



SCHOOL OF COMPUTER SCIENCE ENGINEERING

WINTER SEMESTER 2022-2023

LAB ASSIGNMENT - 3

Slot: L11 – L12

Class: VL2022230504038

Programme Name & Branch: B. Tech CSE

Course code & Title: BECE204P – Microprocessors and Microcontrollers Lab

Faculty Name: Venu Allapakam

Implementation of Timers and counters

Program 1: Up Counting till 10

Aim: To design an ALP that counts the no. of people entering a room and when the count equals 10, it turns on the LED at P0.1

Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV TMOD, #06H;
3  HERE: CLR P0.1
4  MOV TH0, #00H
5  AGAIN: CLR P3.4;
6  SETB TR0
7  SETB P3.4;
8  MOV A, TL0
9  MOV P1, A
10 CJNE A, #0AH, AGAIN
11 SETB P0.1;
12 SJMP HERE
13 END

```

Output:

Before Execution Register status:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC \$	C:0x0000
auxr1	0x00
dp1r	0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:

Registers

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
PC	0x0005

Disassembly

```

2: MOV TMOD,#06H;
C:0x0000 758906 MOV TMOD(0x89),#0x06
3: HERE: CLR P0.1
C:0x0003 C281 CLR 0x80.1
4: MOV TH0,#00H
C:0x0005 758C00 MOV TH0(0x8C),#0x00
5: AGAIN: CLR P3.4;
C:0x0008 C2B4 CLR T0(0xB0.4)
6: SETB TR0
C:0x000A D28C SETB TR0(0x88.4)
7: SETB P3.4;
C:0x000C D2B4 SETB T0(0xB0.4)

```

LED ON OFF.asm

```

1 ORG 0000H
2 MOV TMOD,#06H;
3 HERE: CLR P0.1
4 MOV TH0,#00H
5 AGAIN: CLR P3.4;
6 SETB TR0
7 SETB P3.4;
8 MOV A,TLO
9 MOV P1,A
10 CJNE A,#0AH,AGAIN
11 SETB P0.1;
12 SJMP HERE
13 END

```

Parallel Port 0

Port 0: 0xFD 7 Bits 0

Pins: 0xFD

Registers

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
PC	0x0017
states	83

Disassembly

```

11: SETB P0.1;
C:0x0015 D281 SETB 0x80.1
12: SJMP HERE
C:0x0017 80EA SJMP HERE (C:0003)
C:0x0019 00 NOP
C:0x001A 00 NOP
C:0x001B 00 NOP
C:0x001C 00 NOP
C:0x001D 00 NOP
C:0x001E 00 NOP
C:0x001F 00 NOP
C:0x0020 00 NOP

```

LED ON OFF.asm

```

1 ORG 0000H
2 MOV TMOD,#06H;
3 HERE: CLR P0.1
4 MOV TH0,#00H
5 AGAIN: CLR P3.4;
6 SETB TR0
7 SETB P3.4;
8 MOV A,TLO
9 MOV P1,A
10 CJNE A,#0AH,AGAIN
11 SETB P0.1;
12 SJMP HERE
13 END

```

Parallel Port 0

Port 0: 0xFF 7 Bits 0

Pins: 0xFD

Result-

Hence, the no. of people entering the room are counted using counter 0 operated in mode 2 by giving the input via TR0 bit. Once the count completes to 10, the LED linked to P0.1 is turned on.

Program 2: Count and display

Aim: To design an ALP that counts the pulses provided via pin T1 using counter 1 and display the state of the TL1 count on P2, which connects to 8 LEDs.

Software Requirement: Keil Software

Program:

```

1 ORG 0000H
2 MOV TMOD, #01100000B
3 MOV TH1, #00H
4 SETB P3.5
5 AGAIN: SETB TR1
6 BACK: MOV A, TL1
7 MOV P2, A
8 JNB TF1, BACK
9 CLR TR1
10 CLR TF1
11 SJMP AGAIN
12 END

```

Output:

Before Execution Register status:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC \$	C:0x0000
auxr1	0x00
dp1r	0x0000
states	0
sec	0.00000000
psw	0x00

After Execution:

The screenshot displays the Keil IDE interface during the execution of an 8051 assembly program. The **Registers** window on the left shows the state of various registers, with register **a** containing the value **0x4e**. The **Disassembly** window in the center shows the following instructions:

```

8: JNB TF1, BACK
C:0x000E 308FF9 JNB TF1 (0x88.7), BACK (C:000A)
9: CLR TR1
C:0x0011 C28E CLR TR1 (0x88.6)
10: CLR TF1
C:0x0013 C28F CLR TF1 (0x88.7)
11: SJMP AGAIN
C:0x0015 80F1 SJMP AGAIN (C:0008)
C:0x0017 00 NOP
C:0x0018 00 NOP
C:0x0019 00 NOP
C:0x001A 00 NOP

```

Below the disassembly, the **COUNTER 2.asm** source file is visible, showing the following code:

```

1 ORG 0000H
2 MOV TMOD, #01100000B
3 MOV TH1, #00H
4 SETB P3.5
5 AGAIN: SETB TR1
6 BACK: MOV A, TL1
7 MOV P2, A
8 JNB TF1, BACK
9 CLR TR1
10 CLR TF1
11 SJMP AGAIN
12 END

```

Two pop-up windows, **Parallel Port 2** and **Parallel Port 3**, are displayed on the right. Both windows show the port value as **0x4E** and the pins as **0x4E**, indicating the current state of the ports during execution.

Result—

Hence, the pulses provided via T1 are counted using counter 1 and the state of TL1 is displayed on P2 which is further connected to 8 LEDs and therefore the LEDs glow as per the input.

Program 3:

Aim: To generate a square wave on P1 of any frequency without using timers

Software Requirement: Keil Software

Program:

```

1  L2: MOV A, #00H
2  MOV P1,A
3  CALL L1
4  MOV A,#0FFH
5  MOV P1,A
6  CALL L1
7  SJMP L2
8  L1: MOV R0,#100H
9  L3: NOP
10 NOP
11 DJNZ R0,L3
12 RET
13 END

```

Output:

BEFORE & AFTER Execution Register status:

The screenshot displays the Keil IDE interface. On the left, the 'Registers' window shows the status of various registers. On the right, the 'Logic Analyzer' window shows the execution trace for the 'SQUARE WAVE.asm' program.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

The Logic Analyzer window shows the execution trace for the 'SQUARE WAVE.asm' program. The trace is displayed in a table with columns for Setup, Load, Save, Min Time, Max Time, Grid, Zoom, Min/Max, Update Screen, Transition, Jump to, Signal Info, and Amplitude. The trace shows the program execution starting at 0s and ending at 10ms. The signal P1 is shown as a square wave, alternating between 0 and 1. The trace is displayed in a table with columns for Setup, Load, Save, Min Time, Max Time, Grid, Zoom, Min/Max, Update Screen, Transition, Jump to, Signal Info, and Amplitude.

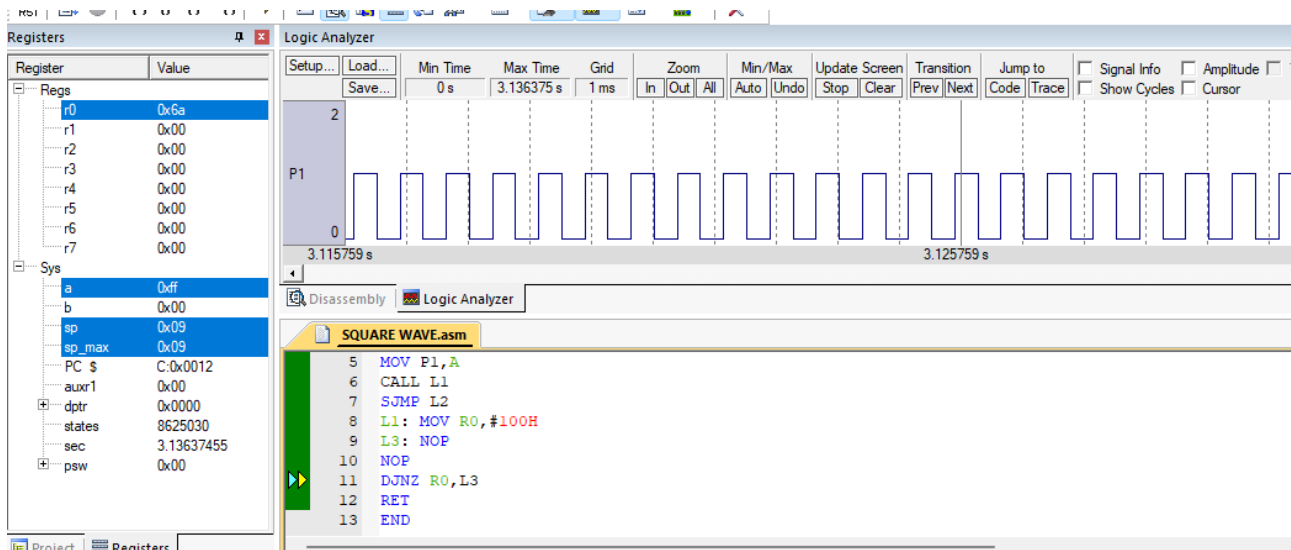
Setup...	Load...	Save...	Min Time	Max Time	Grid	Zoom	Min/Max	Update Screen	Transition	Jump to	Signal Info	Amplitude
			0 s	3.636364 us	1 ms	In Out All	Auto Undo	Stop Clear	Prev Next	Code Trace	<input type="checkbox"/> Show Cycles	<input type="checkbox"/> Cursor

The Logic Analyzer window also shows the disassembly of the program, which matches the code provided in the 'Program' section.

```

5  MOV P1,A
6  CALL L1
7  SJMP L2
8  L1: MOV R0,#100H
9  L3: NOP
10 NOP
11 DJNZ R0,L3
12 RET
13 END

```



Result:

Hence, the square wave with 50% duty cycle and a randomly selected time (delay) has been generated on P1 and is visualized using the logic analyzer.

Program 4:

Aim: To generate a 500 Hz square wave with 50% duty cycle using timer 0 on P1

Software Requirement: Keil Software

Program:

```

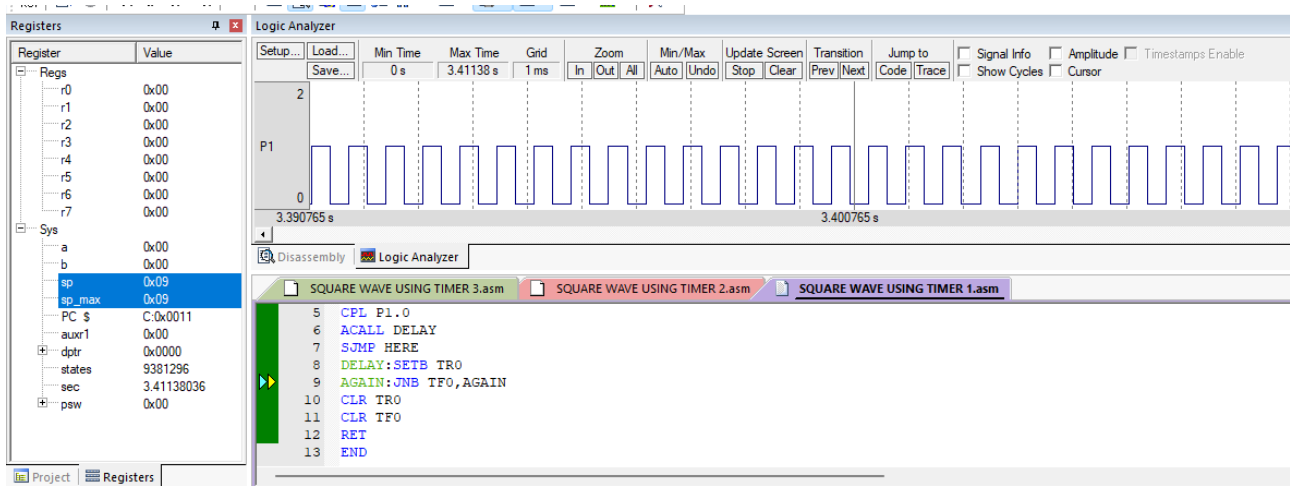
1  ORG 0000H
2  MOV TMOD,#01H
3  HERE:MOV TL0,#66H
4  MOV TH0,#0FCH
5  CPL P1.0
6  ACALL DELAY
7  SJMP HERE
8  DELAY:SETB TR0
9  AGAIN:JNB TF0,AGAIN
10 CLR TR0
11 CLR TF0
12 RET
13 END

```

Output:

BEFORE & AFTER Execution Register status:

The screenshot displays the Keil IDE interface. On the left, the 'Registers' window shows the status of various registers (r0-r7, Sys, a, b, sp, sp_max, PC, auxr1, dptr, states) with their values. The 'Logic Analyzer' window is open, showing a timing diagram for pin P1. The diagram displays a square wave signal, indicating the output of the program. The Logic Analyzer window also includes a disassembly view of the program code, showing the assembly instructions and their addresses.

**Result:**

Hence, the 500 Hz square wave with 50% duty cycle has been generated on P1 using timer 0 and is visualized using the logic analyzer.

Program 5: timer

Aim: To generate a 1 kHz square wave on one of the pins of P1 using timer 1.

Software Requirement: Keil Software

Program:

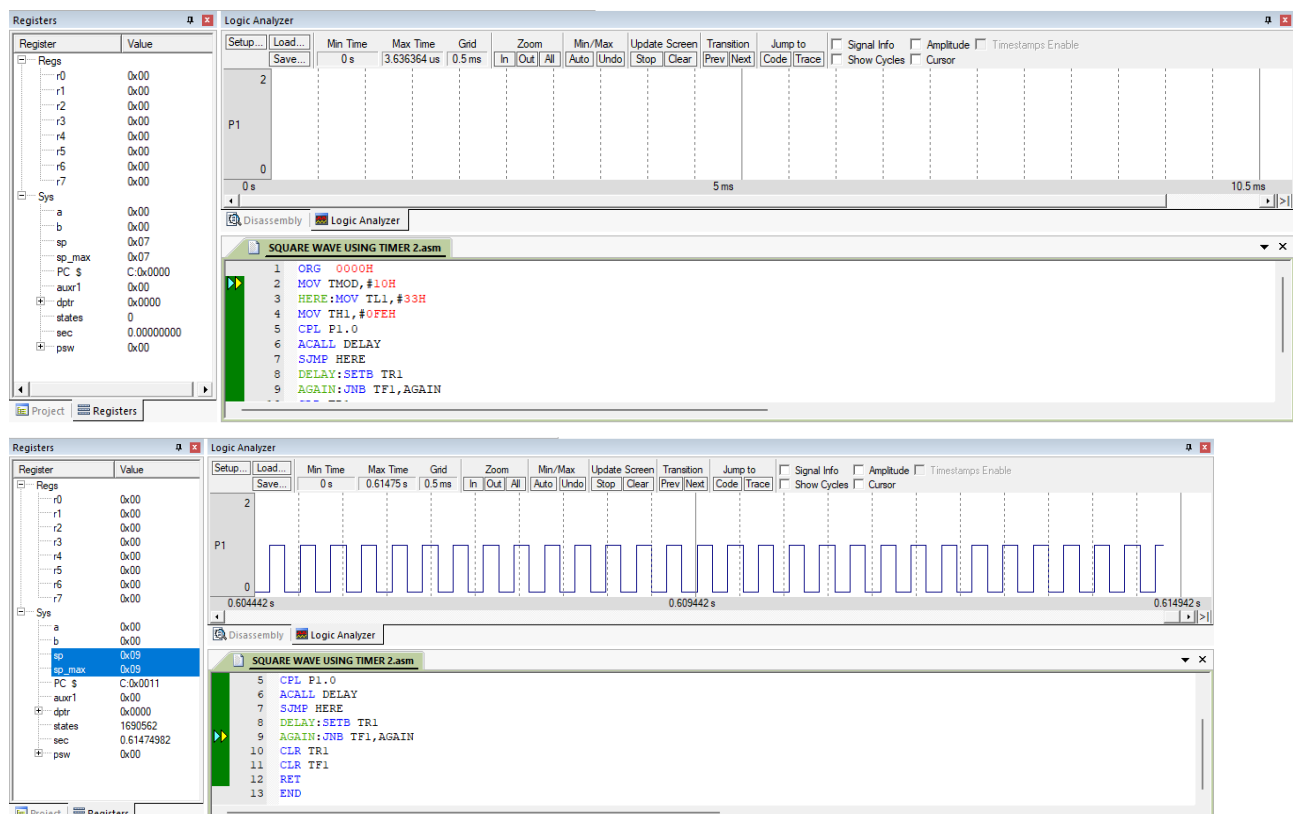
```

1  ORG 0000H
2  MOV TMOD, #10H
3  HERE: MOV TL1, #33H
4  MOV TH1, #0FEH
5  CPL P1.0
6  ACALL DELAY
7  SJMP HERE
8  DELAY: SETB TR1
9  AGAIN: JNB TF1, AGAIN
10 CLR TR1
11 CLR TF1
12 RET
13 END

```

Output:

Before Execution & After execution:



Result:

Hence, the 1 kHz square wave with 50% duty cycle has been generated on P1 using timer 1 and is visualized using the logic analyzer.

Program 6:

Aim: To generate a 2 kHz square wave on the pin of P1.0 using timer 1.

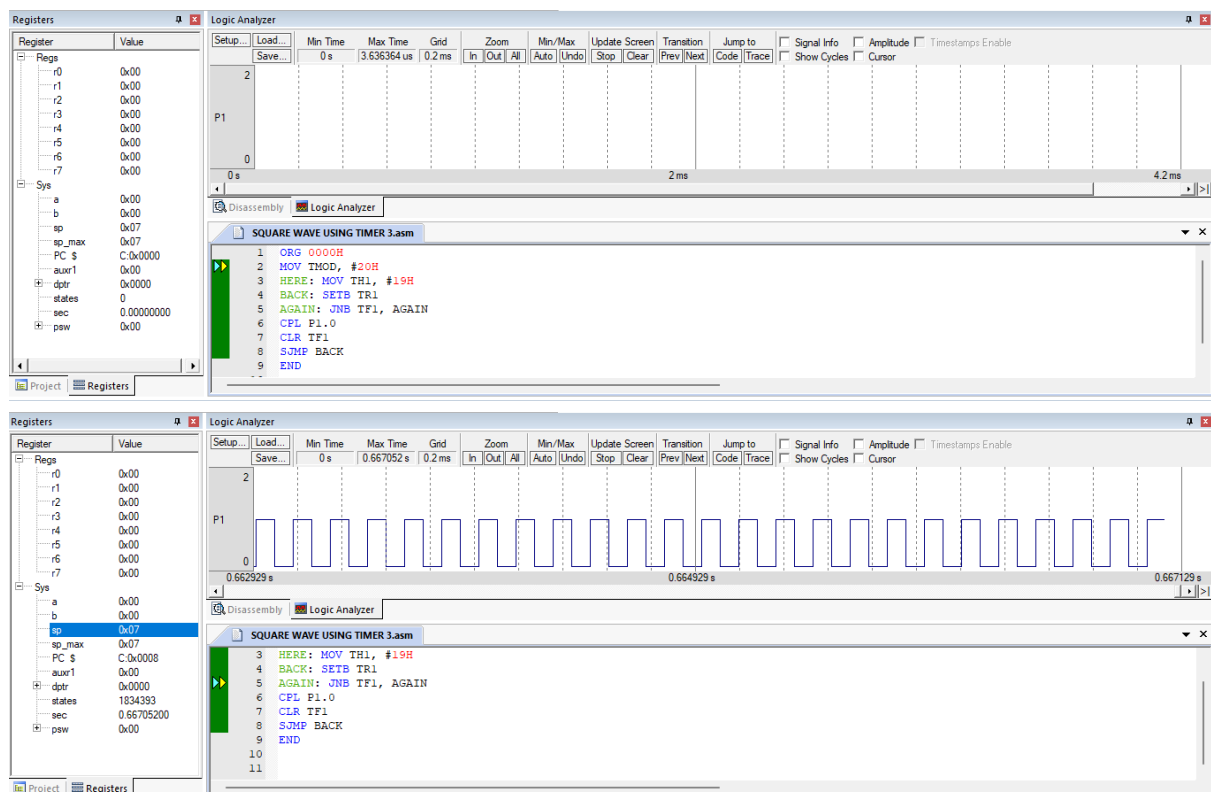
Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV TMOD, #20H
3  HERE: MOV TH1, #19H
4  BACK: SETB TR1
5  AGAIN: JNB TF1, AGAIN
6  CPL P1.0
7  CLR TF1
8  SJMP BACK
9  END

```

Output:**Before and After Execution:****Result:**

Hence, the 2 kHz square wave with 50% duty cycle has been generated on P.0 using timer 1.