INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION

EC49001

MICROCONTROLLER SYSTEMS LABORATORY

ASSIGNMENT 3

SHIHKAR MOHAN

18EC10054

E & ECE

(B. Tech)

2021 – 2022

Questions

1. Design a digital clock (mm:ss) using 7-seg module and a switch for changing modes. Mode switch changes the clock mode to stopwatch.
2. Stopwatch mode has two modes - start and stop.

Objectives

1. Apply the concept of interrupts in MCU via assembly code.
2. Design an implementation of a clock-cum-stopwatch.

Description

1. We store the BCD data in registers 70H to 79H and use registers 50H-53H for clock output and 60H-63H for stopwatch output. Our display functions use the BCD representations stored in registers 70H-79H to display the required numbers on the seven segment display, so all that is left is the increment modules and mode switch control flow.

2. We use the interrupt functionality to increment the clock whenever the timer overflows, which happens every one (perceivable) second. Although switch 1 is used for mode selection and switch 0 for start/stop, the update happens in either only the clock or the clock and stopwatch both so that the clock update keeps happening in the background.

Code

```
1.  ORG 0000H
2.  LJMP START
3.
4.  ORG 001BH
5.  LJMP ISR_OUT
6.  ORG 0030H
7.  START:
8.  mov 80H, #11000000B        ; Codes for digits stored from 80H
9.  mov 81H, #11111001B
10. mov 82H, #10100100B
11. mov 83H, #10110000B
12. mov 84H, #10011001B
13. mov 85H, #10010010B
```

```
14. mov 86H, #10000010B
15. mov 88H, #11111000B
16. mov 88H, #10000000B
17. mov 89H, #10010000B
18. MOV R0,#0
19. MOV R1,#0
20. MOV R2,#0
21.
22. MOV 50H,#80H
23. MOV 51H,#80H
24. MOV 52H,#80H
25. MOV 53H,#80H
26.
27. MOV 60H,#80H
28. MOV 61H,#80H
29. MOV 62H,#80H
30. MOV 63H,#80H
31.
32. MOV TMOD,#10H ;Timer 1, mode 1
33. MOV TL1,#018H ;TL1=18 low byte of -1000
34. MOV TH1,#0FFH ;TH1=FC high byte of -1000
35. MOV IE,#88H ;10001000 enable Timer 1 int
36. SETB TR1
37.
38. DISPLAY:
39. JNB P2.0, D_STOPW
40. JB P2.0, D_CLOCK
41.
42. D_CLOCK:      ; Display Clock
43. CLR P3.3
44. CLR P3.4
45. MOV R0,50H
46. MOV P1,@R0
47. ACALL DELAY
48.
49. SETB P3.3
50. MOV R0,51H
51. MOV P1,@R0
52. ACALL DELAY
53.
54. SETB P3.4
55. CLR P3.3
56. MOV R0,52H
57. MOV P1,@R0
58. ACALL DELAY
59.
60. SETB P3.3
61. MOV R0,53H
62. MOV P1,@R0
63. ACALL DELAY
64.
65. JMP DISPLAY
66.
67. D_STOPW:      ; Display Stopwatch
68. CLR P3.3
69. CLR P3.4
70. MOV R0,60H
71. MOV P1,@R0
```

```
72. ACALL DELAY
73.
74. SETB P3.3
75. MOV R0,61H
76. MOV P1,@R0
77. ACALL DELAY
78.
79. SETB P3.4
80. CLR P3.3
81. MOV R0,63H
82. MOV P1,@R0
83. ACALL DELAY
84.
85. SETB P3.3
86. MOV R0,63H
87. MOV P1,@R0
88. ACALL DELAY
89.
90. JMP DISPLAY
91.
92.
93. DELAY:
94. MOV R3,#10
95. HERE: DJNZ R3, HERE
96. MOV P1,#255
97. RET
98.
99. ISR_OUT:
100.CLR TR1
101.MOV TL1,#018H ;TL1=18 low byte of -1000
102.MOV TH1,#0FFH ;TH1=FC high byte of -1000
103.SETB TR1
104.ACALL UPDATE_CLK
105.ACALL UPDATE_STP
106.RETI
107.
108.UPDATE_CLK:       ; Update the clock by incrementing
109.INC 50H
110.MOV A,50H
111.CJNE A,#8AH,NEXT
112.MOV 50H,#80H
113.
114.INC 51H
115.MOV A,51H
116.CJNE A,#86H,NEXT
117.MOV 51H,#80H
118.
119.INC 52H
120.MOV A,52H
121.CJNE A,#8AH,NEXT
122.MOV 52H,#80H
123.
124.INC 53H
125.MOV A,53H
126.CJNE A,#86H,NEXT
127.MOV 53H,#80H
128.
129.NEXT:RET
```

```
130.
131.UPDATE_STP:       ; Update stopwatch by incrementing, similar to UPDATE_CLK
132.JB P2.0,STOP
133.JB P2.1,STOP
134.INC 60H
135.MOV A,60H
136.CJNE A,#8AH,NEXT
137.MOV 60H,#80H
138.
139.INC 61H
140.MOV A,61H
141.CJNE A,#86H,NEXT
142.MOV 61H,#80H
143.
144.INC 62H
145.MOV A,62H
146.CJNE A,#8AH,NEXT
147.MOV 62H,#80H
148.
149.INC 63H
150.MOV A,63H
151.CJNE A,#86H,NEXT
152.MOV 63H,#80H
153.SJMP EXIT
154.STOP: ACALL RESET
155.EXIT: RET
156.
157.RESET:
158.MOV 60H,#80H
159.MOV 61H,#80H
160.MOV 62H,#80H
161.MOV 63H,#80H
162.RET                 ; Return to where the function was called
```

Screenshots

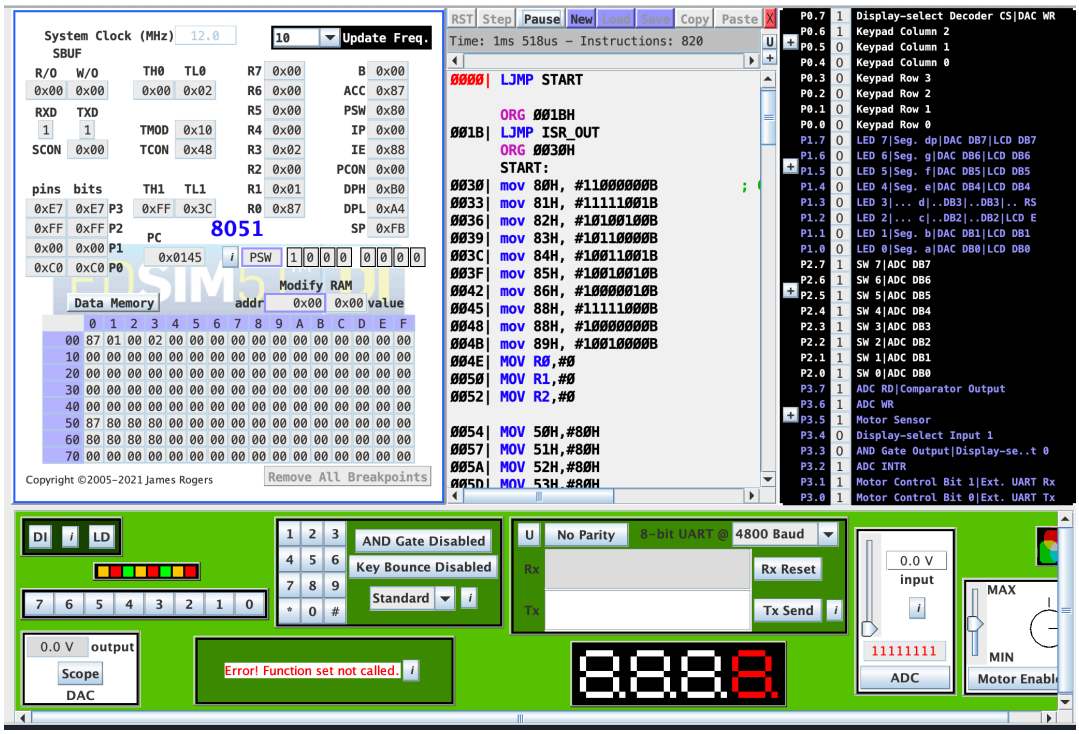

Fig. 1: Clock mode at t=8s

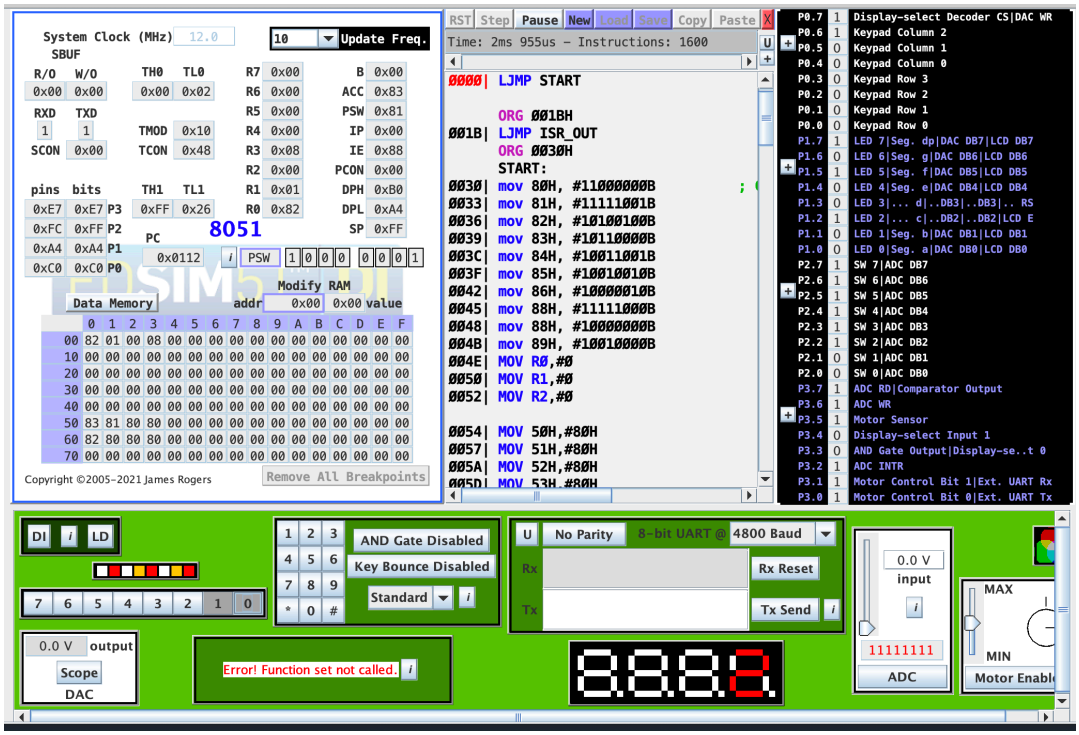

Fig. 2: Stopwatch mode at Start state, counts 2 seconds

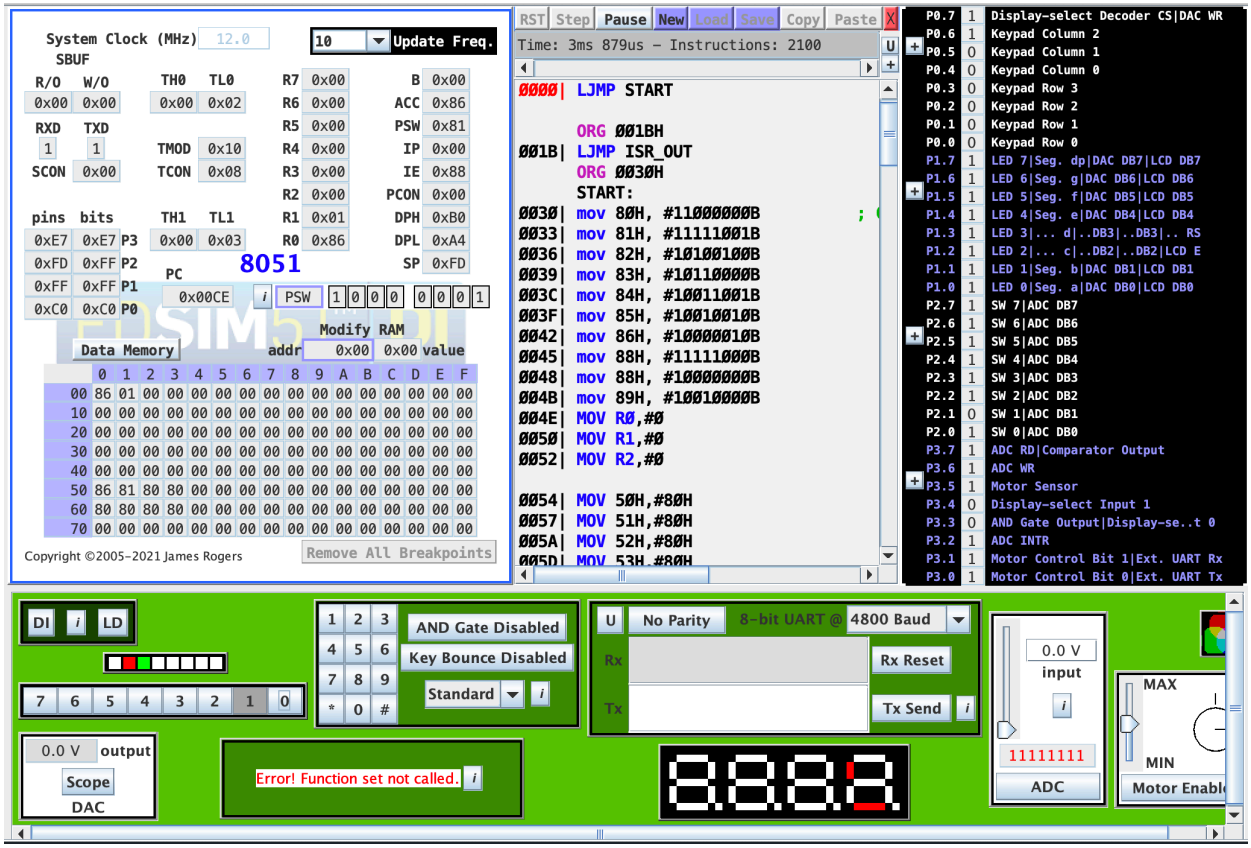

Fig. 3: Stopwatch mode when in stop state shows the clock output. This is in-between refresh states, but the time is t=12s.

Results

1. We observe that the LED clock implementation with stopwatch is working fine.
2. With the given implementation, the clock phase resumes once stopwatch mode is in stop state.

Discussion

1. When the stopwatch mode is in STOP state, it goes back to the clock state due to the effective AND nature of the two switches we use.
2. This display is difficult to capture on screen-record video due to refresh rate issues as well as the 25% PWM every 7-segment display gets due it being

connected by a decoder. This issue however is much less prominent in physical applications since then the PWM is not noticeable due to persistence of vision.

Summary

We utilise an efficient algorithm to update, display and switch between multiple routines and subroutines of a clock-cum-stopwatch module.