# 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?**

**Answer:**

The LCD distinguish between commands and data using the RS pin (input).

* When RS is set to 0, the data provided through pins D0 to D7 is interpreted as a command,
* When RS is set to 1, the input is treated as data, representing the actual character (in ASCII format, stored in DDRAM) to be displayed on the screen.

However, this only works in **WRITE mode**, meaning the **RW pin is set to LOW**.

If the **RW pin is set to HIGH**, the LCD enters **READ mode**. The **EN pin** must be pulsed to initiate the read operation. The **RS pin** is set to **LOW** when checking for the **busy flag** on pin D7 and is set to **HIGH** when reading data from the DDRAM of the LCD, such as ASCII characters or the cursor position.

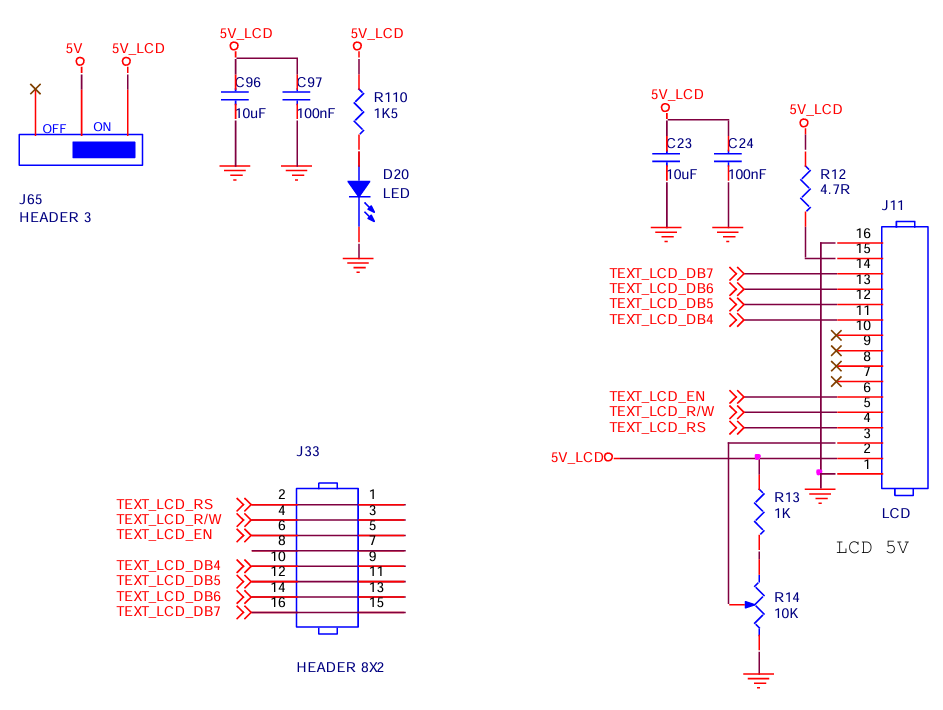
* 1. **Besides checking the BUSY bit, what other method ensures that the LCD is ready to receive data/command?**

**Answer:**

Beside checking the BUSY bit (pin D7) of an LCD, another method to ensure the LCD is ready for new data or commands is to use a fixed delay after each operation. Each command or data write requires processing time, typically ranging from microseconds to milliseconds. One of the longest instruction "Return Home”, takes approximately 1.52 ms to complete. Therefore, by using a fixed delay of 2 to 5 milliseconds after sending a command or data, we can ensure that the LCD has completed its operation and is ready for the next input. This will help avoid the need for continuously polling the BUSY flag.

* 1. **Describe the connections on the experimental kit.**

**Answer:**



The LCD on the experiment kit is configured to operate in 4-bit mode only. It interfaces with the microcontroller through a single header that contains 8 pins, with one pin left unused. When connecting the BUS wire to link the LCD and the AVR, both control and data signals share the same PORT. In this setup, data pins D4 to D7 correspond to the high nibble of that PORT, while the RS, RW, and EN control pins are mapped to pins 0, 1, and 2, respectively. The LCD backlight is permanently powered when the device is on, and the power to the LCD itself is managed by connecting the power pins on header J65. Additionally, there is a potentiometer available for adjusting the display contrast.

* 1. **Source code for the program with comments.**

**Answer:**

# EXPERIMENT 2:

1. Answer the following questions:
   1. What happens when button debouncing is not implemented?
   2. Describe the connections on the experimental kit.
   3. Source code for the program without button debouncing and comments.

|  |
| --- |
| ........................................................................................................................ |

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

**Answer:**

The simplest debouncing method is to read the signal twice consecutively after an appropriate delay. If the results of these two reads are the same, we consider it as the correct result. If the result is incorrect, we read it again for the third time, also after a time period T. If the result matches the second read, we accept it as correct, otherwise, we repeat the process. The time period T is typically chosen as 50ms. We can increase accuracy by reducing the time period T and increasing the number of reads.

|  |
| --- |
| ........................................................................................................................ |

# EXPERIMENT 3:

1. Answer the following questions:
   1. Describe the connections of the modules in the experiment.
   2. Is there any button debounce issue with the matrix keypad? If so, how is it handled?
   3. Present the source code of the program with comments.

|  |
| --- |
| ........................................................................................................................ |