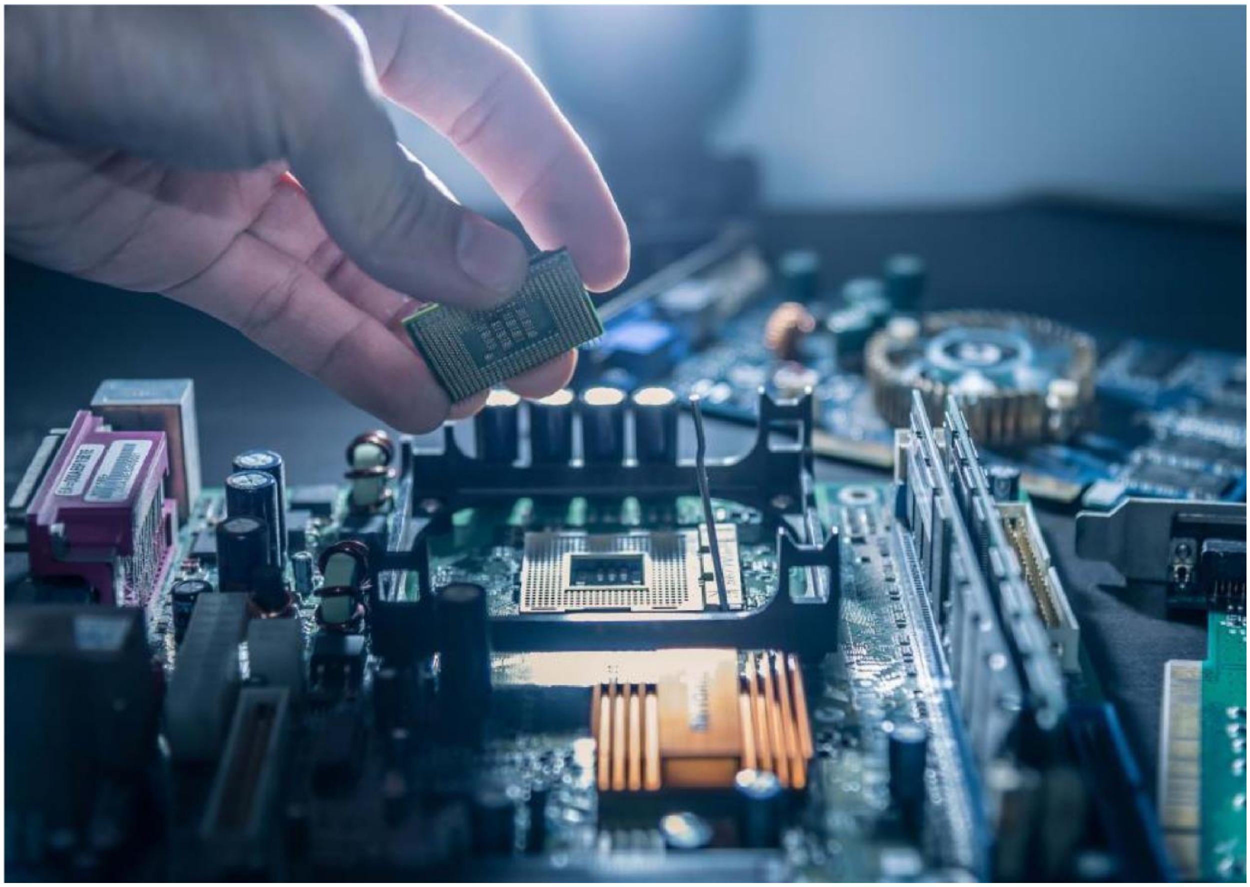
EMBEDDEDSYSTEMSLAB

Experiment IV : INTERFACING AN ADC



SAMINENI RAHUL: 21EC30045

CHANAKYA DUPPATLA:21EC30017

Group: 34

Embedded Systems Laboratory(EC39202)

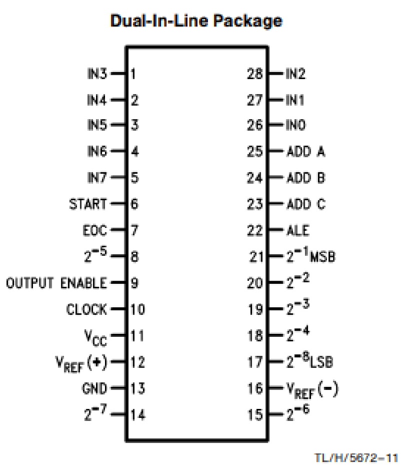
03-04-2023

Part-A

OBJECTIVE

To construct an ADC circuit to run in stand alone mode and verify its operation by applying its input using a potential divider in which the outputs of the ADC are observed by driving eight LEDs

BLOCK DIAGRAM:



Order Number ADC0808CCN, ADC0809CCN, ADC0808CCJ or ADC0808CJ See NS Package J28A or N28A

CIRCUIT DIAGRAM:

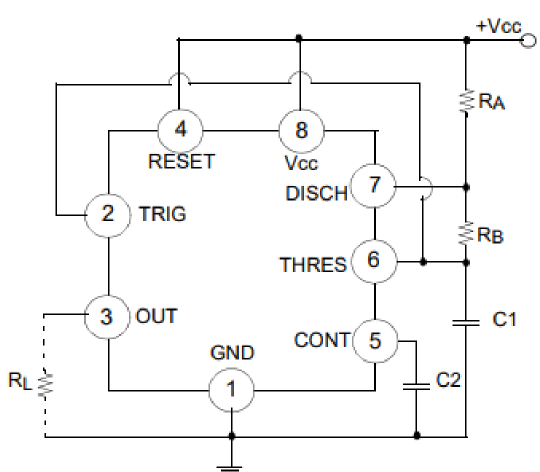
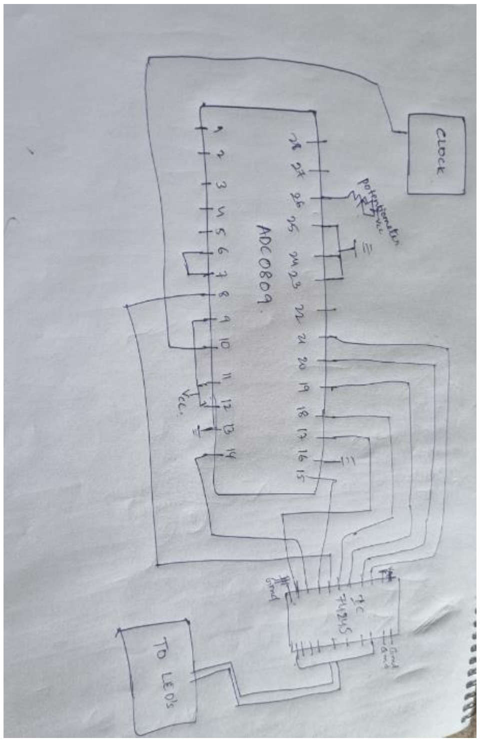


Figure 5. Astable Circuit

CLOCK CIRCUIT

Where Ra=1Kohm,Rb=1kohm,C1=1Microfarad



Timing Diagram

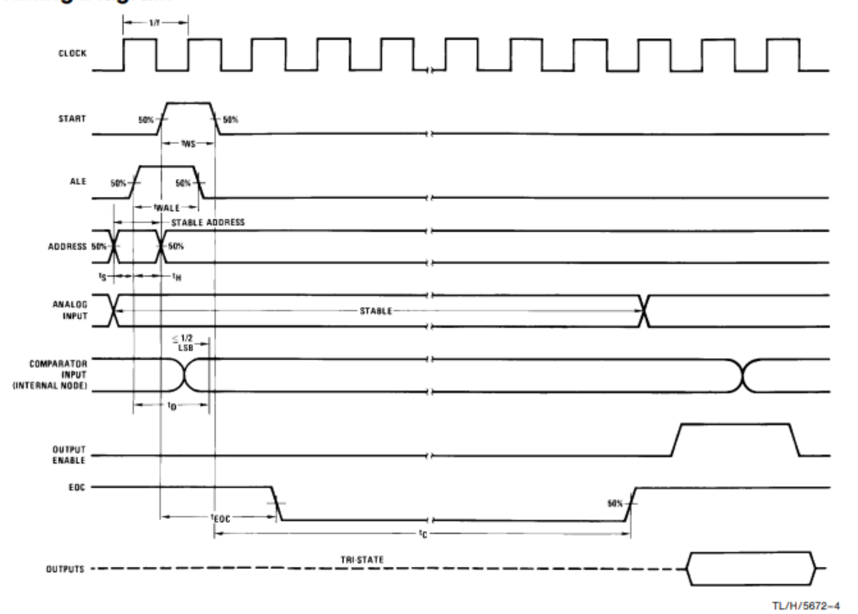
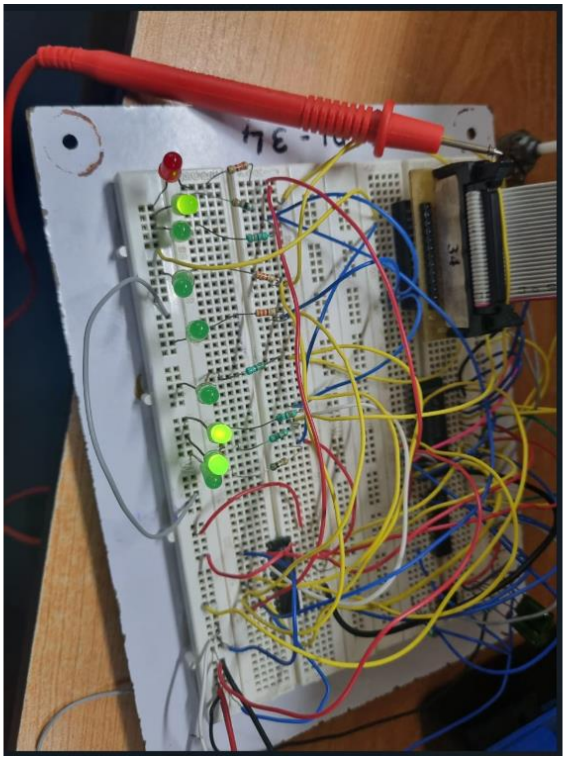
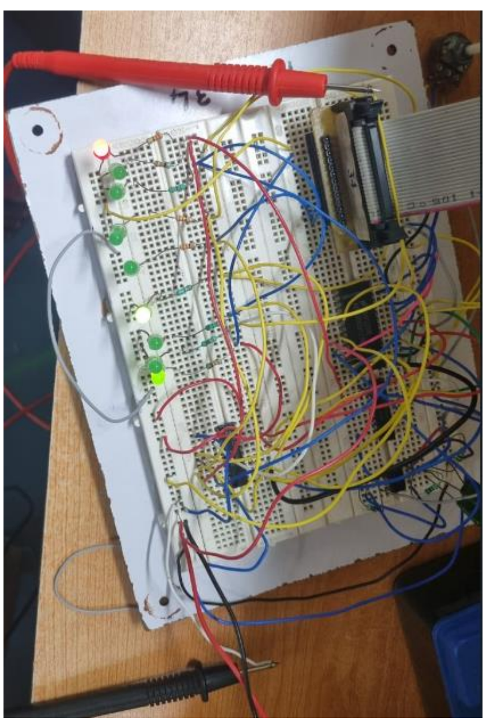


FIGURE 5

RESULTS:





Part-B

OBJECTIVE

Construct the circuit to access O/P of ADC through 8255 port (J3 / J7) and save the ADC O/P in a memory location 9070H. Compare this with a fixed value located at 9090H and generate a signal going through 8255 port (J3/J7) which is “HIGH” when the acquired data is greater than the fixed value and remains “LOW” otherwise.

PSEUDO CODE

```
Start:
  // Initialize memory locations with specific values
  memory[0xE803] = 0b10001001
  memory[0x9090] = 0b10000001

  // Enter the main loop
  Loop:
    // Read from memory location 0xE802 into accumulator
    accumulator = memory[0xE802]

    // Move the value from accumulator to memory location 0x9070
    memory[0x9070] = accumulator

    // Compare the value in accumulator with 0b10000001
    if accumulator != 0b10000001:
      goto Loop // Continue loop

    // If accumulator equals 0b10000001, turn on some functionality
    memory[0xE801] = 0x01 // Set memory location 0xE801 to 0x01

  // End of main loop, back to Loop

  // If carry flag is set, turn off some functionality
  memory[0xE801] = 0x00 // Set memory location 0xE801 to 0x00

  // Return to the beginning of the main loop
  goto Loop

End.
```

CODE:

```
ORG 8100H
MOV DPTR,#0E803H
MOV A,#10001001B
MOVX @DPTR,A
MOV DPTR,#9090H
MOV A,#10000001B
MOVX @DPTR,A
INTAKE:
MOV DPTR,#0E802H
MOVX A,@DPTR
MOV DPTR,#9070H
MOVX @DPTR,A
CJNE A,#10000001B,COMP
LJMP INTAKE
COMP:
JC OFF
MOV DPTR,#0E801H
MOV A,01H
MOVX @DPTR,A
LJMP INTAKE
OFF:
MOV DPTR,#0E801H
MOV A,#00H
MOVX @DPTR,A
LJMP INTAKE
END
```

CODE EXPLANATION:

1. ORG 8100H: This directive sets the origin (memory address) for the code to 8100H.

2. MOV DPTR,#0E803H: Move the value 0E803H (a memory address) into the data pointer (DPTR).

3. MOV A,#10001001B: Move the binary value 10001001 into the accumulator (A).

4. MOVX @DPTR,A: Move the value in the accumulator to the external memory location pointed to by DPTR.

5. MOV DPTR,#9090H: Move the value 9090H into DPTR.

6. MOV A,#10000001B: Move the binary value 10000001 into the accumulator.

7. MOVX @DPTR,A: Move the value in the accumulator to the external memory location pointed to by DPTR.

8. INTAKE: Label indicating the start of a loop or a subroutine.

9. MOV DPTR,#0E802H: Move the memory address 0E802H into DPTR.

10. MOVX A,@DPTR: Move the value from the external memory location pointed to by DPTR into the accumulator.

11. MOV DPTR,#9070H: Move the value 9070H into DPTR.

12. MOVX @DPTR,A: Move the value in the accumulator to the external memory location pointed to by DPTR.

13. CJNE A,#10000001B,COMP: Compare the value in the accumulator with the binary value 10000001. If they are not equal, jump to the label COMP.

14. LJMP INTAKE: Unconditional long jump (loop) back to the label INTAKE.

15. COMP: Label indicating the start of a comparison subroutine.

16. JC OFF: Jump to the label OFF if the carry flag is set.

17. MOV DPTR,#0E801H: Move the memory address 0E801H into DPTR.

18. MOV A,01H: Move the hexadecimal value 01 into the accumulator.

19. MOVX @DPTR,A: Move the value in the accumulator to the external memory location pointed to by DPTR.

20. LJMP INTAKE: Unconditional long jump (loop) back to the label INTAKE.

21. OFF: Label indicating the start of a subroutine to turn off some functionality.

22. MOV DPTR,#0E801H: Move the memory address 0E801H into DPTR.

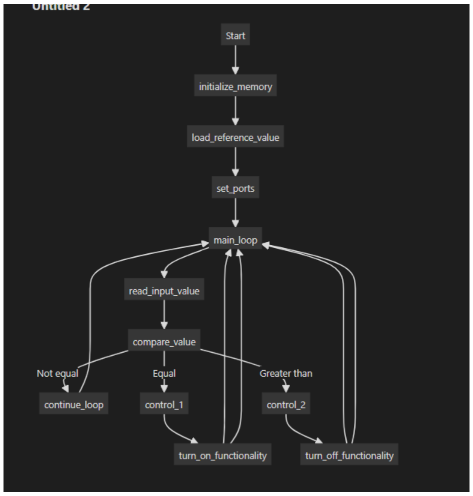
23. MOV A,#00H: Move the hexadecimal value 00 into the accumulator.

24. MOVX @DPTR,A: Move the value in the accumulator to the external memory location pointed to by DPTR.

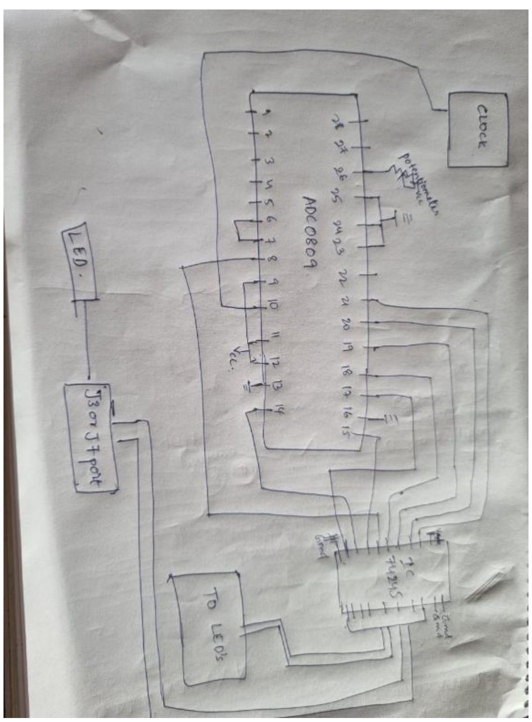
25. LJMP INTAKE: Unconditional long jump (loop) back to the label INTAKE.

26. END: End of the code.

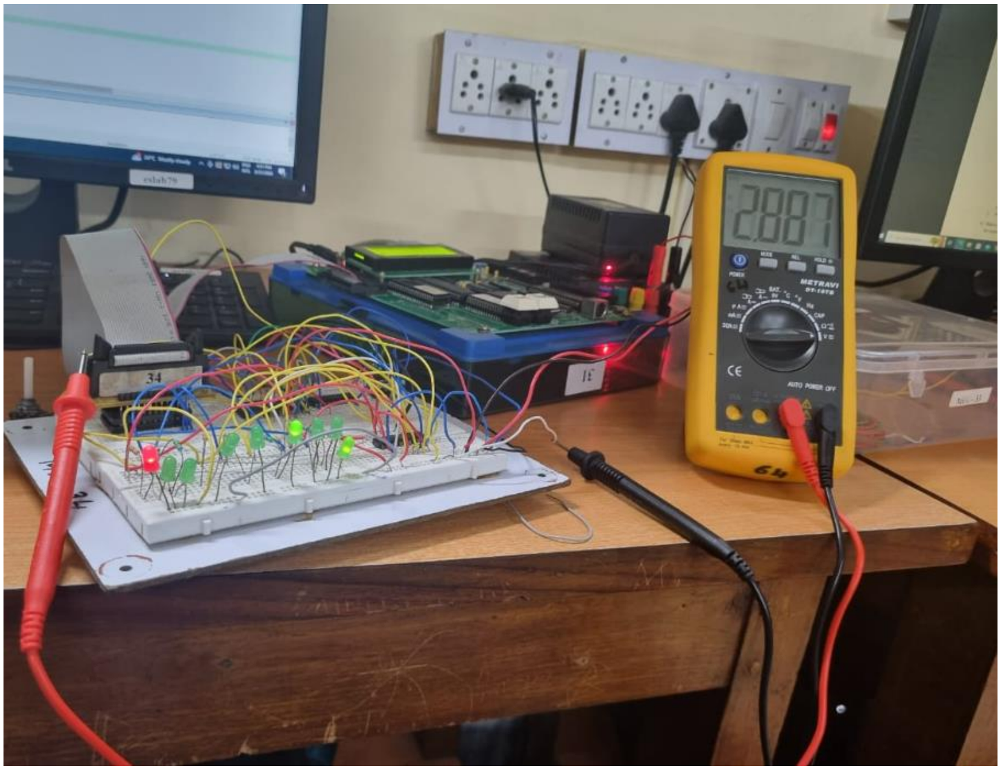
FLOW CHART:

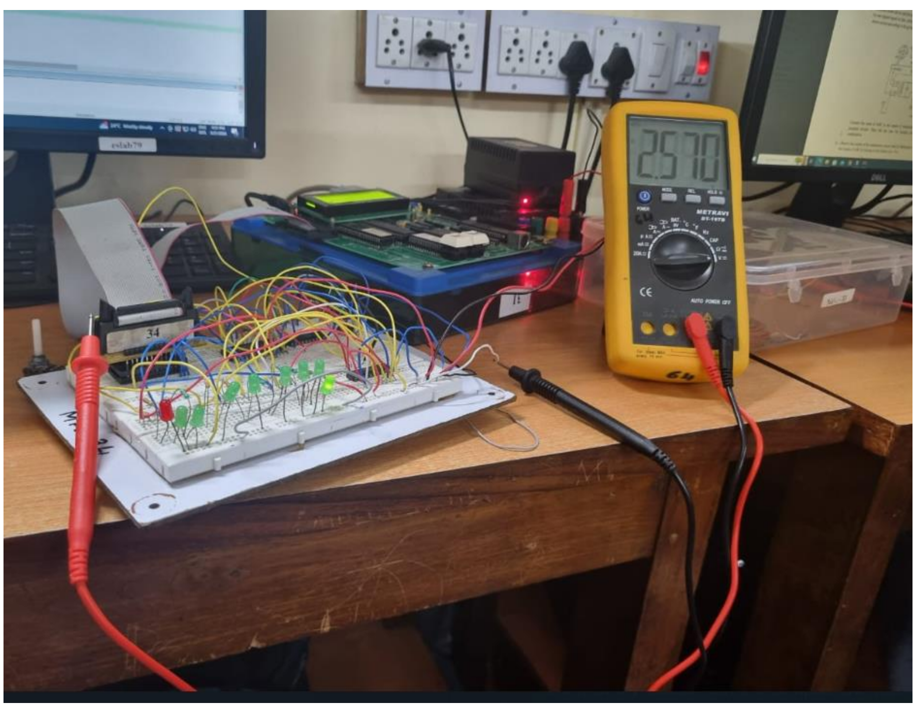


CIRCUIT DIAGRAM:



RESULTS:





DISCUSSION

The experiment involves using an Analog-to-Digital Converter (ADC) to convert analog signals from a variable voltage source into digital values.
A key aspect of the experiment is generating a 500KHz clock signal using a 555 Timer IC, which is supplied to the ADC's clock pin for analog-to-digital conversion. The analog input voltage to the ADC is provided through a potential divider connected to Pin 26 (IN0).

The Address Latch Enable (ALE) signal is set to 1 for 2 seconds to ensure stable address inputs for the ADC. The ADC's input is configured to IN0 by setting the control signals CBA to 000, routing the IN0 input to the ADC's comparator.

The experiment starts by storing a predetermined reference value (#9090H) in external memory, which serves as a fixed reference point for comparison with acquired data. The ports of the 8255 programmable peripheral interface are set up, with Port A as an input port and Port B as an output port.

The fixed reference value is transferred to register B, and the inputted value from the ADC is transferred to register A for comparison. The comparison determines whether the inputted data is less than, greater than, or equal to the reference value.

Based on the comparison result, the operation of the fan or heater is controlled accordingly. This control is achieved using an IC 74245 to connect LEDs in a manner that indicates whether the inputted data is greater than the reference value or not.

The experiment demonstrates the practical application of ADCs in temperature control systems for homes or industrial processes. By accurately converting analog signals to digital values and implementing control logic based on these values, precise temperature regulation can be achieved.