**MPS Assignment 2:**

**Controlling LEDs and Reading Buttons with ATmega32 Microcontroller**

**Objective:**

To learn how to interface LEDs and buttons with the ATmega32 microcontroller using C programming.

**Questions:**

1. **Connecting LEDs**
   1. Describe how you would connect an LED to PORTB. What components are necessary?

**LED (Light Emitting Diode):** The main component.

**Resistor (typically 220Ω to 1kΩ):** This resistor limits the current flowing through the LED to prevent it from burning out.

**Wires:** For connecting the LED to the microcontroller and resistor.

**Power Supply:** For the LED to receive voltage (usually 5V from the ATmega32).

* 1. Write a C code snippet to turn on an LED connected to PORTB. Explain each line of code.

#include <avr/io.h>

int main(void) {

DDRB |= (1 << PB0); // Set PB0 as output

PORTB |= (1 << PB0); // Set PB0 high to turn on the LED

while (1) {

// The LED will remain on

}

return 0;

}

**DDRB |= (1 << PB0);**: This sets the data direction for pin PB0 to "output" (DDRB is the Data Direction Register for PORTB).

**PORTB |= (1 << PB0);**: This sets PB0 to high, providing power to the LED and turning it on.

1. **Button Interfacing**
   1. How would you connect a push button to PORTC? Provide a circuit diagram.

To connect a push button to PORTC:

* One terminal of the button is connected to the desired pin of PORTC (e.g., PC0).
* The other terminal is connected to ground (GND).
* To prevent floating values when the button is not pressed, you should use a **pull-up resistor** (internal or external).
  1. Write a code snippet to read the state of a button connected to PORTC and turn on an LED on PORTB when pressed.

#include <avr/io.h>

int main(void) {

DDRB |= (1 << PB0); // Set PB0 as output

DDRC &= ~(1 << PC0); // Set PC0 as input

PORTC |= (1 << PC0); // Enable pull-up resistor on PC0

while (1) {

if (!(PINC & (1 << PC0))) { // Check if button is pressed (low)

PORTB |= (1 << PB0); // Turn on LED

} else {

PORTB &= ~(1 << PB0); // Turn off LED

}

}

return 0;

}

1. **Debouncing Technique**
   1. Explain what button debouncing is and why it’s necessary.

Button debouncing is the process of eliminating the noise caused by the mechanical bounce when a button is pressed or released. When you press a button, its contacts may bounce between high and low states several times before settling. Without debouncing, this can cause multiple unintended transitions and incorrect readings.

Debouncing is necessary to ensure that the microcontroller reads a stable, clean signal from the button.

* 1. Write a C code snippet implementing a simple debouncing mechanism for a button connected to PORTC.

#include <avr/io.h>

#include <util/delay.h>

#define DEBOUNCE\_DELAY 50 // 50ms debounce delay

int main(void) {

DDRB |= (1 << PB0); // Set PB0 as output

DDRC &= ~(1 << PC0); // Set PC0 as input

PORTC |= (1 << PC0); // Enable pull-up resistor on PC0

uint8\_t button\_state = 0;

while (1) {

if (!(PINC & (1 << PC0))) { // Button pressed (low)

\_delay\_ms(DEBOUNCE\_DELAY); // Wait for debouncing

if (!(PINC & (1 << PC0))) { // Confirm button press

button\_state = 1;

}

} else {

button\_state = 0;

}

if (button\_state) {

PORTB |= (1 << PB0); // Turn on LED

} else {

PORTB &= ~(1 << PB0); // Turn off LED

}

}

return 0;

}

1. **Multiple LEDs Control**
   1. Describe how to connect three LEDs to PORTD. What will the circuit look like?

To connect three LEDs to PORTD:

* Connect each LED in series with a current-limiting resistor (220Ω to 1kΩ).
* Connect the anode (+) of each LED to a pin of PORTD (e.g., PD0, PD1, and PD2).
* Connect the cathodes (-) of the LEDs to ground.
  1. Write a program that turns on LEDs on PORTD based on the binary value read from a set of buttons on PORTA.

#include <avr/io.h>

int main(void) {

DDRD = 0x07; // Set PD0, PD1, PD2 as output for LEDs

DDRA = 0x00; // Set PORTA as input for buttons

while (1) {

uint8\_t buttons = PINA & 0x07; // Read the state of the first 3 buttons (PA0-PA2)

PORTD = buttons; // Turn on corresponding LEDs on PORTD

}

return 0;

}

1. **Using Switch-Case**

How can a switch-case statement be used to control multiple LEDs based on button presses? Provide a code example.

#include <avr/io.h>

int main(void) {

DDRB = 0xFF; // Set PORTB as output (for LEDs)

DDRA = 0x00; // Set PORTA as input (for buttons)

while (1) {

uint8\_t buttons = PINA & 0x07; // Read first 3 buttons

switch (buttons) {

case 0x01: // Button 1 pressed

PORTB = 0x01; // Turn on LED 1

break;

case 0x02: // Button 2 pressed

PORTB = 0x02; // Turn on LED 2

break;

case 0x03: // Button 1 and 2 pressed

PORTB = 0x03; // Turn on both LED 1 and 2

break;

default:

PORTB = 0x00; // Turn off all LEDs

}

}

return 0;

}

1. **LED Blinking**

Write a program that blinks an LED on PORTB at a 1-second interval using a loop and delay function.

#include <avr/io.h>

#include <util/delay.h>

int main(void) {

DDRB |= (1 << PB0); // Set PB0 as output

while (1) {

PORTB |= (1 << PB0); // Turn on LED

\_delay\_ms(1000); // 1 second delay

PORTB &= ~(1 << PB0); // Turn off LED

\_delay\_ms(1000); // 1 second delay

}

return 0;

}

1. **Reading Multiple Buttons**

Describe how to read the states of multiple buttons connected to PORTA and turn on corresponding LEDs on PORTB. Write the code for this.

#include <avr/io.h>

int main(void) {

DDRB = 0xFF; // Set PORTB as output (for LEDs)

DDRA = 0x00; // Set PORTA as input (for buttons)

while (1) {

uint8\_t buttons = PINA; // Read all button states

PORTB = buttons; // Turn on corresponding LEDs on PORTB

}

return 0;

}

1. **LED Patterns**

Write a program that creates a running LED pattern on PORTB (e.g., shifting lights). Describe the logic behind your implementation.

#include <avr/io.h>

#include <util/delay.h>

int main(void) {

DDRB = 0xFF; // Set PORTB as output

while (1) {

for (uint8\_t i = 0; i < 8; i++) {

PORTB = (1 << i); // Shift LED on PORTB

\_delay\_ms(250); // Delay

}

}

return 0;

}

1. **Using Functions**

Write a function that initializes PORTA for button input and another function that initializes PORTB for LED output. Provide the code.

#include <avr/io.h>

void initButtons() {

DDRA = 0x00; // Set PORTA as input (for buttons)

}

void initLEDs() {

DDRB = 0xFF; // Set PORTB as output (for LEDs)

}

int main(void) {

initButtons();

initLEDs();

while (1) {

uint8\_t buttons = PINA;

PORTB = buttons; // Turn on corresponding LEDs

}

return 0;

}

1. **Using External Libraries**

Discuss how external libraries can simplify the process of controlling LEDs and buttons. Provide an example of a library you might use.

#include <avr/io.h>

#include <util/delay.h>

int main(void) {

DDRB = 0xFF; // Set PORTB as output

while (1) {

PORTB |= (1 << PB0); // Turn on LED

\_delay\_ms(1000); // Delay

PORTB &= ~(1 << PB0); // Turn off LED

\_delay\_ms(1000); // Delay

}

return 0;

}

**Submission:**

* Provide detailed answers to all questions.
* Include code snippets for programming questions.
* Draw circuit diagrams where applicable.

**Deadline:**

* 29/09/2024