EE 337: MICROPROCESSOR LAB

Lab No. 09 - B

Yaswanth Ram Kumar
23B1277

Dept. of Electrical Engineering
Indian Institute of Technology Bombay

Aim

In this part of the project, we have implemented board-to-board communication using the SPI protocol. The Master board first sends two prime numbers to the Slave board via SPI. Upon receiving the numbers, the Slave performs their addition and displays the sum on its LCD. It then checks whether the resulting sum is a prime number. Based on this check, the Slave sends a response back to the Master, which then displays either "PRIME" or "NOT PRIME" on its own LCD.

SPCON Register Configuration

From the AT89C5131A datasheet `https://ww1.microchip.com/downloads/en/DeviceDoc/doc4337.pdf`, the SPCON register is configured with the hexadecimal value `0x7F` for Master and `0x6F` for Slave, based on the following bit settings:

Bit	Name	Description	Master Value	Slave Value
7	SPR2	Reserved (must be 0)	0	0
6	SPEN	1 = Enable SPI	1	1
5	SSDIS	1 = Disable Slave Select (SSBAR)	1	1
4	MSTR	1 = Set MCU as SPI Master	1	0
3	CPOL	1 = SCK idle high	1	1
2	CPHA	1 = Data sampled on the second edge	1	1
1–0	SPR1:0	11 = SPI Clock = $F_{clk}/16$	11	11

Master Configuration:
Binary Representation: `0b01111111`
Hexadecimal Value: `0x7F`

Slave Configuration:
Binary Representation: `0b01101111`
Hexadecimal Value: `0x6F`

Code Files and SPI Register Explanation

The SPI communication setup in this project was implemented from scratch based on the AT89C5131 datasheet. Two separate files were created: `master_SPI.c` and `slave_SPI.c`, which contain the full implementation for the SPI Master and SPI Slave respectively. Instead of using the provided `SPI.h` file, we directly accessed and configured the necessary Special Function Registers (SFRs) for SPI, allowing for better understanding and finer control over the protocol.

SPDAT (SPI Data Register): This is the data register used for both transmitting and receiving a byte via SPI. Writing to `SPDAT` initiates the transmission, and once the transfer is complete, the received data from the slave (or master, depending on the mode) is available in the same register.

SPSTA (SPI Status Register): This register contains status flags related to SPI transmission. The most important bit used in our code is:

- `SPIF (SPI Interrupt Flag) { Bit 7:` This bit is automatically set by hardware when a data transfer is complete, indicating that the `SPDAT` register now contains valid received data. Our code continuously checks this bit (using `SPSTA & 0x80`) to determine when it's safe to read from `SPDAT`.

Overall, this approach ensures tight control of SPI communication and provides valuable experience in low-level embedded system programming.

Master Code Explanation

```
// LCD is initialized and a greeting message with animated dots is shown
lcd_init();
master_greet_lcd();  // Displays "SPI Master" and "Running..." with animation

// SPI initialized in Master mode (CPOL=1, CPHA=1), Fclk/16
SPCON = 0x7F;
msdelay(500);

// Clear LCD, notify that primes are being sent
lcd_cmd(0x01);
lcd_cmd(0x80);
lcd_write_string("Sending Primes");

// Two primes (2 and 3) are sent with 1s delay
spi_send(prime1);
msdelay(1000);
spi_send(prime2);
msdelay(1000);

// Display waiting message and delay to ensure slave has processed
lcd_cmd(0xC0);
lcd_write_string("Waiting Result");
msdelay(1500);

// Send dummy byte to receive result from slave
response = spi_send(0x00);

// Interpret response: 1 = Prime, 2 = Not Prime
lcd_cmd(0x01);
lcd_cmd(0x80);
if (response == 1) {
    lcd_write_string("Sum is Prime");
} else if (response == 2) {
    lcd_write_string("Sum Not Prime");
} else {
    lcd_write_string("Unexpected Val");
    lcd_cmd(0xC0);
    lcd_write_char(response+'0');
}
```

Listing 1: SPI Master Code

Slave Code Explanation

```
// LCD initialized, and "SPI Slave" + "Waiting..." animation is displayed
lcd_init();
slave_greet_lcd();

// SPI initialized in Slave mode (CPOL=1, CPHA=1)
SPCON = 0x6F;
msdelay(500);

// Show waiting for incoming data
lcd_cmd(0x01);
lcd_cmd(0x80);
lcd_write_string("Waiting Data");

while (1) {
    // Wait for SPI data
    while (!(SPSTA & 0x80));
    received = SPDAT;

    // First byte received is prime1, store and show it
    if (!received_first) {
        prime1 = received;
        received_first = 1;

        lcd_cmd(0x01);
        lcd_cmd(0x80);
        lcd_write_string("Got Prime1:");
        lcd_cmd(0xC0);
        int_to_string(prime1, buffer);
        lcd_write_string(buffer);
    }
    // Second byte received is prime2, then compute and send result
    else {
        prime2 = received;
        lcd_cmd(0x01);
        lcd_cmd(0x80);
        lcd_write_string("Got Prime2:");
        lcd_cmd(0xC0);
        int_to_string(prime2, buffer);
        lcd_write_string(buffer);

        msdelay(1000);

        sum = prime1 + prime2;
        lcd_cmd(0x01);
        lcd_cmd(0x80);
        lcd_write_string("Sum:");
        int_to_string(sum, buffer);
        lcd_write_string(buffer);

        msdelay(1000);
```

Listing 2: SPI Slave Code

```
// Slave Code Continued...
        if (is_prime(sum)) {
            SPDAT = 1;  // Send 1 if prime
            lcd_cmd(0xC0);
            lcd_write_string("Sum is Prime");
            while(1);  // Halt
        } else {
            SPDAT = 2;  // Send 2 if not prime
            lcd_cmd(0xC0);
            lcd_write_string("Not Prime");
            while(1);  // Halt
        }

        msdelay(1000);  // Wait before resetting state
        received_first = 0;
        lcd_cmd(0x01);
        lcd_cmd(0x80);
        lcd_write_string("Waiting Data");
    }
}
```

Listing 3: SPI Slave Code Continuation

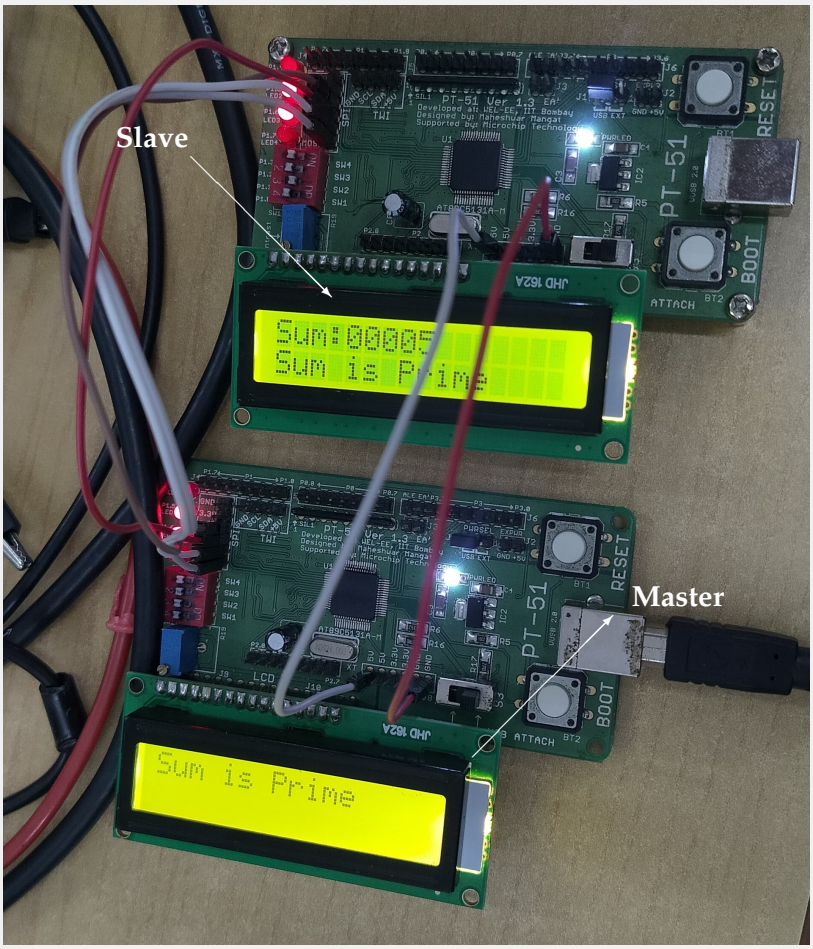

Figure 1: Outputs on the LCD Screens when 2 and 3 are given as inputs

Demonstration and Verification

Everything has been successfully implemented and tested on the hardware. The SPI communication between the master and slave microcontrollers was verified using the LCD outputs at both ends. The final versions of both `master_SPI.c` and `slave_SPI.c` were uploaded and executed as intended. The system was able to send two numbers from the master to the slave, compute the sum on the slave side, determine whether the sum is a prime number, and send back the appropriate response. This entire process was demonstrated live and verified in the presence of the teaching assistants (TAs).

Thank You

23B1277