**Name: LƯƠNG THÀNH VỸ (2151280)**

**TRẦN TRUNG TOÀN (2151265)**

# OBJECTIVE:

* Communicate with a 7-segment LED display.
* Communicate with a matrix LED display.

# REFERENCES:

* Lab manual chapter 4
* Atmel-2505-Setup-and-Use-of-AVR-Timers\_ApplicationNote\_AVR130.pdf

# EXPERIMENT 1:

1. Connect one AVR port to header J34. Connect two other port pins to signals nLE0 and nLE1 on header J82. Set jumpers to power the 7-segment LED display.
2. Use sample programs from the experiment guide to write a program that displays the numbers 0, 1, 2, and 3 on the 4 7-segment LED displays. Use Timer 0 to scan the LEDs at a scanning frequency of 50 Hz.

# EXPERIMENT 2:

1. Connect an AVR port to a dip switch, assuming it's PORTA.
2. Write a program to display the value of PORTA \* 9 on the 4 7-segment LED displays.
3. Change the dip switch value and observe the results.

# EXPERIMENT 3:

1. Connect the necessary signals to control the matrix LED display.
2. Use the provided sample program, modify it if necessary, to display the letter 'A' on the matrix LED display. Scan the matrix LED display using a timer to create a delay with a scanning frequency of 25 Hz.
3. Modify the program to achieve a scanning frequency of 125 Hz.

# EXPERIMENT 1:

1. Answer the following questions:
   1. To have a scanning frequency of 50 Hz, how long does one LED stay lit?

**ANSWER:**

There are four 7-segment LEDs, and for the display to achieve a scanning frequency of 50 Hz, the entire display (all four 7-segment LEDs) must complete one full cycle of updates every T = 1/f = 0.02 s = 20 ms. This means that the AVR microcontroller should scan and update all four 7-segment LEDs within this 20 ms period.

Since there are four LEDs to be refreshed sequentially, each 7-segment LED will be updated for approximately 20 ms / 4 = 5 ms before the control signal switches to the next LED. In other words, each 7-segment LED will remain lit for 5 ms before the control signal switches to the next LED, turning off the current one and lighting up the next in sequence. This process repeats cyclically, ensuring that each LED is refreshed every 20 ms, resulting in a perceived scanning frequency of 50 Hz across the display.

* 1. How do you configure the timer to achieve this delay?

**ANSWER:**

With an 8 MHz clock (where each cycle takes 0.125 µs), achieving a 5 ms delay requires approximately 40,000 cycles between each LED update in a scanning cycle.

To increase precision, we can first write and simulate the program without any delays to measure the number of cycles required for the LED update routine alone. We then subtract this cycle count from the planned 40,000 cycles. Next, we create a delay subroutine that provides the remaining cycles needed to reach exactly 40,000.

In the delay subroutine, we calculate the exact number of cycles and derive a function with variables representing the number of iterations (counters). Using a Python or C program, we iterate through all possible combinations of these iterations to find the optimal one that achieves the closest match to the desired delay cycles.

1. Provide the source code with comments.

|  |
| --- |
| ; J34 CONNECT TO PORTD  ; NLE0 CONNECT TO PB4  ; NLE1 CONNECT TO PB5  ; OUTPUT: NONE  .ORG 0x00  JMP MAIN  TABLE\_7SEG\_DATA: ; Lookup table for 7-segment codes for decimals  .DB 0xC0, 0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x8  .DB 0xC6,0xA1,0x86,0x8E  TABLE\_7SEG\_CONTROL: ; Lookup table for four LEDs Common Anode pins  .DB 0b00000001, 0b00000010, 0b00000100, 0b00001000  .EQU LED7SEG\_PORT = PORTA  .EQU LED7SEG\_DDR = DDRA  .EQU LED7SEG\_LATCH\_PORT = PORTC  .EQU LED7SEG\_LATCH\_DDR = DDRC  .EQU Common\_Anode\_DDR = 0b00001111  .EQU nLE0\_PIN = 5  .EQU nLE1\_PIN = 4    MAIN: RCALL LED\_7SEG\_PORT\_INIT  CLR R26 ; R16 = Index for LED\_0 -> LED\_3Start  LDI R27, 1 ; R17 = Number needed to be display (0 to 9)  LOOP: CALL DISPLAY\_7SEG  CALL DELAY\_4ms  INC R26  INC R27 ; R17 goes from 1 to 4  ANDI R26, 0x0F  CPI R26, 4 ; Return to LED\_0, when finished displaying LED\_3  BRNE LOOP  LDI R27, 1 ; Initialization for the next scanning cycle  CLR R26  JMP LOOP  LED\_7SEG\_PORT\_INIT:  PUSH R20 ; Save R20 before using it  LDI R20, 0xFF ; Set LED7SEG port as output  OUT LED7SEG\_DDR, R20  ; Set as output for signal each common Anode of 7SEG LEDs  LDI R20, Common\_Anode\_DDR  ORI R20, (1 << nLE0\_PIN) | (1 << nLE1\_PIN)  OUT LED7SEG\_LATCH\_DDR, R20  POP R20  RET  ; INPUT: R27 contains the value to display  ; R26 contain the index for LED (from 0 -> 3, as for LED\_0 to LED\_3)  ; If using a common cathode display, invert the values in the table above  DISPLAY\_7SEG:  PUSH R16 ; Save R16 and R18 before using it  PUSH R18  CLR R16  ; Z = base address of the look-up table  LDI ZH, HIGH(TABLE\_7SEG\_DATA << 1)  LDI ZL, LOW(TABLE\_7SEG\_DATA << 1)  ADD ZL, R27 ; Add R27 to Z (16-bit)  ADC ZH, R16 ; Add carry to ZH if needed    LPM R18, Z ; Load the code to the 7SEG pins  OUT LED7SEG\_PORT, R18  SBI LED7SEG\_LATCH\_PORT, nLE0\_PIN ; Pulse the latch to update  NOP  CBI LED7SEG\_LATCH\_PORT, nLE0\_PIN    ; Z = base address of the look-up control table  LDI ZH, HIGH(TABLE\_7SEG\_CONTROL << 1)  LDI ZL, LOW(TABLE\_7SEG\_CONTROL << 1)  ADD ZL, R26 ; Add R27 to Z (16-bit)  ADC ZH, R16 ; Add carry to ZH if needed    LPM R18, Z ; Load the code to the 7SEG pins  OUT LED7SEG\_LATCH\_PORT, R18  SBI LED7SEG\_LATCH\_PORT, nLE1\_PIN ; Pulse the latch to update  NOP  CBI LED7SEG\_LATCH\_PORT, nLE1\_PIN  POP R18 ; Restore the temporary register  POP R16  RET      .EQU TCCR0B\_mode = 0b00000100  .EQU TCNT\_init = 100 ; 0 to 255  DELAY\_4ms:  LDI R16, TCNT\_init ; Set initial value for Timer (TNCT)  OUT TCNT0, R16  LDI R16, 0b00000000 ; Choose Normal mode  OUT TCCR0A, R16  LDI R16, TCCR0B\_mode ; Choose Prescale and start Timer  OUT TCCR0B, R16  AGAIN: SBIS TIFR0, TOV0 ; Check for Overflow  RJMP AGAIN    LDI R16, 0 ; Stop the TImer  OUT TCCR0B, R16  LDI R16, (1 << TOV0) ; Clear TImer overflow flag  OUT TIFR0, R16  RET |

**SIMULATION RESULT:**

**A computer screen shot of a circuit board

Description automatically generated**

# EXPERIMENT 2:

1. Answer the following questions:
   1. How many bits is the value of PORTA \* 9?

**ANSWER:**

The maximum value PORTA can hold is 255. When multiplied by 9, this gives a maximum result of 2295. In binary, 2295 is represented as 1000 1111 0111 (or 0x8F7). This shows that the result of PORTA \* 9 could require up to 12 bits for full representation.

* 1. How can you display each digit on the 4 7-segment LEDs?

**ANSWER:**

Since the result is a 12-bit number, using the Shift-Add-3 method for binary-to-decimal conversion is impractical due to its complexity and the need for multiple storage operations and loops. Instead, we can extract each decimal digit directly by successively dividing the result by 1000, 100, and 10 to extract the thousands, hundreds, and tens digits, while the final remainder gives the units digit.

To implement this, a division subroutine is required. This can be implemented by repeatedly subtracting the divisor until the result turns negative, with each subtraction counted to determine the quotient. The last positive remainder represents the remaining part of the division.

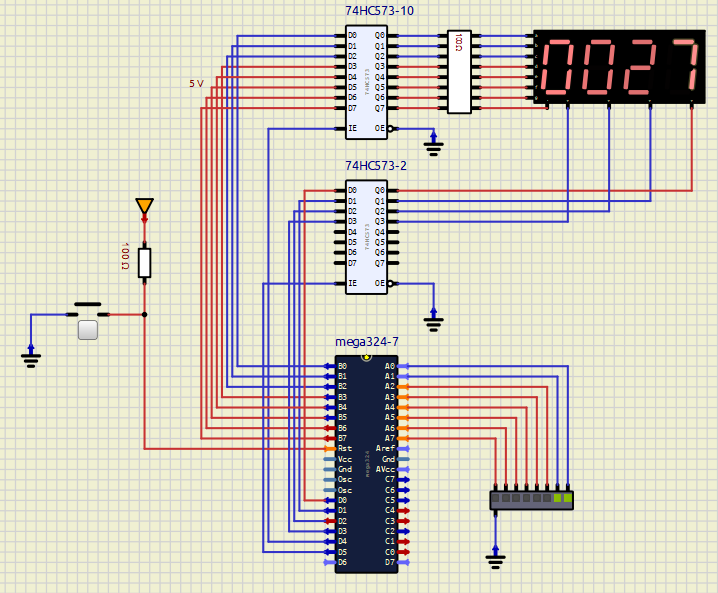
For numbers larger than 255, we need 16-bit registers (like X, Y, and Z) to handle the calculations. When performing 16-bit addition and subtraction on these registers, we use combinations of ADD-ADC and SUB-SBC instructions, passing the carry from the lower byte to the higher byte through the ADC or SBC operations​​.

1. Provide the source code with comments.

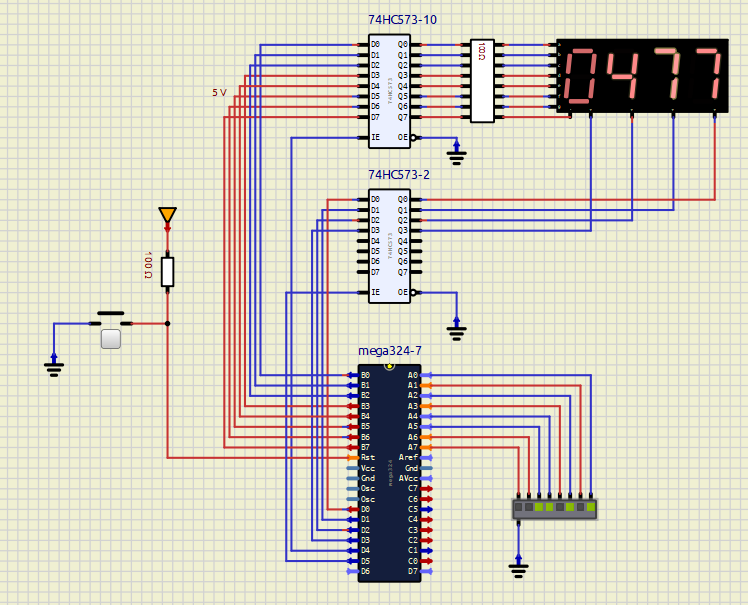
|  |
| --- |
| ; J34 CONNECT TO PORTD  ; NLE0 CONNECT TO PB4  ; NLE1 CONNECT TO PB5  ; OUTPUT: NONE  .ORG 0x00  JMP MAIN  TABLE\_7SEG\_DATA: ; Lookup table for 7-segment codes  .DB 0xC0, 0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x8  .DB 0xC6,0xA1,0x86,0x8E  TABLE\_7SEG\_CONTROL: ; Lookup table for led control  .DB 0b00000001, 0b00000010, 0b00000100, 0b00001000  .EQU LED7SEG\_PORT = PORTB  .EQU LED7SEG\_DDR = DDRB  .EQU LED7SEG\_LATCH\_PORT = PORTD  .EQU LED7SEG\_LATCH\_DDR = DDRD  .EQU DIP\_SWITCH\_PORT = PORTA  .EQU DIP\_SWITCH\_DDR = DDRA  .EQU DIP\_SWITCH\_PIN = PINA  .EQU Common\_Anode\_DDR = 0b00001111  .EQU nLE0\_PIN = 4  .EQU nLE1\_PIN = 5  MAIN: RCALL LED\_7SEG\_PORT\_INIT  LDI R24, 0x00 ; Set Input PORT for DIP switch  OUT DIP\_SWITCH\_DDR, R24  LDI R24, 0xFF ; Enable pull-up resistors for DIP swtich  OUT DIP\_SWITCH\_PORT, R24  LDI R20, 0x09 ; For multiplication by 9  READ: IN R21, DIP\_SWITCH\_PIN ; Input from 0 to 15  COM R21  MUL R21, R20 ; Result is store in R1 and R0, Result = [0, 2295]  LOOP: LDI R24, 3 ; The thousands-digit  MOV YH, R1 ; Perform Y/X = Result / 1000  MOV YL, R0  LDI XH, 0x03  LDI XL, 0xE8  CALL DIVISION ; R17:R16 = Quotient  ; R19:R18 = Remainder  OUT PORTC, R18  SBI PORTD, 6  NOP  NOP  CBI PORTD, 6  MOV R25, R16 ; Display quotient (R16 surely < 10)  CALL DISPLAY\_7SEG  CALL DELAY\_4ms  LDI R24, 2 ; Next: the hundreds-digit  MOV YH, R19 ; Perform Y/X = Remainder / 100  MOV YL, R18  LDI XH, 0x00  LDI XL, 0x64  CALL DIVISION ; R17:R16 = Quotient  ; R19:R18 = Remainder  MOV R25, R16 ; Display quotient (R16 surely < 10)  CALL DISPLAY\_7SEG  CALL DELAY\_4ms  LDI R24, 1 ; Next: the tenth-digit  MOV YH, R19 ; Perform Y/X = Result / 10  MOV YL, R18  LDI XH, 0x00  LDI XL, 0x0A  CALL DIVISION ; R17:R16 = Quotient  ; R19:R18 = Remainder  MOV R25, R16 ; Display quotient (R16 surely < 10)  CALL DISPLAY\_7SEG  CALL DELAY\_4ms  LDI R24, 0 ; Next: the last digit  MOV R25, R18 ; Display remainder  CALL DISPLAY\_7SEG  CALL DELAY\_4ms  JMP READ  ; Y divide by X,  ; Quotient = R17:R16  ; Remainder = R19:R18  DIVISION: PUSH YH  PUSH YL  PUSH XH  PUSH XL  PUSH R10  CLR R10 ; R10 = 0  CLR R17 ; To count "quotient"  CLR R16  DIVISION\_LOOP:  MOV R18, YL  MOV R19, YH ; Save Remainder (Useful for the last iteration)  SUB YL, XL ; Y - X  SBC YH, XH  BRMI DIVISION\_STOP ; If negative, exit  INC R16  ADC R17, R10 ; R17 + carry from INC R18  RJMP DIVISION\_LOOP  DIVISION\_STOP:  POP R10  POP XL  POP XH  POP YL  POP YH  RET  LED\_7SEG\_PORT\_INIT:  PUSH R20 ; Save R20 before using it  LDI R20, 0xFF ; Set LED7SEG port as output  OUT LED7SEG\_DDR, R20  ; Set as output for signal each common Anode of 7SEG LEDs  LDI R20, Common\_Anode\_DDR  ORI R20, (1 << nLE0\_PIN) | (1 << nLE1\_PIN)  OUT LED7SEG\_LATCH\_DDR, R20  POP R20  RET  ; INPUT: R25 contains the value to display  ; R24 contain the index for LED (from 0 -> 3, as for LED\_0 to LED\_3)  ; If using a common cathode display, invert the values in the table above  ; Look up the 7-segment code for the value in R18  DISPLAY\_7SEG:  PUSH R16 ; Save R16 and R18 before using it  PUSH R18  CLR R16  LDI ZH, HIGH(TABLE\_7SEG\_DATA << 1) ; Z = base address of the look-up table  LDI ZL, LOW(TABLE\_7SEG\_DATA << 1)  ADD ZL, R25 ; Add R25 to Z (16-bit)  ADC ZH, R16 ; Add carry to ZH if needed  LPM R18, Z ; Load the code to the 7SEG pins  OUT LED7SEG\_PORT, R18  SBI LED7SEG\_LATCH\_PORT, nLE0\_PIN ; Pulse the latch to update  NOP  NOP  CBI LED7SEG\_LATCH\_PORT, nLE0\_PIN  ; Z = base address of the look-up control table ;  LDI ZH, HIGH(TABLE\_7SEG\_CONTROL << 1)  LDI ZL, LOW(TABLE\_7SEG\_CONTROL << 1) ;  ADD ZL, R24 ; Add R25 to Z (16-bit)  ADC ZH, R16 ; Add carry to ZH if needed    LPM R18, Z ; Load the code to the 7SEG pins  OUT LED7SEG\_LATCH\_PORT, R18  SBI LED7SEG\_LATCH\_PORT, nLE1\_PIN ; Pulse the latch to update  NOP  NOP  CBI LED7SEG\_LATCH\_PORT, nLE1\_PIN  POP R18 ; Restore the temporary register  POP R16  RET    .EQU TCCR0B\_mode = 0b00000100  .EQU TCNT\_init = 100 ; 0 to 255  DELAY\_4ms:  PUSH R16  LDI R16, TCNT\_init ; Set initial value for Timer (TNCT)  OUT TCNT0, R16  LDI R16, 0b00000000 ; Choose Normal mode  OUT TCCR0A, R16  LDI R16, TCCR0B\_mode ; Choose Prescale and start Timer  OUT TCCR0B, R16  AGAIN: SBIS TIFR0, TOV0 ; Check for Overflow  RJMP AGAIN    LDI R16, 0 ; Stop the TImer  OUT TCCR0B, R16  LDI R16, (1 << TOV0) ; Clear TImer overflow flag  OUT TIFR0, R16    POP R16  RET |

**SIMULATION RESULT:**

* PORTA = 3 (0b00000011)



* PORTA = 53 (0b00110101)



* PORTA = 255 (0b11111111)

A computer screen shot of a circuit board

Description automatically generated

* PORTA = 0 (0b00000000)

A computer screen shot of a circuit board

Description automatically generated

# EXPERIMENT 3:

1. Answer the following questions:
2. How many bits is the value of PORTA \* 9?
3. How can you display each digit on the 4 7-segment LEDs?
4. Provide the source code with comments.

|  |
| --- |
| .INCLUDE "M324PADEF.INC" ; INCLUDE ATMEGA324PA DEFINITIONS  .ORG 0x0000 ; INTERRUPT VECTOR TABLE  RJMP RESET\_HANDLER ; RESET  .ORG 0x001A ; Timer 1 Overflow Interrupt Vector  RJMP TIMER1\_COMP\_ISR  ; J38 CONNECT TO PORTD  ; CLEAR signal pin to PIN 0 of PORTB  ; SHIFT CLOCK pin to PIN 1 of PORTB  ; LATCH pin to PIN 0 of PORTB  ; SHIFT DATA pin to PIN 3 of PORTB    .EQU SHIFT\_REGISTER\_PORT = PORTB  .EQU SHIFT\_REGISTER\_DDR = DDRB  .EQU CLEAR\_SIGNAL\_PIN = 3 ; SET CLEAR SIGNAL PIN TO PIN 3 OF PORTB  .EQU SHIFT\_CLK\_PIN = 2 ; SET SHIFT CLOCK PIN TO PIN 2 OF PORTB  .EQU LATCH\_PIN = 1 ; SET LATCH PIN TO PIN 1 OF PORTB  .EQU SHIFT\_DATA\_PIN = 0 ; SET SHIFT DATA PIN TO PIN 0 OF PORTB    .EQU LEDMATRIX\_PORT = PORTD  .EQU LEDMATRIX\_DDR = DDRD  RESET\_HANDLER:  LDI R16, 0xFF  OUT DDRC, R16  LDI R16, 0x00  MOV R10, R16  LDI R16, HIGH(RAMEND) ; Initialize stack pointer  OUT SPH, R16  LDI R16, LOW(RAMEND)  OUT SPL, R16  CALL SHIFT\_REGISTER\_INIT\_PORT  CALL SHIFT\_REGISTER\_CLEAR\_DATA  CALL INIT\_TIMER1\_CTC  SEI ; ENABLE GLOBAL INTERRUPTS  CALL LEDMATRIX\_PORT\_INIT  MAIN: JMP MAIN  SHIFT\_REGISTER\_INIT\_PORT: ; Initialize PORTS as Outputs  PUSH R24  LDI R24, (1<<CLEAR\_SIGNAL\_PIN)|(1<<SHIFT\_CLK\_PIN)|(1<<LATCH\_PIN)|(1<<SHIFT\_DATA\_PIN)  OUT SHIFT\_REGISTER\_DDR, R24 ; Set SHIFT\_REGISTER\_PORT as OUTPUT  POP R24  RET    SHIFT\_REGISTER\_CLEAR\_DATA:  ; Pulse CLEAR pin of the Shift Register (ACTIVE LOW)  CBI SHIFT\_REGISTER\_PORT, CLEAR\_SIGNAL\_PIN  NOP  SBI SHIFT\_REGISTER\_PORT, CLEAR\_SIGNAL\_PIN  RET  ; Shift out R27 to bar LED  SHIFT\_REGISTER\_OUT\_DATA:  PUSH R18  CBI SHIFT\_REGISTER\_PORT, SHIFT\_CLK\_PIN ; Clk = Low  CBI SHIFT\_REGISTER\_PORT, LATCH\_PIN  LDI R18, 8 ; Counter to track 8-bit shift  SHIFT\_LOOP:  SBRC R27, 7 ; Check if the MSB of shiftdata is 1  SBI SHIFT\_REGISTER\_PORT, SHIFT\_DATA\_PIN ; Set shift data pin to high  NOP  SBI SHIFT\_REGISTER\_PORT, SHIFT\_CLK\_PIN ; Set shift clock pin to high  LSL R27 ; Shift left  CBI SHIFT\_REGISTER\_PORT, SHIFT\_CLK\_PIN ; Set shift clock pin to low  NOP  CBI SHIFT\_REGISTER\_PORT, SHIFT\_DATA\_PIN ; Set shift data pin to low  DEC R18  BRNE SHIFT\_LOOP  SBI SHIFT\_REGISTER\_PORT, LATCH\_PIN ; Pulse Latch pin to update Output  CBI SHIFT\_REGISTER\_PORT, LATCH\_PIN  POP R18  RET    LEDMATRIX\_COL\_CONTROL:  .DB 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01 ; LOOKUP table for column control  LEDMATRIX\_FONT\_A:  .DB 0b11111100, 0b00010010, 0b00010001, 0b00010001, 0b00010010, 0b11111100, 0b00000000, 0b00000000 ; LOOKUP TABLE FOR FONT  .DSEG  .ORG SRAM\_START ; Starting address is 0x100  LEDMATRIXBUFFER: .BYTE 8  LEDMATRIX\_COL\_index: .BYTE 1  .CSEG  .ALIGN 2  LEDMATRIX\_PORT\_INIT:  PUSH R20  PUSH R21  LDI R20, 0xFF ; Set PORT as OUTPUT  OUT LEDMATRIX\_DDR, R20  OUT LEDMATRIX\_PORT, R20 ; Clear LED matrix PORT before sending data  LDI R20, 0 ; COL index START AT 0  LDI ZH , HIGH(LEDMATRIX\_COL\_index)  LDI ZL , LOW (LEDMATRIX\_COL\_index)  ST Z , R20 ; Initialize Column index VAR to start at 0  LDI ZH , HIGH(LEDMATRIX\_FONT\_A << 1) ; Z Register point to fonta value  LDI ZL , LOW (LEDMATRIX\_FONT\_A << 1)  LDI YH , HIGH(LEDMATRIXBUFFER) ; Y Register point to fonta value  LDI YL , LOW (LEDMATRIXBUFFER)  LDI R20, 8 ; Column Tracker/Counter  LEDMATRIX\_PORT\_INIT\_LOOP: ; COPY FONT TO DISPLAY BUFFER  LPM R21, Z+ ; Load all 8 bytes of Column data for displaying letter A  ST Y+ , R21 ; Store those all 8 bytes of Column data into Y (VAR)  DEC R20  CPI R20, 0 ; If the Final Column is reached Exit Loop  BRNE LEDMATRIX\_PORT\_INIT\_LOOP  POP R21  POP R20  RET  ; INPUT: R27 CONTAINS THE VALUE TO DISPLAY  ; R26 CONTAIN THE COL INDEX (3..0)  LEDMATRIX\_DISPLAY\_COL: ; DISPLAY A COLLUMN OF LED MATRIX  PUSH R16 ; SAVE THE TEMPORARY REGISTER  PUSH R27  LDI ZH , HIGH(LEDMATRIX\_COL\_CONTROL << 1)  LDI ZL , LOW (LEDMATRIX\_COL\_CONTROL << 1)  CLR R16 ; Clear R16 to and add only carry from ZL to ZH  ADD ZL, R26  ADC ZH, R16  LPM R27, Z  COM R27  OUT LEDMATRIX\_PORT, R27  POP R27  CALL SHIFT\_REGISTER\_OUT\_DATA ; Output of Shift Register = R27  POP R16 ; RESTORE THE TEMPORARY REGISTER  RET ; RETURN FROM THE FUNCTION  TIMER1\_COMP\_ISR:  PUSH R16  PUSH R26  PUSH R27  LDI ZH , HIGH(LEDMATRIX\_COL\_index) ; Column index = 0 (initially)  LDI ZL , LOW (LEDMATRIX\_COL\_index)  LD R16, Z ; R16 = Column index  MOV R26, R16 ; R26 = R16 = column index  ISR\_LOOP\_DISPLAY\_COL:  CLR R16  LDI ZH , HIGH(LEDMATRIXBUFFER)  LDI ZL , LOW (LEDMATRIXBUFFER)  ADD ZL , R26 ; Add the column index to get correct column data  ADC ZH , R16  LD R27, Z ; Load to column data to R27  CALL LEDMATRIX\_DISPLAY\_COL ; Display content in R27 to Shift Register    INC R26 ; Increment column index  CPI R26, 8  BRNE ISR\_LOOP\_DISPLAY\_COL ; Exit and Reset R26 when the last column is reached    TIMER1\_COMP\_ISR\_CONT:  LDI R26, 0  LDI ZH , HIGH(LEDMATRIX\_COL\_index)  LDI ZL , LOW (LEDMATRIX\_COL\_index)  ST Z , R26 ; Save the reset column index (R26 = 0)  POP R27  POP R26  POP R16  RETI    INIT\_TIMER1\_CTC:  PUSH R16  LDI R16, HIGH(2500) ; Set OCR1A of timer1 = 2500  STS OCR1AH, R16  LDI R16, LOW (2500)  STS OCR1AL, R16    LDI R16, 0x00 ; Select CTC mode, No Prescale  STS TCCR1A, R16  LDI R16, (1 << CS10) | (1<< WGM12)  STS TCCR1B, R16 ; Start timer1  LDI R16, (1 << OCIE1A) ; Enable Output Comare Interrupt for OCR1A  STS TIMSK1, R16  POP R16  RET |

**SIMULATION RESULT:**

A computer screen shot of a computer

Description automatically generated