EE690 Embedded Systems

10 October 2024

Lab 7 Report

Group 9

Ashis Pradhan 210020003

Gurvir Singh 210020012

Aim: Learn to use the UART

Problem Statement:

1. Program your micro-controller to transmit the 8-bit value "0xF0" if SW1 is pressed and "0xAA" if SW2 is pressed over UART with baud rate 9600 and odd parity. Read the relevant sections of the datasheet and board manual.

2. Sketch the expected waveforms for both cases with indicative timings.

3. Connect the Scope to the TX pin and verify that the captured signals match what you have drawn in part2.

4. Add code to your program to also listen for incoming data on the UART with the same baud and parity config. If "0xAA" is received, turn LED should light up GREEN. If "0xF0" is received, the LED should be BLUE and if any error is detected LED should be RED. Test this by communicating with your neighbouring group. Remember to connect RX of one board to TX of the other, and make sure to connect the board grounds together.

Theory:

First we use IO interrupt to check if the button is pressed or not. Then we enable clock for transmission using UART-1 on port 8. Next we configure PB0 and PB1 for UART transmission and set the baud rare 9600 along with odd parity as the configuration for UART-1. Finally we use the button interrupts to send singles on each button press, and hence the required output is transmitted.

For reception, we configure the UART in same way as transmission, and we listen on specified ports always, so whenever a signal is discovered and each character is read and processed, after which we turn on the specified lights according to the inout received.

Code for transmitter:

```c
int main(void) {
    // Enable clock for Port F
    SYSCTL_RCGCGPIO_R |= (1 << 5);
    while ((SYSCTL_PRGPIO_R & (1 << 5)) == 0);

    // Unlock PF0 and PF4 (Switches)
    GPIO_PORTF_LOCK_R = 0x4C4F434B;  // Unlock GPIO Port F
    GPIO_PORTF_CR_R |= 0x11;         // Allow changes to PF0 and PF4

    // Configure PF0 and PF4 (Switches) as input and PF1 (LED) as output
    GPIO_PORTF_DIR_R &= ~(0x11);     // Set PF0 and PF4 as input (switches)
    GPIO_PORTF_DIR_R |= (0x02);      // Set PF1 as output (LED)
    GPIO_PORTF_DEN_R |= (0x13);      // Enable digital functionality for PF0, PF4, and PF1
    GPIO_PORTF_PUR_R |= (0x11);      // Enable pull-up resistors on PF0 and PF4

    // Enable clock for UART1 and Port B
    SYSCTL_RCGCUART_R |= (1 << 1);   // Enable UART1 clock
    SYSCTL_RCGCGPIO_R |= (1 << 1);   // Enable GPIO Port B clock
    while ((SYSCTL_PRGPIO_R & (1 << 1)) == 0);

    // Configure PB0 and PB1 for UART1
    GPIO_PORTB_AFSEL_R |= (1 << 0) | (1 << 1);  // Enable alternate function on PB0 and PB1
    GPIO_PORTB_PCTL_R |= (1 << 0) | (1 << 4);   // Set UART functionality on PB0 (U1Rx) and PB1 (U1Tx)
    GPIO_PORTB_DEN_R |= (1 << 0) | (1 << 1);    // Enable digital functionality on PB0 and PB1

    // Configure UART1
    UART1_CTL_R &= ~(0x01);            // Disable UART1
    UART1_IBRD_R = 104;                // Integer portion of BRD (for 9600 baud rate at 16 MHz clock)
    UART1_FBRD_R = 11;                 // Fractional portion of BRD
    UART1_LCRH_R = (0x3 << 5);         // 8-bit, no parity, 1 stop bit
    UART1_CC_R = 0x0;                  // Use system clock for UART
    UART1_CTL_R |= (0x01 | 0x200);     // Enable UART1 and Tx
```

```c

    // Configure interrupt for PF0 and PF4
    GPIO_PORTF_IM_R &= ~(0x11);        // Disable interrupts for PF0 and PF4
    GPIO_PORTF_IS_R &= ~(0x11);        // Make PF0 and PF4 edge-sensitive
    GPIO_PORTF_IBE_R &= ~(0x11);       // Not both edges
    GPIO_PORTF_IEV_R &= ~(0x11);       // Falling edge triggers
    GPIO_PORTF_ICR_R |= 0x11;          // Clear any prior interrupts
    GPIO_PORTF_IM_R |= 0x11;           // Enable interrupts on PF0 and PF4

    // Enable GPIO Port F interrupt in NVIC (IRQ30 for Port F)
    NVIC_EN0_R |= (1 << 30);

    // Enable global interrupts
    __asm("     CPSIE I");

    while (1) {
        // Main loop
    }
}
```

Receiver code:

```
5    void UART1_Init(void);
6    char UART1_ReadChar(void);
7    void LED_Init(void);
8    void LED_Control(char receivedData);
9
10   int main(void) {
11       // Initialize UART1 and LED pins
12       UART1_Init();
13       LED_Init();
14
15       while (1) {
16           char received = UART1_ReadChar();  // Read character from UART1
17           LED_Control(received);             // Control LED based on received data
18       }
19   }
```

```
21   void UART1_Init(void) {
22       // Enable clock for UART1 and Port B
23       SYSCTL_RCGCUART_R |= (1 << 1);   // Enable UART1 clock
24       SYSCTL_RCGCGPIO_R |= (1 << 1);   // Enable GPIO Port B clock
25       while ((SYSCTL_PRGPIO_R & (1 << 1)) == 0);
26
27       // Configure PB0 and PB1 for UART1
28       GPIO_PORTB_AFSEL_R |= (1 << 0) | (1 << 1);  // Enable alternate function on PB0 and PB1
29       GPIO_PORTB_PCTL_R |= (1 << 0) | (1 << 4);   // Set UART functionality on PB0 (U1Rx) and PB1 (U1Tx)
30       GPIO_PORTB_DEN_R |= (1 << 0) | (1 << 1);    // Enable digital functionality on PB0 and PB1
31
32       // Configure UART1
33       UART1_CTL_R &= ~(0x01);           // Disable UART1
34       UART1_IBRD_R = 104;               // Integer portion of BRD (for 9600 baud rate at 16 MHz clock)
35       UART1_FBRD_R = 11;                // Fractional portion of BRD
36       UART1_LCRH_R = (0x3 << 5);        // 8-bit, no parity, 1 stop bit
37       UART1_CC_R = 0x0;                 // Use system clock for UART
38       UART1_CTL_R |= (0x01 | 0x200);    // Enable UART1 and Rx
39   }
40
41   char UART1_ReadChar(void) {
42       while ((UART1_FR_R & 0x10) != 0);  // Wait until the Rx buffer is not empty
43       return (char)(UART1_DR_R & 0xFF);  // Read the received character
44   }
```

a

```
46  void LED_Init(void) {
47      // Enable clock for Port F
48      SYSCTL_RCGCGPIO_R |= (1 << 5);
49      while ((SYSCTL_PRGPIO_R & (1 << 5)) == 0);
50
51      // Configure PF1, PF2, and PF3 as output (Red, Blue, Green LEDs)
52      GPIO_PORTF_DIR_R |= 0x0E;          // Set PF1, PF2, PF3 as output
53      GPIO_PORTF_DEN_R |= 0x0E;          // Enable digital functionality on PF1, PF2, PF3
54  }
55
56  void LED_Control(char receivedData) {
57      // Turn off all LEDs initially
58      GPIO_PORTF_DATA_R &= ~(0x0E);
59
60      // Control LEDs based on received data
61      if (receivedData == 'a') {                // "aa" received as two 'a' characters
62          GPIO_PORTF_DATA_R |= (1 << 3);        // Turn on green LED (PF3)
63      } else if (receivedData == 0xF0) {        // "f0" received as a byte
64          GPIO_PORTF_DATA_R |= (1 << 2);        // Turn on blue LED (PF2)
65      } else {
66          GPIO_PORTF_DATA_R |= (1 << 1);        // Turn on red LED (PF1) for error
67      }
68  }
```

Output:

We observe that the UART-1 configuration follows the 9600 Baud rate, and is useful for Serial transmission of data over specified ports, and can also be used to readily listen for data and process characters to communicate desired data values over serial transmission using desired ports.