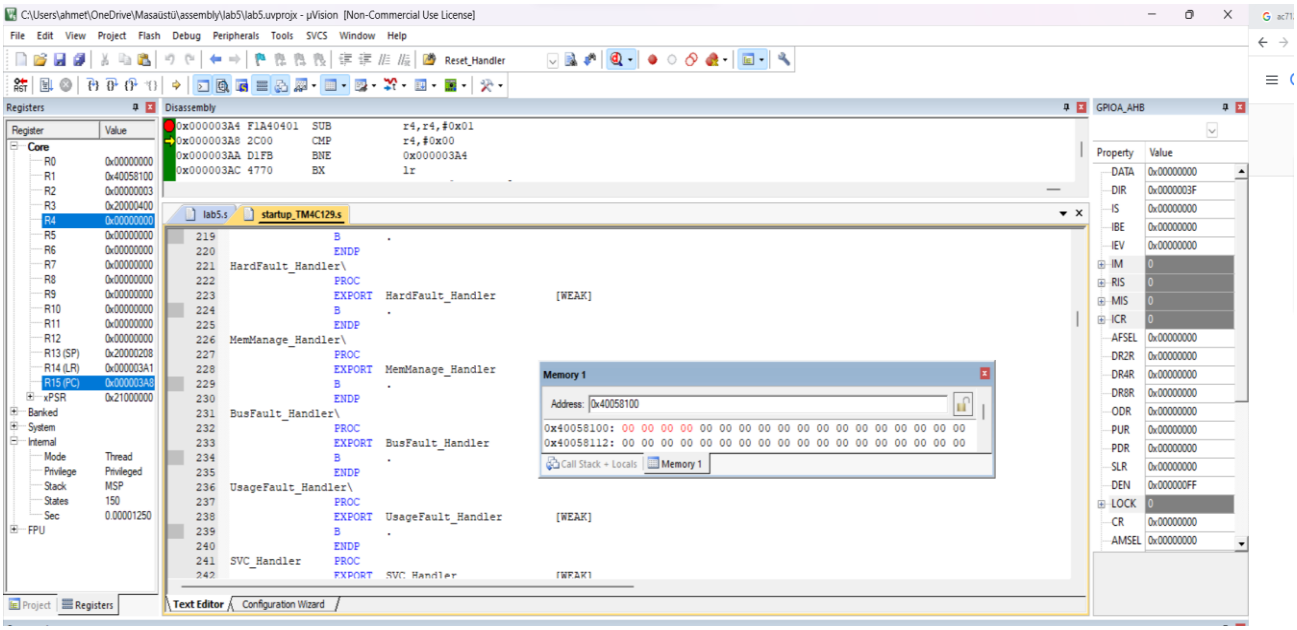

Figure 20

After this step, he went to the chechswitch branch to check the button on the D port again. The button has performed similar actions depending on the situation.