Laboratory 7: UART Receiving and Transmitting

Experiment Sheet

Purpose

The purpose of this laboratory is to understand the configuration and usage of UART (Universal Asynchronous Receiver/Transmitter) on the TM4C123GH6PM microcontroller. This experiment involves setting up UART to transmit and receive data, allowing the microcontroller to interact with external devices or a terminal emulator. Students will learn how to:

1. Configure UART for communication.
2. Transmit characters or strings.
3. Receive input from a terminal or other UART-enabled devices.

Essential Knowledge

UART Overview

UART is a serial communication protocol for asynchronous data transmission, typically using two lines:

- **TX (Transmit)**: Sends data.
- **RX (Receive)**: Receives data.

Key Registers and Their Purposes

1. **SYSCTL_RCGCUART**: Enables the clock for the UART module.
2. **SYSCTL_RCGCGPIO**: Enables the clock for GPIO ports associated with UART pins.
3. **GPIO_AFSEL**: Configures pins for alternate functions (e.g., TX/RX).
4. **GPIO_PCTL**: Sets the UART function for selected pins.
5. **UART_CTL**: Controls the UART module (enable/disable).
6. **UART_IBRD and UART_FBRD**: Set the baud rate (integer and fractional parts).
7. **UART_LCRH**: Configures the data frame (e.g., word length, parity, stop bits).
8. **UART_FR**: Flag register for monitoring TX and RX buffer status.
    - **TXFF (Transmit FIFO Full)**: Indicates the transmit buffer is full.
    - **RXFE (Receive FIFO Empty)**: Indicates the receive buffer is empty.
9. **UART_DR**: Data register for transmitting and receiving data.

Steps for UART Configuration

Step 1: Enable Clocks for UART and GPIO

1. Enable the clock for the UART module using the `SYSCTL_RCGCUART` register.
2. Enable the clock for the GPIO port associated with TX and RX pins using the `SYSCTL_RCGCGPIO` register.

Step 2: Configure GPIO Pins for UART

1. Set the TX and RX pins as alternate functions using `GPIO_AFSEL`.
2. Configure the pins for UART operation using `GPIO_PCTL`.
3. Set the TX pin as output and the RX pin as input using `GPIO_DIR`.

Step 3: Configure the UART Module

1. **Disable UART**: Clear the UART enable bit in `UART_CTL` before making changes.
2. **Set Baud Rate**: Use the following example for a 16 MHz clock and 9600 baud rate:

```
UART_IBRD = 104;  // Integer part
UART_FBRD = 11;   // Fractional part
```

3. **Configure Frame Settings**:
   - 8-bit data, no parity, 1 stop bit using the `UART_LCRH` register.
4. **Enable UART**: Set the UART enable bit in `UART_CTL`.

Step 4: Transmit and Receive Data

1. **Transmit Data**:
   - Check the `TXFF` flag in `UART_FR` to ensure the transmit FIFO is not full.
   - Write data to `UART_DR` to send it.
2. **Receive Data**:
   - Check the `RXFE` flag in `UART_FR` to ensure data is available in the receive FIFO.
   - Read data from `UART_DR` to process the received byte.

Tasks For Students

Part A: Display Welcome Message

- Write a UART string transmission routine to send the following message upon startup:

```
markdown
Kodu kopyala
**************************************************
*                                                *
*    ** Welcome to the TM4C123 UART Control! **    *
*                                                *
**************************************************

** Available Commands:
   -> Type 'R' - Turn ON the RED LED.
   -> Type 'G' - Turn ON the GREEN LED.
   -> Type 'B' - Turn ON the BLUE LED.
   -> Type 'S' - Turn OFF ALL LEDs.

-> Press Enter to submit your command.
```

```
Let the LED magic begin! Start typing below:
```

Part B: Command-Based LED Control

- Implement a function that reads a character from UART and processes it:
  - Turn on or off LEDs based on the commands above.
  - If an invalid character is received, do nothing.

Part C: Echo Received Commands

- Echo each received command to the terminal for user feedback.

Part D: Switch Feedback (Optional)

- Detect SW1 and SW2 presses and transmit appropriate messages via UART:
  - SW1: Send `"SW1 pressed"`.
  - SW2: Send `"SW2 pressed"`.