In the name of Allah

Microcontrollers - Final Project

Server Temperature Control System with Atmega32

Master Code (Master Atmega32)

Detailed Breakdown:

1. Initialization:
    - Ports: Configures data direction registers (DDRx) for GPIO pins used for LCD, LEDs, keypad, and SPI communication.
    - SPI: Initializes SPI communication in master mode (SPI_Init() function).
    - LCD: Initializes the LCD display (init_LCD() function) for user interface.
2. Password Management:
    - Keypad Input: Utilizes keypad_getkey() function to receive numeric input from a 4x3 matrix keypad.
    - Password Entry: Users enter a 5-digit password which is displayed on the LCD.
    - SPI Transmission: Sends the entered password byte-by-byte to the slave via SPI using SPI_Transmit().
3. Access Control:
    - Password Verification: Receives a status response ('1' for correct password, '0' for incorrect) from the slave via SPI using SPI_Receive().
    - Access Granted: If the password is correct, grants access to the system:
        - Displays access granted message on the LCD.
        - Allows the user to select between displaying motor status or temperature readings.
4. Main Loop:
    - Mode Selection: Allows the user to choose between motor status display or temperature display (select_mode() function).
    - Temperature Reading: Receives temperature data from the slave via SPI.
    - Motor Control: Calculates and displays motor duty cycles based on the received temperature using call_of_duty() function.
    - System Status: Checks the overall system status based on the calculated duty cycles and updates LED indicators (check_system_status() function).
5. Error Handling:
    - Access Denied: If the password is incorrect:

        - Displays access denied message on the LCD.
        - After three consecutive failed attempts, locks the system for 30 seconds (wait_for_pass_faults() function).

Functions Used:

- LCD Functions: LCD_init_display(), LCD_String() for initializing and displaying messages on the LCD.
- SPI Functions: SPI_Init(), SPI_Transmit(), SPI_Receive() for initializing SPI and transmitting/receiving data.
- Keypad Function: keypad_getkey() for capturing user input from the keypad.
- Temperature to Duty Cycle Conversion: call_of_duty() for converting temperature to motor duty cycle percentage.
- System Status Function: check_system_status() for managing LED indicators based on system conditions.

Slave Code (Slave Atmega32)

Detailed Breakdown:

1. Initialization:
    - Ports: Configures data direction registers (DDRx) for GPIO pins used for SPI communication, ADC, and PWM.
    - SPI: Initializes SPI communication in slave mode (SPI_init() function).
    - ADC: Initializes the ADC (init_ADC() function) for reading temperature from an LM35 sensor.
    - PWM: Initializes PWM for motor control (init_TimerCounter() function).
2. Password Reception:
    - SPI Reception: Receives the password from the master via SPI using SPI_Receive().
3. Password Verification:
    - Password Check: Compares the received password with a predefined correct password (correct_password[]).
    - SPI Transmission: Sends a status response ('1' for correct password, '0' for incorrect) back to the master via SPI using SPI_Transmit().
4. Temperature Sensing:
    - ADC Read: Continuously reads temperature from an LM35 sensor using ADC (get_temperature() function).
    - Temperature Transmission: Sends the temperature data back to the master via SPI using SPI_Transmit().

5. Motor Control:
    - PWM Duty Cycle Calculation: Adjusts PWM duty cycles (set_motor_duty_cycle() function) based on the received temperature.
    - Motor Output: Controls motor operations based on the calculated duty cycles.

Functions Used:

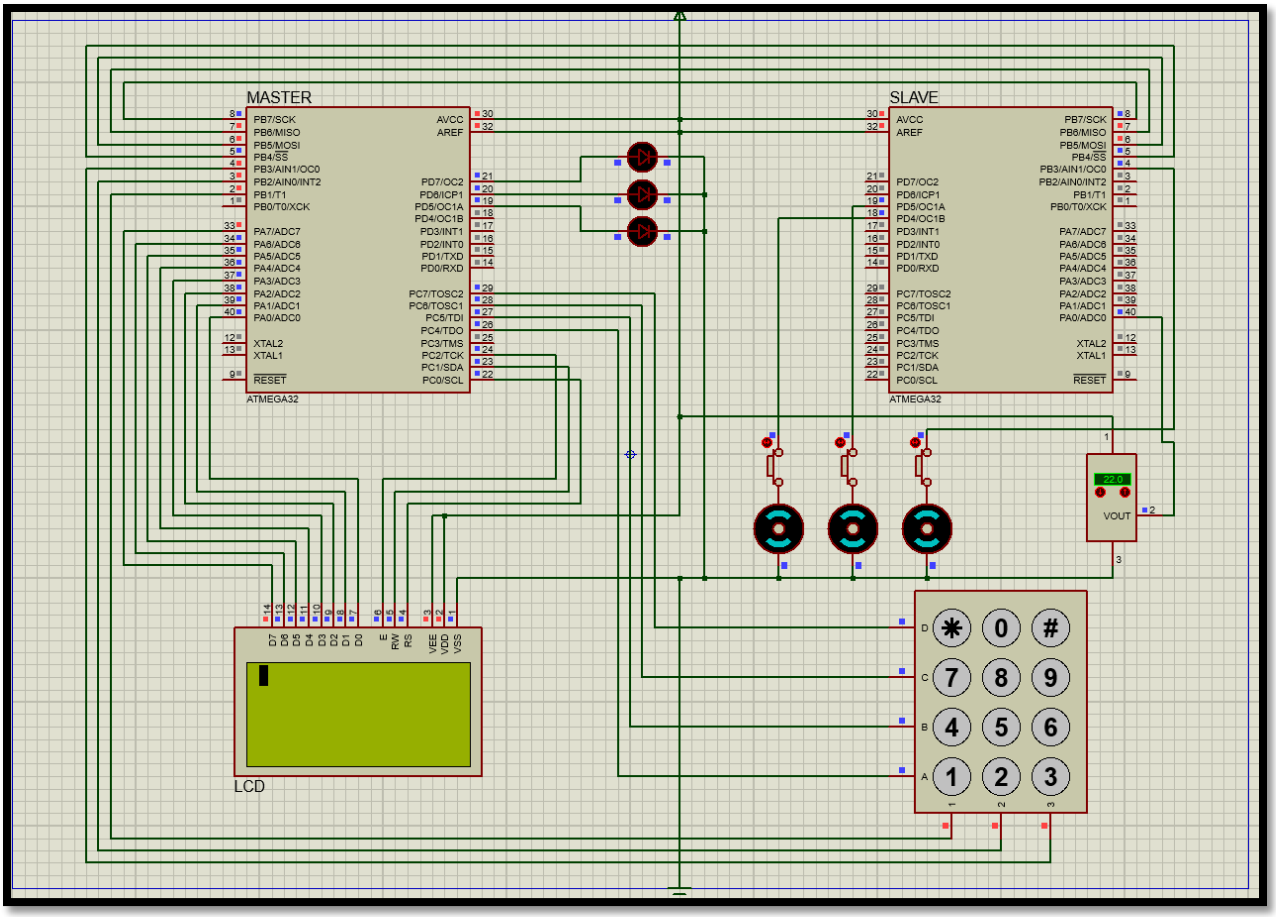
- SPI Functions: SPI_init(), SPI_Transmit(), SPI_Receive() for initializing SPI and transmitting/receiving data.
- ADC Functions: init_ADC(), get_temperature() for initializing ADC and reading temperature from LM35 sensor.
- PWM Function: init_TimerCounter(), set_motor_duty_cycle() for initializing PWM and setting duty cycles for motor control.
- Password Verification Function: check_password() for comparing received password with correct password and sending status response.

Interaction Between Master and Slave:

- SPI Communication: Master initiates communication by sending the password. Slave receives and verifies the password.
- Data Exchange: After verification, slave continuously sends temperature data to the master. Master adjusts motor operations based on received data.
- Control Flow: Master manages user interface (LCD), user input (keypad), and system status (LEDs). Slave handles sensor reading (ADC), motor control (PWM), and responds to master commands.

Conclusion:

The master-slave system implemented on Atmega32 microcontrollers demonstrates efficient communication via SPI for real-time temperature monitoring and motor control. Each part (master and slave) plays a crucial role in ensuring the overall functionality of the Server Temperature Control System, showcasing integration of input/output operations, sensor interfacing, and control logic for industrial applications.



Good luck! ☺

Seyed Mahdi Mahdavi Mortazavi

Student number: 40030490

Summer 2024