

Lakshman B H

BU21EECE0100553S

8051(MICRO-CONTROLLER) (Trainer: Dr. Jeevan K M)

Learning Objective:

- Understanding how to create patterns of LED blinking using the 8051

microcontroller. Inputs and Outputs:

- Inputs: None
- Outputs: LED states (ON/OFF)

Logic:

- LEDs are connected to the microcontroller. By assigning a hexadecimal value to the port, the microcontroller converts this value to binary. Each bit in the binary representation controls an individual LED: a bit value of '1' turns the LED on, and '0' turns it off. For example, if $P1 = 0x01$, only the first LED (connected to the least significant bit) will be on, and the rest will be off.
- The other terminals of the LEDs are connected to the ground, creating a simple circuit for controlling the LEDs based on the port value.
- Various LED blinking patterns can be implemented using this basic principle. Two main approaches can be utilized:
 1. Direct assignment of hexadecimal values to the port.
 2. Using bitwise left shift operations within a loop to create sequential blinking patterns.

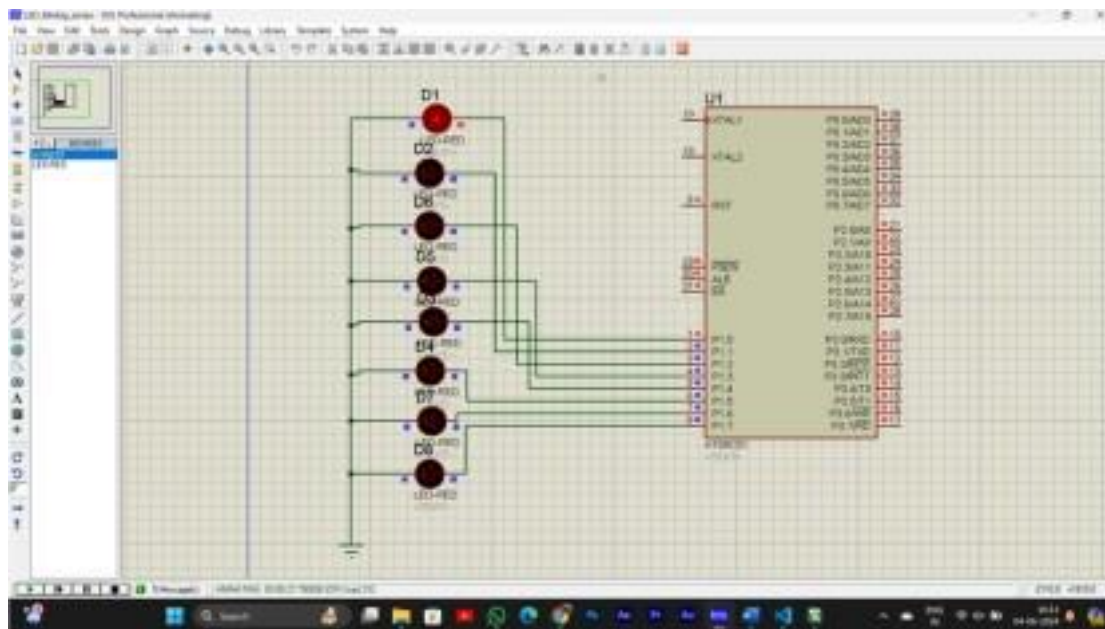
Common Mistakes:

- Syntax Errors: Common when writing embedded C code.
- Indentation Errors: Proper code formatting is crucial to avoid logical errors.
- Port Mismatch: Ensuring the correct port is being used for LED control is essential to achieve the desired output.

```

1 #include <reg51.h>
2 //sbit sw1 = P2^1;
3 //sbit sw2 = P2^2;
4 //void dealy(void);
5 void delay(unsigned int);
6 void main(void)
7 {
8     unsigned char k = 0x01;
9     unsigned int n;
10
11     while(1)
12     {
13         for( n=0;n<7;n++)
14             P1 = k;
15         delay(500);
16         k=k<<1;
17     }
18     if(k==0)
19         k=0x01;
20 }
21
22
23
24
25
26
27
28
29
30 // sw1 = 0;
31 // sw2 = 0;
32 //while(1)
33 {
34     /* if(sw1==0 && sw2 ==0 )
35     {
36         P1 = 0x00;
37     }
38 }

```



ADC Using 8051

Learning Objective:

- Understanding the process of converting an analog signal to a digital signal using the ADC 0808.
- Learning how to interface the ADC 0808 with the 8051

microcontroller. Inputs and Outputs:

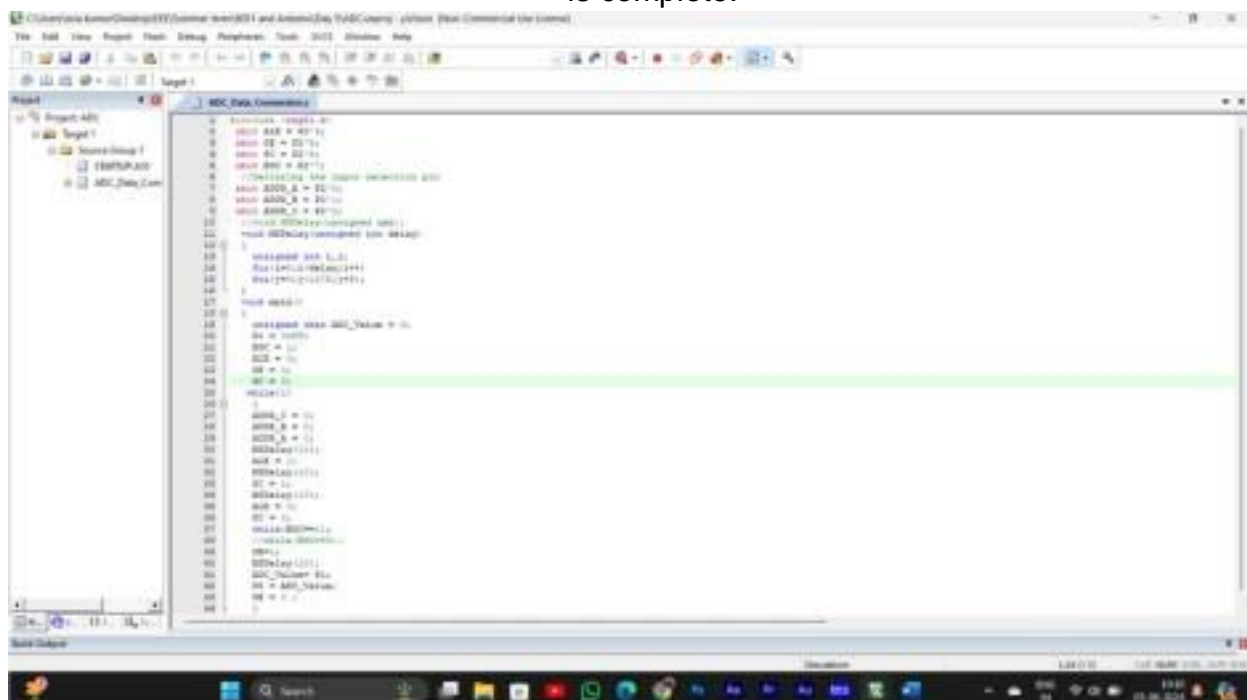
- Input: A potentiometer (1k or 10k ohms).
- Outputs: LEDs.

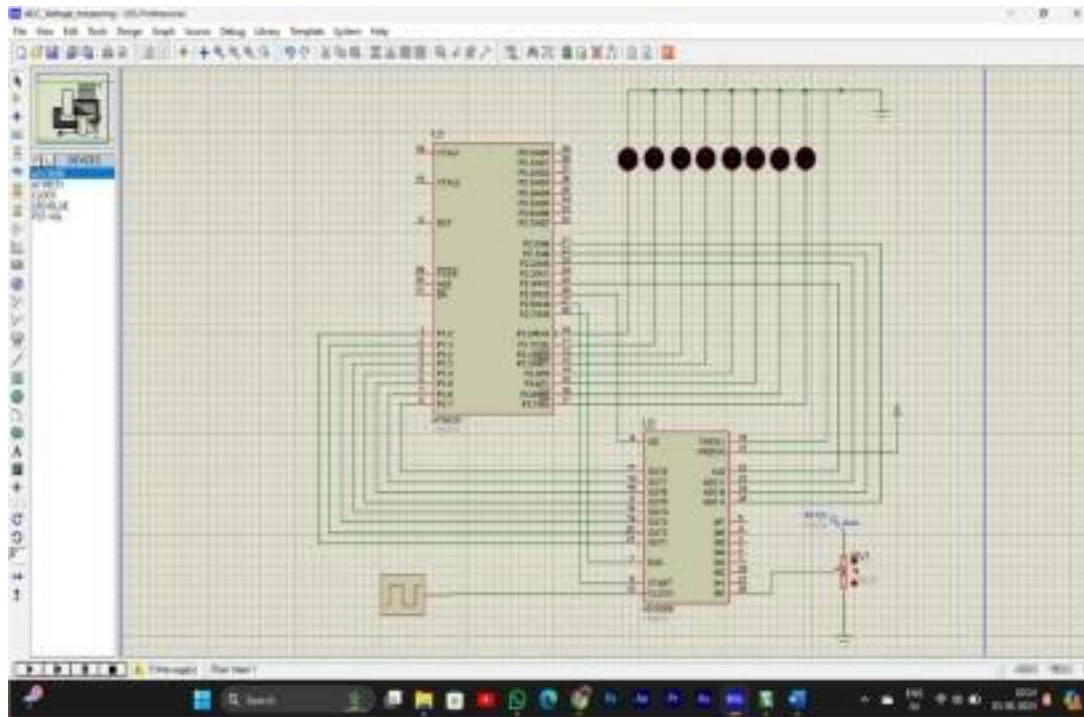
Logic:

- The analog output from the potentiometer is fed into one of the input channels of the ADC 0808.
- The digital output pins (D0-D7) of the ADC 0808 are connected to Port 1 of the 8051 microcontroller.
- The selection lines (A, B, C) of the ADC are connected to Port 2 of the 8051 to select the input channel.
- Additional control lines such as End of Conversion (EOC), Start, Address Latch Enable (ALE), and Clock Pulse are also connected to Port 2.
- The LEDs, which display the digital value of the potentiometer reading, are connected to Port 3 of the 8051 microcontroller.
 - Based on the variation in resistance of the potentiometer, the digital value output from the ADC will change, which in turn will change the state of the LEDs connected to Port 3.

Common Mistakes:

- Selection Line Configuration: Double-check the configuration and connections of the selection lines (A, B, C) to ensure the correct input channel is selected on the ADC 0808.
- EOC Pin Connection: Ensure the End of Conversion (EOC) pin of the ADC 0808 is properly connected to the 8051 to correctly detect when a conversion is complete.





8051

Learning Objective:

- Learning how to control LED blinking using the 8051

microcontroller. Inputs and Outputs:

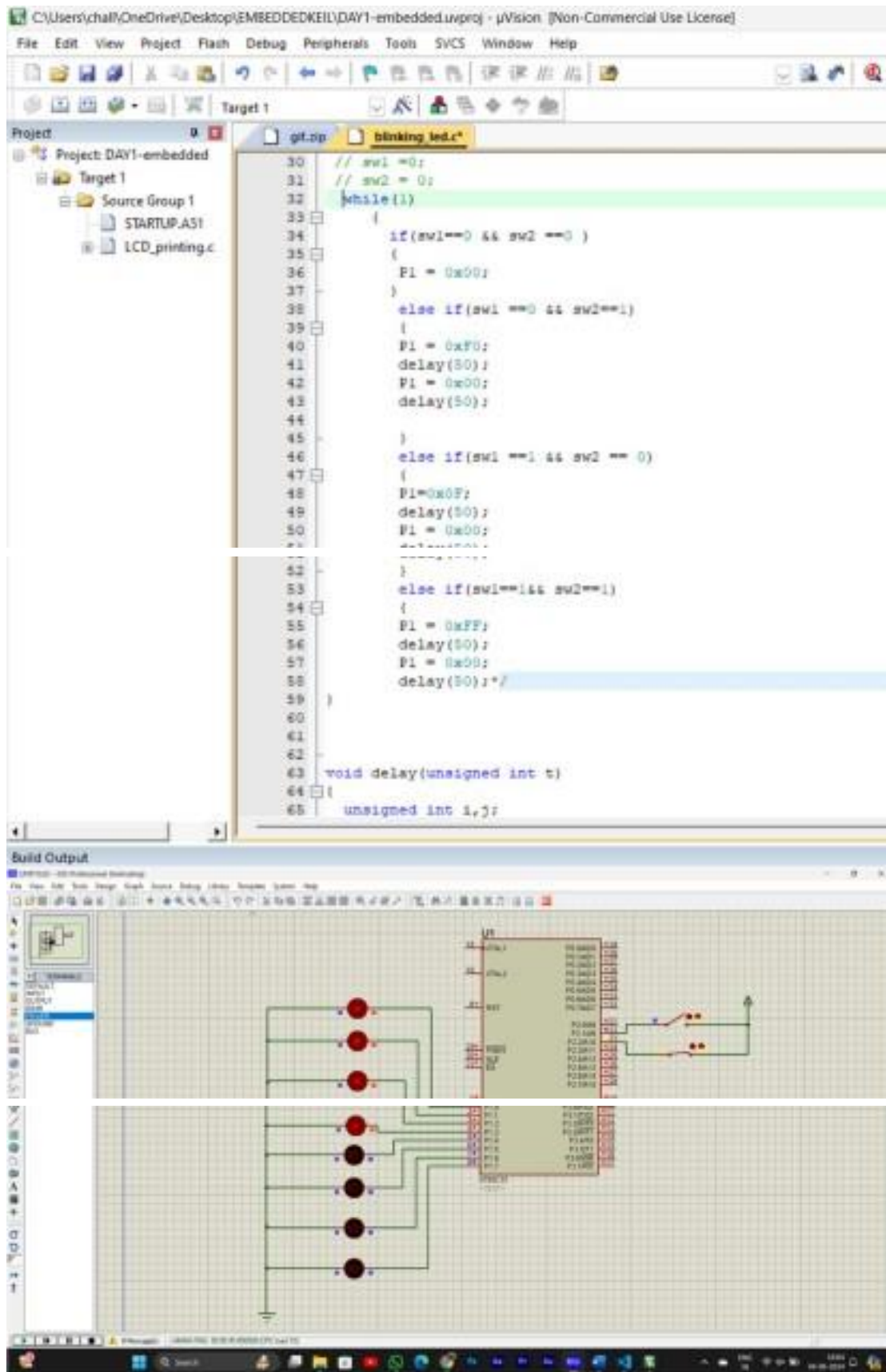
- Inputs: Switches
- Outputs: LEDs

Logic:

- LEDs are connected to Port 1 (P1) of the 8051 microcontroller.
- The code involves passing hexadecimal values to Port 1, which determines the state of each LED.
- A delay function is used to create the blinking effect by toggling the LEDs on

and off. Common Mistakes:

- Project Creation in Keil Software: Errors in creating a project or missing the step of creating a target before running the code.
- Syntax and Indentation Errors: Mistakes in code syntax and improper indentation can lead to compilation errors.
- Port Confusion: Errors in connecting the LEDs to the correct ports on the microcontroller.



Voltage Measurement and LCD displaying(8051)

Learning Objective:

- Understanding how to measure voltage variations using an ADC.
- Displaying the voltage variations on LEDs.
- Displaying the voltage measurement status on an LCD display.

Inputs and Outputs:

- Input: Potentiometer.
- Outputs: LEDs and LCD display.

Logic:

1. Connections:
 - Connect the potentiometer output to an input channel of the ADC 0808.
 - Provide a 5V power supply to the ADC 0808.
 - Connect the digital output pins (D0-D7) of the ADC 0808 to a port on the 8051 microcontroller.
 - Connect the LCD display and LEDs to the 8051 microcontroller.
2. Operation:
 - The ADC converts the analog signal from the potentiometer to a digital signal.
 - The 8051 microcontroller reads the digital signal from the ADC.
 - Based on the digital value, control the LEDs to indicate voltage levels.
 - Convert the digital value to a corresponding voltage value and display it on the LCD.

Common Mistakes:

- Port Mismatch: Ensure that the port names used in the code match the actual circuit connections.
- Proper Connections: Verify that all connections are secure. A single loose connection can lead to data loss and no output.
- Input Levels: Ensure that input levels to the microcontroller and ADC are correct (either high or low as required).

-
- The top screenshot shows the initial assembly code for the 'Blink' program. The code is written in assembly language with comments in Italian. It starts by setting up the LED pin (10) and the initial value (0). It then enters a loop where it toggles the LED state (0 to 1 and 1 to 0) and delays for 1000 units of time. The code is as follows:
- ```

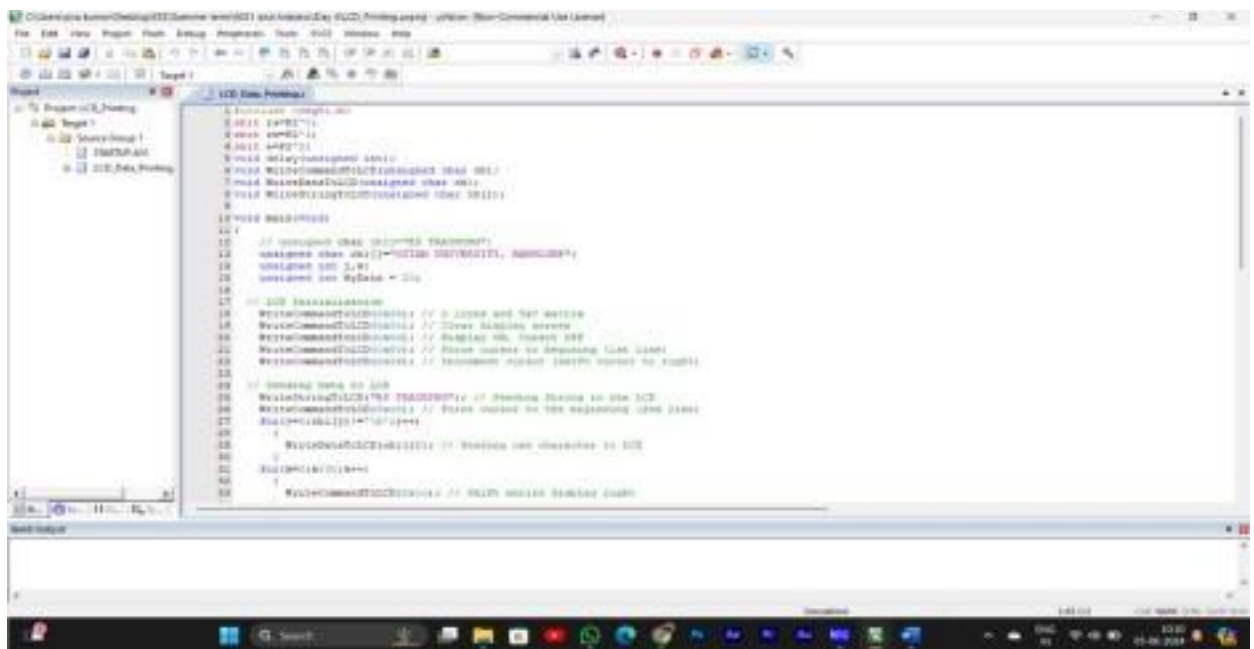
10: // Project: Blink
11: // Target: AVR
12: // Source: Blink.asm
13: // Author: [Name]
14: // Date: [Date]
15: // Description: Blink the LED on pin 10.
16: // Comments: [Comments]
17: //
18: // Pin 10 is connected to the LED.
19: //
20: // Initial value: 0
21: //
22: // Delay: 1000 units of time.
23: //
24: // Position: [Position]
25: //
26: // Comments: [Comments]
27: //
28: // Delay: 1000 units of time.
29: //
30: // Position: [Position]
31: //
32: // Comments: [Comments]
33: //
34: // Delay: 1000 units of time.
35: //
36: // Position: [Position]
37: //
38: // Comments: [Comments]
39: //
40: // Delay: 1000 units of time.
41: //
42: // Position: [Position]
43: //
44: // Comments: [Comments]
45: //
46: // Delay: 1000 units of time.
47: //
48: // Position: [Position]
49: //
50: // Comments: [Comments]
51: //
52: // Delay: 1000 units of time.
53: //
54: // Position: [Position]
55: //
56: // Comments: [Comments]
57: //
58: // Delay: 1000 units of time.
59: //
60: // Position: [Position]
61: //
62: // Comments: [Comments]
63: //
64: // Delay: 1000 units of time.
65: //
66: // Position: [Position]
67: //
68: // Comments: [Comments]
69: //
70: // Delay: 1000 units of time.
71: //
72: // Position: [Position]
73: //
74: // Comments: [Comments]
75: //
76: // Delay: 1000 units of time.
77: //
78: // Position: [Position]
79: //
80: // Comments: [Comments]
81: //
82: // Delay: 1000 units of time.
83: //
84: // Position: [Position]
85: //
86: // Comments: [Comments]
87: //
88: // Delay: 1000 units of time.
89: //
90: // Position: [Position]
91: //
92: // Comments: [Comments]
93: //
94: // Delay: 1000 units of time.
95: //
96: // Position: [Position]
97: //
98: // Comments: [Comments]
99: //
100: // Delay: 1000 units of time.
101: //
102: // Position: [Position]
103: //
104: // Comments: [Comments]
105: //
106: // Delay: 1000 units of time.
107: //
108: // Position: [Position]
109: //
110: // Comments: [Comments]
111: //
112: // Delay: 1000 units of time.
113: //
114: // Position: [Position]
115: //
116: // Comments: [Comments]
117: //
118: // Delay: 1000 units of time.
119: //
120: // Position: [Position]
121: //
122: // Comments: [Comments]
123: //
124: // Delay: 1000 units of time.
125: //
126: // Position: [Position]
127: //
128: // Comments: [Comments]
129: //
130: // Delay: 1000 units of time.
131: //
132: // Position: [Position]
133: //
134: // Comments: [Comments]
135: //
136: // Delay: 1000 units of time.
137: //
138: // Position: [Position]
139: //
140: // Comments: [Comments]
141: //
142: // Delay: 1000 units of time.
143: //
144: // Position: [Position]
145: //
146: // Comments: [Comments]
147: //
148: // Delay: 1000 units of time.
149: //
150: // Position: [Position]
151: //
152: // Comments: [Comments]
153: //
154: // Delay: 1000 units of time.
155: //
156: // Position: [Position]
157: //
158: // Comments: [Comments]
159: //
160: // Delay: 1000 units of time.
161: //
162: // Position: [Position]
163: //
164: // Comments: [Comments]
165: //
166: // Delay: 1000 units of time.
167: //
168: // Position: [Position]
169: //
170: // Comments: [Comments]
171: //
172: // Delay: 1000 units of time.
173: //
174: // Position: [Position]
175: //
176: // Comments: [Comments]
177: //
178: // Delay: 1000 units of time.
179: //
180: // Position: [Position]
181: //
182: // Comments: [Comments]
183: //
184: // Delay: 1000 units of time.
185: //
186: // Position: [Position]
187: //
188: // Comments: [Comments]
189: //
190: // Delay: 1000 units of time.
191: //
192: // Position: [Position]
193: //
194: // Comments: [Comments]
195: //
196: // Delay: 1000 units of time.
197: //
198: // Position: [Position]
199: //
200: // Comments: [Comments]
201: //
202: // Delay: 1000 units of time.
203: //
204: // Position: [Position]
205: //
206: // Comments: [Comments]
207: //
208: // Delay: 1000 units of time.
209: //
210: // Position: [Position]
211: //
212: // Comments: [Comments]
213: //
214: // Delay: 1000 units of time.
215: //
216: // Position: [Position]
217: //
218: // Comments: [Comments]
219: //
220: // Delay: 1000 units of time.
221: //
222: // Position: [Position]
223: //
224: // Comments: [Comments]
225: //
226: // Delay: 1000 units of time.
227: //
228: // Position: [Position]
229: //
230: // Comments: [Comments]
231: //
232: // Delay: 1000 units of time.
233: //
234: // Position: [Position]
235: //
236: // Comments: [Comments]
237: //
238: // Delay: 1000 units of time.
239: //
240: // Position: [Position]
241: //
242: // Comments: [Comments]
243: //
244: // Delay: 1000 units of time.
245: //
246: // Position: [Position]
247: //
248: // Comments: [Comments]
249: //
250: // Delay: 1000 units of time.
251: //
252: // Position: [Position]
253: //
254: // Comments: [Comments]
255: //
256: // Delay: 1000 units of time.
257: //
258: // Position: [Position]
259: //
260: // Comments: [Comments]
261: //
262: // Delay: 1000 units of time.
263: //
264: // Position: [Position]
265: //
266: // Comments: [Comments]
267: //
268: // Delay: 1000 units of time.
269: //
270: // Position: [Position]
271: //
272: // Comments: [Comments]
273: //
274: // Delay: 1000 units of time.
275: //
276: // Position: [Position]
277: //
278: // Comments: [Comments]
279: //
280: // Delay: 1000 units of time.
281: //
282: // Position: [Position]
283: //
284: // Comments: [Comments]
285: //
286: // Delay: 1000 units of time.
287: //
288: // Position: [Position]
289: //
290: // Comments: [Comments]
291: //
292: // Delay: 1000 units of time.
293: //
294: // Position: [Position]
295: //
296: // Comments: [Comments]
297: //
298: // Delay: 1000 units of time.
299: //
300: // Position: [Position]
301: //
302: // Comments: [Comments]
303: //
304: // Delay: 1000 units of time.
305: //
306: // Position: [Position]
307: //
308: // Comments: [Comments]
309: //
310: // Delay: 1000 units of time.
311: //
312: // Position: [Position]
313: //
314: // Comments: [Comments]
315: //
316: // Delay: 1000 units of time.
317: //
318: // Position: [Position]
319: //
320: // Comments: [Comments]
321: //
322: // Delay: 1000 units of time.
323: //
324: // Position: [Position]
325: //
326: // Comments: [Comments]
327: //
328: // Delay: 1000 units of time.
329: //
330: // Position: [Position]
331: //
332: // Comments: [Comments]
333: //
334: // Delay: 1000 units of time.
335: //
336: // Position: [Position]
337: //
338: // Comments: [Comments]
339: //
340: // Delay: 1000 units of time.
341: //
342: // Position: [Position]
343: //
344: // Comments: [Comments]
345: //
346: // Delay: 1000 units of time.
347: //
348: // Position: [Position]
349: //
350: // Comments: [Comments]
351: //
352: // Delay: 1000 units of time.
353: //
354: // Position: [Position]
355: //
356: // Comments: [Comments]
357: //
358: // Delay: 1000 units of time.
359: //
360: // Position: [Position]
361: //
362: // Comments: [Comments]
363: //
364: // Delay: 1000 units of time.
365: //
366: // Position: [Position]
367: //
368: // Comments: [Comments]
369: //
370: // Delay: 1000 units of time.
371: //
372: // Position: [Position]
373: //
374: // Comments: [Comments]
375: //
376: // Delay: 1000 units of time.
377: //
378: // Position: [Position]
379: //
380: // Comments: [Comments]
381: //
382: // Delay: 1000 units of time.
383: //
384: // Position: [Position]
385: //
386: // Comments: [Comments]
387: //
388: // Delay: 1000 units of time.
389: //
390: // Position: [Position]
391: //
392: // Comments: [Comments]
393: //
394: // Delay: 1000 units of time.
395: //
396: // Position: [Position]
397: //
398: // Comments: [Comments]
399: //
400: // Delay: 1000 units of time.
401: //
402: // Position: [Position]
403: //
404: // Comments: [Comments]
405: //
406: // Delay: 1000 units of time.
407: //
408: // Position: [Position]
409: //
410: // Comments: [Comments]
411: //
412: // Delay: 1000 units of time.
413: //
414: // Position: [Position]
415: //
416: // Comments: [Comments]
417: //
418: // Delay: 1000 units of time.
419: //
420: // Position: [Position]
421: //
422: // Comments: [Comments]
423: //
424: // Delay: 1000 units of time.
425: //
426: // Position: [Position]
427: //
428: // Comments: [Comments]
429: //
430: // Delay: 1000 units of time.
431: //
432: // Position: [Position]
```

## Logic:

- Implement functions to send commands and data to the LCD.
- Use various LCD commands such as clear display, cursor positioning, and line shifting. ■ Convert numerical values to ASCII before displaying them on the LCD.
- Make basic connections to power supply, ground, and enable pins.

## Common Mistakes:

- Command Mismatch: Ensure the correct commands are sent to the LCD. ■ Command Positioning: Correct positioning of commands is crucial for proper display. ■ Clear Display and New Line Commands: Use appropriate commands to clear the display or move to a new line when needed.



```
1 // Function prototypes
2 void lcd_init();
3 void lcd_clear();
4 void lcd_gotoxy(int x, int y);
5 void lcd_print(const char *str);
6 void lcd_println(const char *str);
7 void lcd_println(const char *str, int cols);
8 void lcd_println(const char *str, int cols, int line);
9
10 // LCD module functions
11
12 // Initialize the LCD (4-bit mode)
13 void lcd_init() {
14 pinMode(2, OUTPUT); // RS pin
15 pinMode(3, OUTPUT); // RW pin
16 pinMode(4, OUTPUT); // E pin
17
18 // LCD initialization
19 digitalWrite(2, HIGH); // RS pin
20 digitalWrite(3, LOW); // RW pin
21 digitalWrite(4, LOW); // E pin
22 delay(100); // Wait for the LCD to initialize
23 lcd_clear(); // Clear the LCD
24 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
25 lcd_print("LCD Module"); // Print the text "LCD Module"
26 lcd_println("v1.0"); // Print the version number
27 }
28
29 // Clear the LCD
30 void lcd_clear() {
31 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
32 lcd_print("\n"); // Print a new line character
33 }
34
35 // Move the cursor to the specified position (x, y)
36 void lcd_gotoxy(int x, int y) {
37 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
38 lcd_print("\n"); // Print a new line character
39 }
40
41 // Print the text "str" on the LCD
42 void lcd_print(const char *str) {
43 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
44 lcd_println(str); // Print the text "str"
45 }
46
47 // Print the text "str" on the LCD, followed by a new line
48 void lcd_println(const char *str) {
49 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
50 lcd_print(str); // Print the text "str"
51 lcd_println(); // Print a new line character
52 }
53
54 // Print the text "str" on the LCD, followed by a new line and a specified number of columns
55 void lcd_println(const char *str, int cols) {
56 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
57 lcd_print(str); // Print the text "str"
58 lcd_println(); // Print a new line character
59 }
60
61 // Print the text "str" on the LCD, followed by a new line and a specified number of columns and a specified number of lines
62 void lcd_println(const char *str, int cols, int line) {
63 lcd_gotoxy(0, 0); // Move cursor to the top-left corner
64 lcd_print(str); // Print the text "str"
65 lcd_println(); // Print a new line character
66 }
```

