# objective:

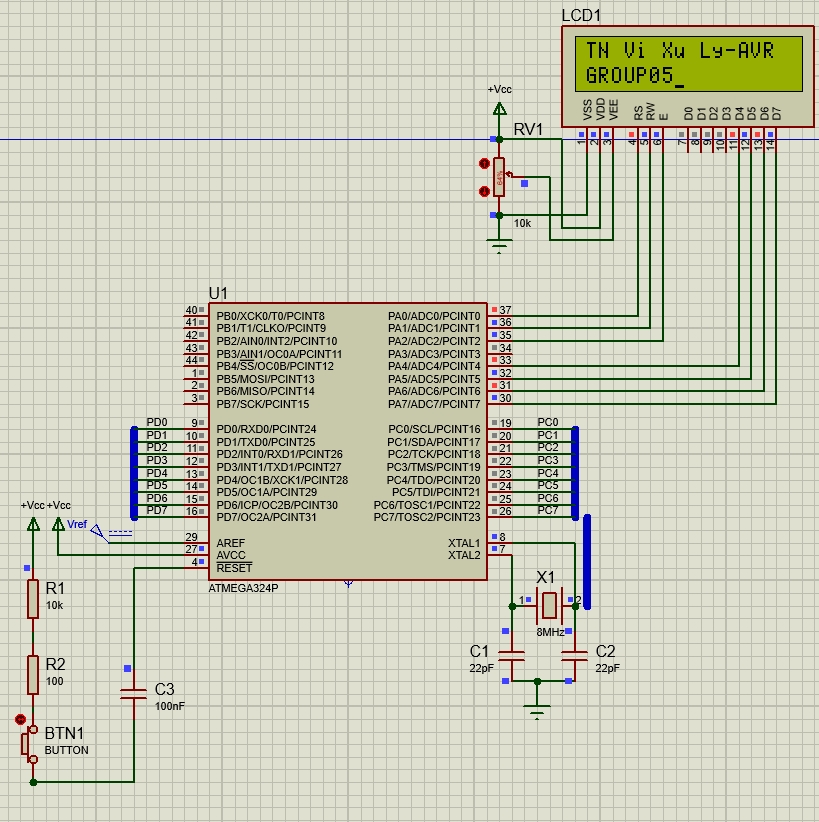
* Understand how to debounce a button.
* Understand how to communicate with an LCD.
* Understand how to communicate with a single button.
* Understand how to communicate with a matrix keypad.

# Reference:

* Lab manual chapter 1, 2 , 3 ,6

# EXPERIMENT 1:

1. Connect one AVR PORT to J33 (LCD control header) on the experimental kit.
2. Use sample programs from the experiment guide, write a program to initialize the LCD and display the following on the LCD (XX is the group number):



| EX VXL-AVR  GROUP: XX |
| --- |

# EXPERIMENT 2:

1. Connect a switch to a port pin of AVR, connect a BAR LED module to another AVR port, and connect an LCD to a port of AVR.
2. Write a program to count the number of button presses and display the result on the BAR LED and LCD (without debouncing).
3. Add button debounce functionality to the program.
4. Execute the program, press/release the button, and observe the results.

# EXPERIMENT 3:

1. Connect signals from one port of AVR to the matrix keypad module, and connect the BAR LED and LCD to two different ports of AVR.
2. Write a subroutine SCANKEY to scan the matrix keypad and return a value from 0x0 to 0xF corresponding to the pressed key's code. If no key is pressed, return 0xFF. The returned value should be stored in R24.
3. Using this subroutine, write a program to scan the keypad and display the read value on the BAR LED and LCD.
4. Execute the program and observe the results

# EXPERIMENT 1:

1. Answer the following questions:
   1. How does the LCD distinguish between command and data?

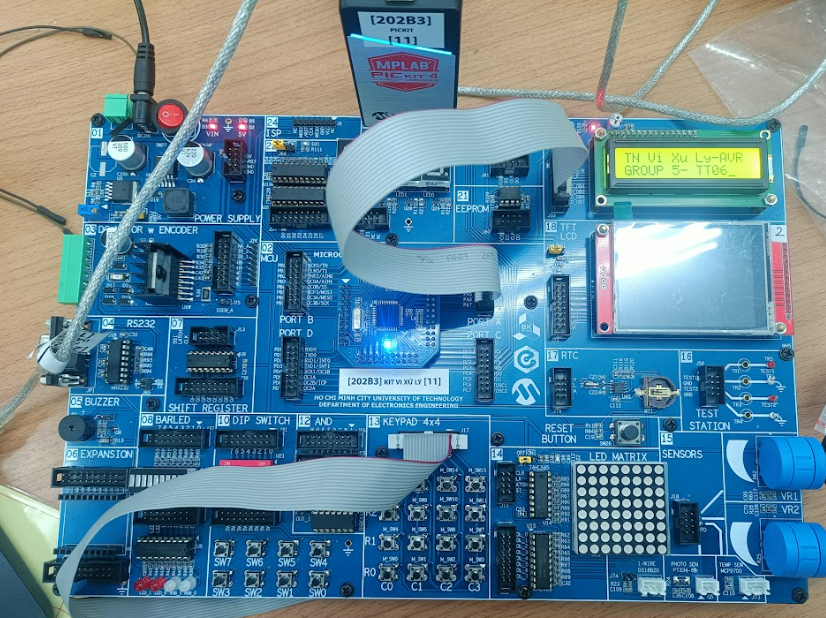
| To send a command to the LCD, you place the command code on the DATA signals, set RS to 0 to indicate the command register, set RW to 0 for write mode, and initially keep EN at 0. Then, you generate a pulse on EN by toggling it from 0 to 1 and back to 0 to transmit the command.  To write data to the LCD for display, you set the data on the DATA signals, set RS to 1 to indicate the data register, set RW to 0 for write mode, and initially keep EN at 0. Again, you create a pulse on EN by toggling it from 0 to 1 and back to 0 to send the data to the data register.  In summary, the LCD distinguishes between commands and data based on the RS signal. When RS is 0, it expects a command, and when RS is 1, it receives data for display. |
| --- |

* 1. Besides checking the BUSY bit, what other method ensures that the LCD is ready to receive data/command?

| By reading the command register, you may determine if the LCD is prepared to accept a command. Set the RS signal to 0 to choose the command register, the RW signal to 1 to choose the read mode, and the EN signal to 0 to start. Next, you establish a pulse.  EN is activated by switching the EN signal from 0 to 1 and back again. The LCD may now be seen. The data pins will have data available. |
| --- |

* 1. Describe the connections on the experimental kit.

| JTAG is chosen, and an 8x8 BUS is used to link the PORTA directly to the LCD.  Connect PA0 to LCD\_RS  Connect PA1 to LCD\_RW  Connect PA2 to LCD\_EN  Connect PA4…PA7 to LCD\_DB4…LCD\_DB7 |
| --- |



* 1. Source code for the program with comments.

| .include "m324padef.inc" ; Include Atmega324pa definitions  .org 0x0000 ; interrupt vector table  rjmp reset\_handler ; reset  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program ID \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .org INT\_VECTORS\_SIZE  course\_name**:**  .db "TN Vi Xu Ly-AVR"**,**0  course\_time**:**  .db "HK2 2022-2023"**,**0  reset\_handler**:**  **call** LCD\_Init  ; display the first line of information  ldi ZH**,** high**(**course\_name**)** ; point to the information that is to be  ldi ZL**,** low**(**course\_name**)**  **call** LCD\_Send\_String  ldi r16**,**1  ldi r17**,**0  **call** LCD\_Move\_Cursor  ldi ZH**,** high**(**course\_time**)** ; point to the information that is to be  ldi ZL**,** low**(**course\_time**)**  **call** LCD\_Send\_String  start**:**  rjmp start    LCD\_Init**:**  ; Set up data direction register for Port A  ldi r16**,** 0b11110111 ; set PA7-PA4 as outputs, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ; Wait for LCD to power up  **call** DELAY\_10MS  **call** DELAY\_10MS  ; Send initialization sequence  ldi r16**,** 0x02 ; Function Set: 4-bit interface  **call** LCD\_Send\_Command  ldi r16**,** 0x28 ; Function Set: enable 5x7 mode for chars  **call** LCD\_Send\_Command  ldi r16**,** 0x0E ; Display Control: Display OFF, Cursor ON  **call** LCD\_Send\_Command  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  **ret**    .def LCDData **=** r16  LCD\_Send\_String**:**  **push** ZH ; preserve pointer registers  **push** ZL  **push** LCDData  ; fix up the pointers for use with the 'lpm' instruction  **lsl** ZL ; shift the pointer one bit left for the  **rol** ZH  ; write the string of characters  LCD\_Send\_String\_01**:**  lpm LCDData**,** Z**+** ; get a character  cpi LCDData**,** 0 ; check for end of string  breq LCD\_Send\_String\_02 ; done  ; arrive here if this is a valid character  **call** LCD\_Send\_Data ; display the character  rjmp LCD\_Send\_String\_01 ; not done, send another character  ; arrive here when all characters in the message have been sent to the LCD module  LCD\_Send\_String\_02**:**  **pop** LCDData  **pop** ZL ; restore pointer registers  **pop** ZH  **ret**    LCD\_Move\_Cursor**:**  cpi r16**,**0 ;check if first row  brne LCD\_Move\_Cursor\_Second  andi r17**,** 0x0F  ori r17**,**0x80  **mov** r16**,**r17  ; Send command to LCD  **call** LCD\_Send\_Command  **ret**  LCD\_Move\_Cursor\_Second**:**  cpi r16**,**1 ;check if second row  brne LCD\_Move\_Cursor\_Exit ;else exit  andi r17**,** 0x0F  ori r17**,**0xC0  **mov** r16**,**r17  ; Send command to LCD  **call** LCD\_Send\_Command  LCD\_Move\_Cursor\_Exit**:**  ; Return from function  **ret**    .equ LCDPORT **=** PORTA ; Set signal port reg to PORTA  .equ LCDPORTDIR **=** DDRA ; Set signal port dir reg to PORTA  .equ LCDPORTPIN **=** PINA ; Set clear signal port pin reg to PORTA  .equ LCD\_RS **=** PINA0  .equ LCD\_RW **=** PINA1  .equ LCD\_EN **=** PINA2  .equ LCD\_D7 **=** PINA7  .equ LCD\_D6 **=** PINA6  .equ LCD\_D5 **=** PINA5  .equ LCD\_D4 **=** PINA4    DELAY\_10MS**:**    LDI R21**,**80 ;1MC  L1**:** LDI R20**,**250 ;1MC  L2**:** **DEC** R20 ;1MC  **NOP** ;1MC  BRNE L2 ;2/1MC  **DEC** R21 ;1MC  BRNE L1 ;2/1MC  **RET** ;4MC    LCD\_Send\_Command**:**  **push** r17  **call** LCD\_wait\_busy ; check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS low to select command register  ; Set RW low to write to LCD  andi r17**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r17  **nop**  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  swap r16  andi r16**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r16  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    LCD\_Send\_Data**:**  **push** r17  **call** LCD\_wait\_busy ;check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17**,**0xF0  ori r17**,**0x01  ; Send data to LCD  **out** LCDPORT**,** r17  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  ; Delay for command execution  ;send the lower nibble  **nop**  swap r16  andi r16**,**0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16**,**0xF0  ori r16**,**0x01  ; Send command to LCD  **out** LCDPORT**,** r16  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    LCD\_wait\_busy**:**  **push** r16  ldi r16**,** 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b11110010 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **nop**  LCD\_wait\_busy\_loop**:**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  **in** r16**,** LCDPORTPIN  cbi LCDPORT**,** LCD\_EN  **nop**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  **nop**  andi r16**,**0x80  cpi r16**,**0x80  breq LCD\_wait\_busy\_loop  ldi r16**,** 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b00000000 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **pop** r16  **ret** |
| --- |

# EXPERIMENT 2:

1. Answer the following questions:
   1. What happens when button debouncing is not implemented?

| - Inaccurate inputs and outputs that are unreliable  - Flickering display  - False triggering  When dealing with mechanical key presses, debouncing is important since they produce numerous pulses when pushed or released, which causes erroneous counting. Because mechanical key pushes generate numerous pulses each time a key is pushed or released, not applying debounce can lead to erroneous counting. |
| --- |

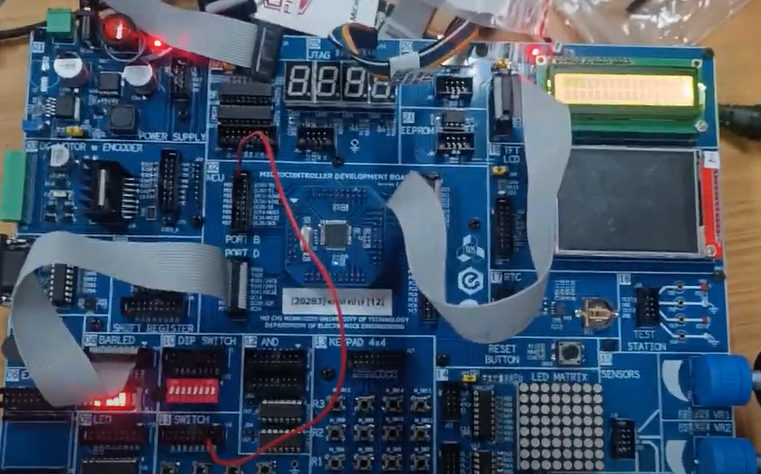
* 1. Describe the connections on the experimental kit.

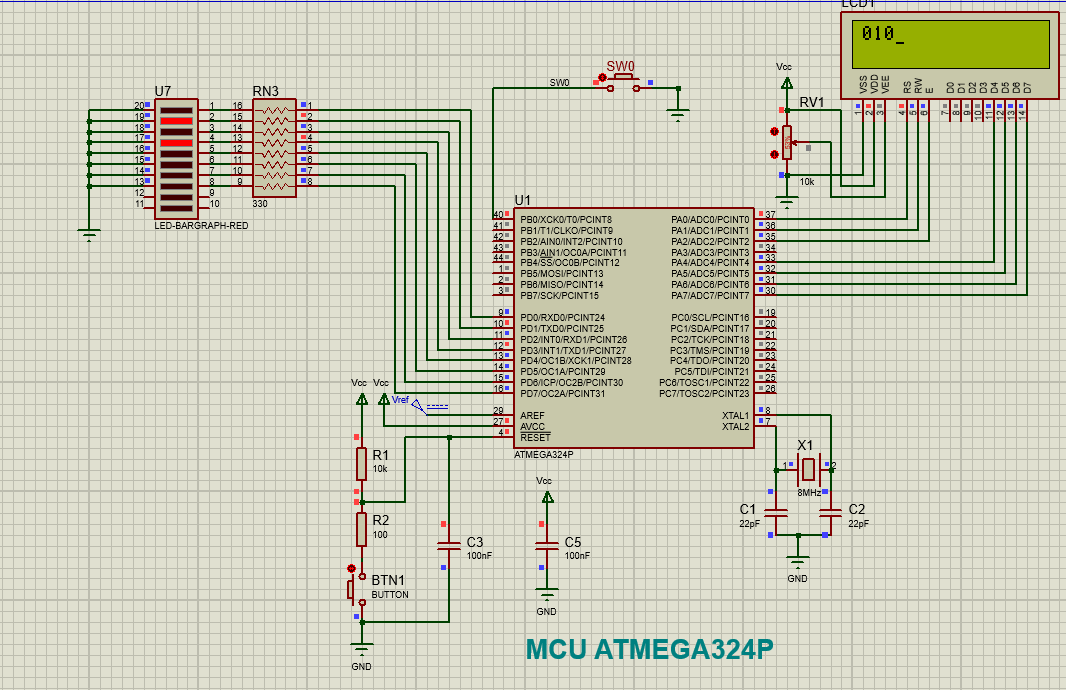
| Choosing JTAG, using 8x8 BUS to connect PORTA directly to LCD  Connect PA0 to LCD\_RS  Connect PA1 to LCD\_RW  Connect PA2 to LCD\_EN  Connect PA4…PA7 to LCD\_DB4…LCD\_DB7  Using 8x8 BUS to connect PORTD to LED  Connect PB0 to switch |
| --- |

* 1. Source code for the program without button debouncing and comments.

| .org 0  rjmp start  .equ LCDPORT **=** PORTA ; Set signal port reg to PORTA  .equ LCDPORTDIR **=** DDRA ; Set signal port dir reg to PORTA  .equ LCDPORTPIN **=** PINA ; Set clear signal port pin reg to PORTA  .equ LCD\_RS **=** PINA0  .equ LCD\_RW **=** PINA1  .equ LCD\_EN **=** PINA2  .equ LCD\_D7 **=** PINA7  .equ LCD\_D6 **=** PINA6  .equ LCD\_D5 **=** PINA5  .equ LCD\_D4 **=** PINA4  .def LCDData **=** r16  start**:**  cbi DDRB**,** 0 ; Set PORTB as input  sbi PORTB**,** 0 ; Enable pull-up resistor  ldi r16**,** 0xff  **out** DDRD**,** r16 ; Set PORTD as output to LED  clr r23  **call** LCD\_Init  Sample**:**  **call** No\_Debouncing  **inc** r23  **out** PORTD**,** r23  ldi r16**,** 0x0E ; Display Control: Display OFF, Cursor ON  **call** LCD\_Send\_Command  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  **mov** r18**,** r23  ldi r19**,** 48  ldi r20**,** 100  clr r21  **call** Loop\_quotient  **add** r21**,** r19  **mov** r16**,** r21  **call** LCD\_Send\_Data  ldi r20**,** 10  clr r21  **call** Loop\_quotient  **add** r21**,** r19  **mov** r16**,** r21  **call** LCD\_Send\_Data  **add** r18**,** r19  **mov** r16**,** r18  **call** LCD\_Send\_Data  rjmp Sample    LCD\_Init**:**  ; Set up data direction register for PortA  ldi r16**,** 0b11110111 ; set PA7-PA4 as outputs, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ; Wait for LCD to power up  **call** Delay10ms  **call** Delay10ms  ; Send initialization sequence  ldi r16**,** 0x02 ; Function Set: 4-bit interface  **call** LCD\_Send\_Command  ldi r16**,** 0x28 ; Function Set: enable 5x7 mode for chars  **call** LCD\_Send\_Command  ldi r16**,** 0x0E ; Display Control: Display OFF, Cursor ON  **call** LCD\_Send\_Command  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  **ret**    LCD\_Send\_Command**:**  **push** r17  **call** LCD\_wait\_busy ; check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS low to select command register  ; Set RW low to write to LCD  andi r17**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r17  **nop**  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  swap r16  andi r16**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r16  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  ;nop  ;nop  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    LCD\_wait\_busy**:**  **push** r16  ldi r16**,** 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b11110010 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **nop**  LCD\_wait\_busy\_loop**:**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  **in** r16**,** LCDPORTPIN  cbi LCDPORT**,** LCD\_EN  **nop**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  **nop**  andi r16**,**0x80  cpi r16**,**0x80  breq LCD\_wait\_busy\_loop  ldi r16**,** 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  **di** r16**,**0b00000000 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **pop** r16  **ret**    LCD\_Send\_Data**:**  **push** r17  **call** LCD\_wait\_busy ;check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17**,**0xF0  ori r17**,**0x01  ; Send data to LCD  **out** LCDPORT**,** r17  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  ; Delay for command execution  ;send the lower nibble  **nop**  swap r16  andi r16**,**0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16**,**0xF0  ori r16**,**0x01  ; Send command to LCD  **out** LCDPORT**,** r16  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    Delay10ms**:**  ldi r21**,**80 ;1MC  L1**:**  ldi r20**,**250 ;1MC  L2**:**  **dec** r20 ;1MC  **nop** ;1MC  brne L2 ;2/1MC  **dec** r21 ;1MC  brne L1 ;2/1MC  **ret** ;4MC    ; Subroutine to obtain the quotient (R21) between dividend (R18) and divisor (R20)  Loop\_quotient**:**  **sub** r18**,** r20  brcc Inc\_quotient  rjmp Obtain\_quotient  Inc\_quotient**:**  **inc** r21  rjmp Loop\_quotient  Obtain\_quotient**:**  **add** r18**,** r20  **ret**    ; Subroutine to count up without debouncing  No\_Debouncing**:**  Sampling\_ND**:**  sbic PINB**,** 0  rjmp Sampling\_ND  Pressed\_ND**:**  sbis PINB**,** 0  rjmp Pressed\_ND  **ret** |
| --- |

* 1. Source code for the program with button debouncing and comments.





| .org 0  rjmp start  .equ LCDPORT **=** PORTA ; Set signal port reg to PORTA  .equ LCDPORTDIR **=** DDRA ; Set signal port dir reg to PORTA  .equ LCDPORTPIN **=** PINA ; Set clear signal port pin reg to PORTA  .equ LCD\_RS **=** PINA0  .equ LCD\_RW **=** PINA1  .equ LCD\_EN **=** PINA2  .equ LCD\_D7 **=** PINA7  .equ LCD\_D6 **=** PINA6  .equ LCD\_D5 **=** PINA5  .equ LCD\_D4 **=** PINA4  .def LCDData **=** r16    start**:**  cbi DDRB**,** 0 ; Set PORTB as input  sbi PORTB**,** 0 ; Enable pull-up resistor  ldi r16**,** 0xff  **out** DDRD**,** r16 ; Set PORTD as output to LED  clr r23  **call** LCD\_Init  Sample**:**  **call** Debouncing  **inc** r23  **out** PORTD**,** r23  ldi r16**,** 0x0E ; Display Control: Display OFF, Cursor ON  **call** LCD\_Send\_Command  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  **mov** r18**,** r23  ldi r19**,** 48  ldi r20**,** 100  clr r21  **call** Loop\_quotient  **add** r21**,** r19  **mov** r16**,** r21  **call** LCD\_Send\_Data  ldi r20**,** 10  clr r21  **call** Loop\_quotient  **add** r21**,** r19  **mov** r16**,** r21  **call** LCD\_Send\_Data  **add** r18**,** r19  **mov** r16**,** r18  **call** LCD\_Send\_Data  rjmp Sample    LCD\_Init**:**  ; Set up data direction register for PortA  ldi r16**,** 0b11110111 ; set PA7-PA4 as outputs, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ; Wait for LCD to power up  **call** Delay10ms  **call** Delay10ms  ; Send initialization sequence  ldi r16**,** 0x02 ; Function Set: 4-bit interface  **call** LCD\_Send\_Command  ldi r16**,** 0x28 ; Function Set: enable 5x7 mode for chars  **call** LCD\_Send\_Command  ldi r16**,** 0x0E ; Display Control: Display OFF, Cursor ON  **call** LCD\_Send\_Command  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  **ret**    LCD\_Send\_Command**:**  **push** r17  **call** LCD\_wait\_busy ; check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS low to select command register  ; Set RW low to write to LCD  andi r17**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r17  **nop**  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  swap r16  andi r16**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r16  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  ;nop  ;nop  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    LCD\_wait\_busy**:**  **push** r16  ldi r16**,** 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b11110010 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **nop**    LCD\_wait\_busy\_loop**:**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  **in** r16**,** LCDPORTPIN  cbi LCDPORT**,** LCD\_EN  **nop**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  **nop**  andi r16**,**0x80  cpi r16**,**0x80  breq LCD\_wait\_busy\_loop  ldi r16**,** 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b00000000 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **pop** r16  **ret**    LCD\_Send\_Data**:**  **push** r17  **call** LCD\_wait\_busy ;check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17**,**0xF0  ori r17**,**0x01  ; Send data to LCD  **out** LCDPORT**,** r17  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  ; Delay for command execution  ;send the lower nibble  **nop**  swap r16  andi r16**,**0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16**,**0xF0  ori r16**,**0x01  ; Send command to LCD  **out** LCDPORT**,** r16  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    Delay10ms**:**  ldi r21**,**80 ;1MC  L1**:**  ldi r20**,**250 ;1MC  L2**:**  **dec** r20 ;1MC  **nop** ;1MC  brne L2 ;2/1MC  **dec** r21 ;1MC  brne L1 ;2/1MC  **ret** ;4MC  ; Subroutine to obtain the quotient (R21) between dividend (R18) and divisor (R20)    Loop\_quotient**:**  **sub** r18**,** r20  brcc Inc\_quotient  rjmp Obtain\_quotient  Inc\_quotient**:**  **inc** r21  rjmp Loop\_quotient  Obtain\_quotient**:**  **add** r18**,** r20  **ret**    ; Subroutine to count up with debouncing  Debouncing**:**  Sampling\_D**:**  sbic PINB**,** 0  rjmp Sampling\_D  Pressed\_D**:**  sbis PINB**,** 0  rjmp Pressed\_D  **call** Delay10ms  **call** Delay10ms  sbis PINB**,** 0  rjmp Pressed\_D  **ret** |
| --- |

# EXPERIMENT 3:

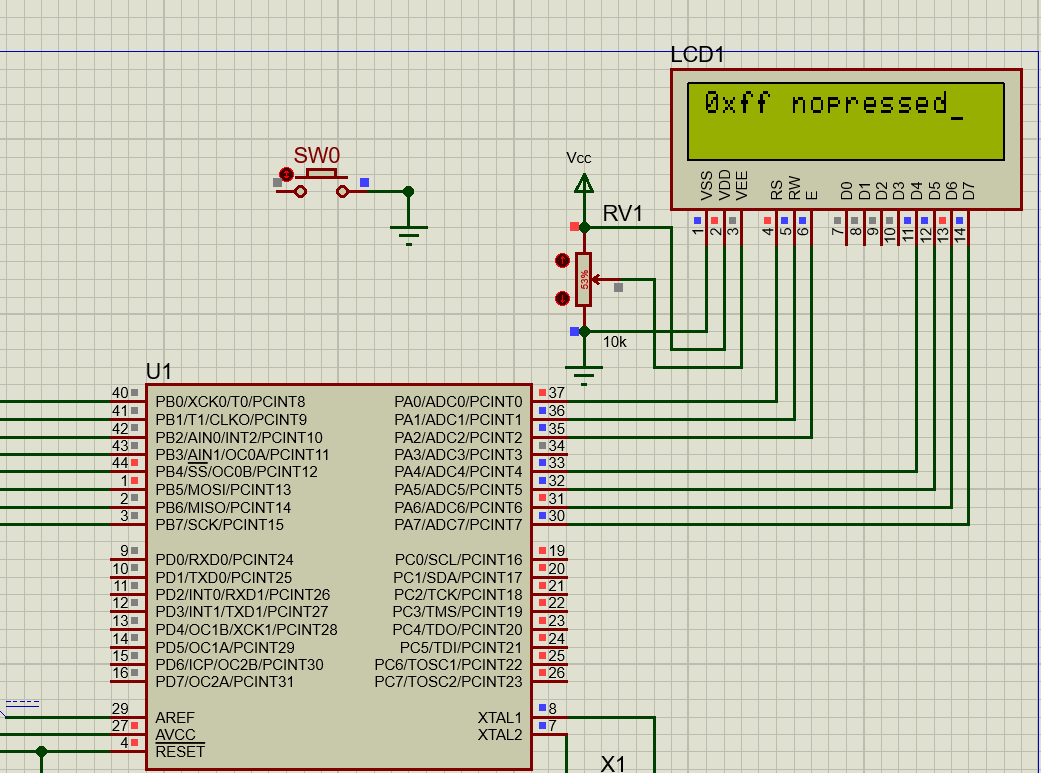
1. Answer the following questions:
   1. Describe the connections of the modules in the experiment.

| Connect Port A to the LCD similarly to Exp 1 and 2  Connect PORTB to Keypad Matrix  Connect PORTD to LED |
| --- |

* 1. Is there any button debounce issue with the matrix keypad? If so, how is it handled?

| All four columns are scanned at a pace that is far quicker than the average rate of key pushes and releases made by a human. This makes sure that any key presses may be accurately detected during the key scanning procedure.  The approach used in this application to deal with the debounce issue entails periodically assessing whether the key has been pushed or released at regular intervals. |
| --- |

* 1. Present the source code of the program with comments.



| .include "m324padef.inc" ; Include Atmega324pa definitions  .org 0x0000 ; interrupt vector table  rjmp start  .org INT\_VECTORS\_SIZE    .equ LCDPORT **=** PORTA ; Set signal port reg to PORTA  .equ LCDPORTDIR **=** DDRA ; Set signal port dir reg to PORTA  .equ LCDPORTPIN **=** PINA ; Set clear signal port pin reg to PORTA  .equ LCD\_RS **=** PINA0  .equ LCD\_RW **=** PINA1  .equ LCD\_EN **=** PINA2  .equ LCD\_D7 **=** PINA7  .equ LCD\_D6 **=** PINA6  .equ LCD\_D5 **=** PINA5  .equ LCD\_D4 **=** PINA4  .def LCDData **=** r16  message**:** .db "0xff nopressed"**,**0    ; Replace with your application code  ; ATmega324PA keypad scan function  ; Scans a 4x4 keypad connected to PORTB  ;C3-C0 connect to PB3-PB0  ;R3-R0 connect to PB7-PB4  ; Returns the key value (0-15) or 0xFF if no key is pressed    start**:**  ldi r16**,** 0xff  **out** ddrc**,** r16  **call** LCD\_Init      **loop:**  **call** keypad\_scan  **out** portc**,** r23  cp r23**,** r25  breq **loop**  **mov** r25**,** r23  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  cpi r23**,** 0xff  brne key\_pressed  ldi ZH**,** high**(**message**)** ; point to the information that is to be displayed  ldi ZL**,** low**(**message**)**  **call** LCD\_Send\_String  rjmp **loop**    key\_pressed**:**  cpi r23**,** 10  brlo less\_than\_A  ldi r24**,** 7  **add** r23**,** r24  less\_than\_A**:**  ldi r24**,** 48  **add** r23**,** r24  **mov** r16**,** r23  **call** LCD\_Send\_Data  rjmp **loop**    ; Replace with your application code  ; ATmega324PA keypad scan function  ; Scans a 4x4 keypad connected to PORTB  ;C3-C0 connect to PB3-PB0  ;R3-R0 connect to PB4-PB7  ; Returns the key value (0-15) or 0xFF if no key is pressed    keypad\_scan**:**  ldi r20**,** 0b00001111 ; set upper 4 bits of PORTB as input with pull-up, lower 4 bits as output  **out** DDRB**,** r20  ldi r20**,** 0b11111111 ; enable pull up resistor  **out** PORTB**,** r20  ldi r22**,** 0b11110111 ; initial col mask  ldi r23**,** 0 ; initial pressed row value  ldi r24**,**3 ;scanning col index  keypad\_scan\_loop**:**  **out** PORTB**,** r22 ; scan current col  **nop** ;need to have 1us delay to stablize  sbic PINB**,** 4 ; check row 0  rjmp keypad\_scan\_check\_col2  rjmp keypad\_scan\_found ; row 0 is pressed  keypad\_scan\_check\_col2**:**  sbic PINB**,** 5 ; check row 1  rjmp keypad\_scan\_check\_col3  ldi r23**,** 1 ; row1 is pressed  rjmp keypad\_scan\_found  keypad\_scan\_check\_col3**:**  sbic PINB**,** 6 ; check row 2  rjmp keypad\_scan\_check\_col4  ldi r23**,** 2 ; row 2 is pressed  rjmp keypad\_scan\_found  keypad\_scan\_check\_col4**:**  sbic PINB**,** 7 ; check row 3  rjmp keypad\_scan\_next\_row  ldi r23**,** 3 ; row 3 is pressed  rjmp keypad\_scan\_found  keypad\_scan\_next\_row**:**  ; check if all rows have been scanned  cpi r24**,**0  breq keypad\_scan\_not\_found  ; shift row mask to scan next row  **ror** r22  **dec** r24 ;increase row index  rjmp keypad\_scan\_loop  keypad\_scan\_found**:**  ; combine row and column to get key value (0-15)  ;key code = row\*4 + col  **lsl** r23 ; shift row value 4 bits to the left  **lsl** r23  **add** r23**,** r24 ; add row value to column value  **ret**  keypad\_scan\_not\_found**:**  ldi r23**,** 0xFF ; no key pressed  **ret**    LCD\_Init**:**  ; Set up data direction register for Port A  ldi r16**,** 0b11110111 ; set PA7-PA4 as outputs, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ; Wait for LCD to power up  **call** DELAY\_10MS  **call** DELAY\_10MS    ; Send initialization sequence  ldi r16**,** 0x02 ; Function Set: 4-bit interface  **call** LCD\_Send\_Command  ldi r16**,** 0x28 ; Function Set: enable 5x7 mode for chars  **call** LCD\_Send\_Command  ldi r16**,** 0x0E ; Display Control: Display OFF, Cursor ON  **call** LCD\_Send\_Command  ldi r16**,** 0x01 ; Clear Display  **call** LCD\_Send\_Command  ldi r16**,** 0x80 ; Clear Display  **call** LCD\_Send\_Command  **ret**    LCD\_Send\_Command**:**  **push** r17  **call** LCD\_wait\_busy ; check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS low to select command register  ; Set RW low to write to LCD  andi r17**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r17  **nop**  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  swap r16  andi r16**,**0xF0  ; Send command to LCD  **out** LCDPORT**,** r16  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  ;nop  ;nop  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**    LCD\_wait\_busy**:**  **push** r16  ldi r16**,** 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b11110010 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **nop**  LCD\_wait\_busy\_loop**:**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  **in** r16**,** LCDPORTPIN  cbi LCDPORT**,** LCD\_EN  **nop**  sbi LCDPORT**,** LCD\_EN  **nop**  **nop**  cbi LCDPORT**,** LCD\_EN  **nop**  andi r16**,**0x80  cpi r16**,**0x80  breq LCD\_wait\_busy\_loop  ldi r16**,** 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output  **out** LCDPORTDIR**,** r16  ldi r16**,**0b00000000 ; set RS=0, RW=1 for read the busy flag  **out** LCDPORT**,** r16  **pop** r16  **ret**    DELAY\_10MS**:**  ldi r21**,**80 ;1MC  L1**:**  ldi r20**,**250 ;1MC  L2**:**  **dec** r20 ;1MC  **nop** ;1MC  brne L2 ;2/1MC  **dec** r21 ;1MC  brne L1 ;2/1MC  **ret** ;4MC    LCD\_Send\_Data**:**  **push** r17  **call** LCD\_wait\_busy ;check if LCD is busy  **mov** r17**,**r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17**,**0xF0  ori r17**,**0x01  ; Send data to LCD  **out** LCDPORT**,** r17  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  ; Delay for command execution  ;send the lower nibble  **nop**  swap r16  andi r16**,**0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16**,**0xF0  ori r16**,**0x01  ; Send command to LCD  **out** LCDPORT**,** r16  **nop**  ; Pulse enable pin  sbi LCDPORT**,** LCD\_EN  **nop**  cbi LCDPORT**,** LCD\_EN  **pop** r17  **ret**      LCD\_Send\_String**:**  **push** ZH ; preserve pointer registers  **push** ZL  **push** LCDData  ; fix up the pointers for use with the 'lpm' instruction  **lsl** ZL ; shift the pointer one bit left for the lpm instruction  **rol** ZH  ; write the string of characters  LCD\_Send\_String\_01**:**  lpm LCDData**,** Z**+** ; get a character  cpi LCDData**,** 0 ; check for end of string  breq LCD\_Send\_String\_02 ; done  ; arrive here if this is a valid character  **call** LCD\_Send\_Data ; display the character  rjmp LCD\_Send\_String\_01 ; not done, send another character  ; arrive here when all characters in the message have been sent to the LCD module  LCD\_Send\_String\_02**:**  **pop** LCDData  **pop** ZL ; restore pointer registers  **pop** ZH  **ret** |
| --- |