Experiment Design

By Tariq Abdullah Alshaya.

UART communication

This report will outline how **UART** communication works and demonstrate different registers that are related to it.

In order to power up **UART**, **PCONP** is used refer to **Table 46**, **UART0** and **UART1** or **ON** by default. Whereas **UART2** and **UART3** are **OFF** and must be turned **ON** before using them, **UART2** have bit number **24** where **UART3** have bit number **25** in **PCONP**.
LPC1769 have four different **UART** registers, **UART0**, **UART1**, **UART2** and **UART3**. Having pins named **RXDn** and **TXDn**, where **n** ranges from **0** to **3** so that **UART0** have **RXD0** and **TXD0** and the same goes for the rest.
To configure pins' functions, use **PINSELn** to be configured as **UART** pins by selecting **RXDn** and **TXDn** refer to **Table 80**.

UART have the following registers **Table 271**:

- **RBR**: Data recently received.
- **THR**: Data to be transmitted.
- **FCR**: **FIFO** control register.
- **LCR**: Line Control Register that controls frame formatting.
- **DLL**: Baud rate generator for least significant byte.
- **DLM**: Baud rate generator for most significant byte.

In the **FCR** register the following can be done **Table 279**:

- **FIFO** enabled for **RX** and **TX** (1<<0).
- Clear **RX FIFO** and reset the pointer (1<<1).
- Clear **TX FIFO** and reset the pointer (1<<2).
- Enable **DMA** mode (1<<3).

In the **LCR** register the following can be done **Table 280**:

- Bit 1:0 is to determine the word length ranging from 5-8 bit character length.
- Bit 2 is to determine the Stop bit either 1 or 2 stop bit.
- To enable parity bit (1<<3).
- Bit 5:4 for Parity Selection.
- Bit 6 Break control.
- To enable **DLAB** (1<<7).

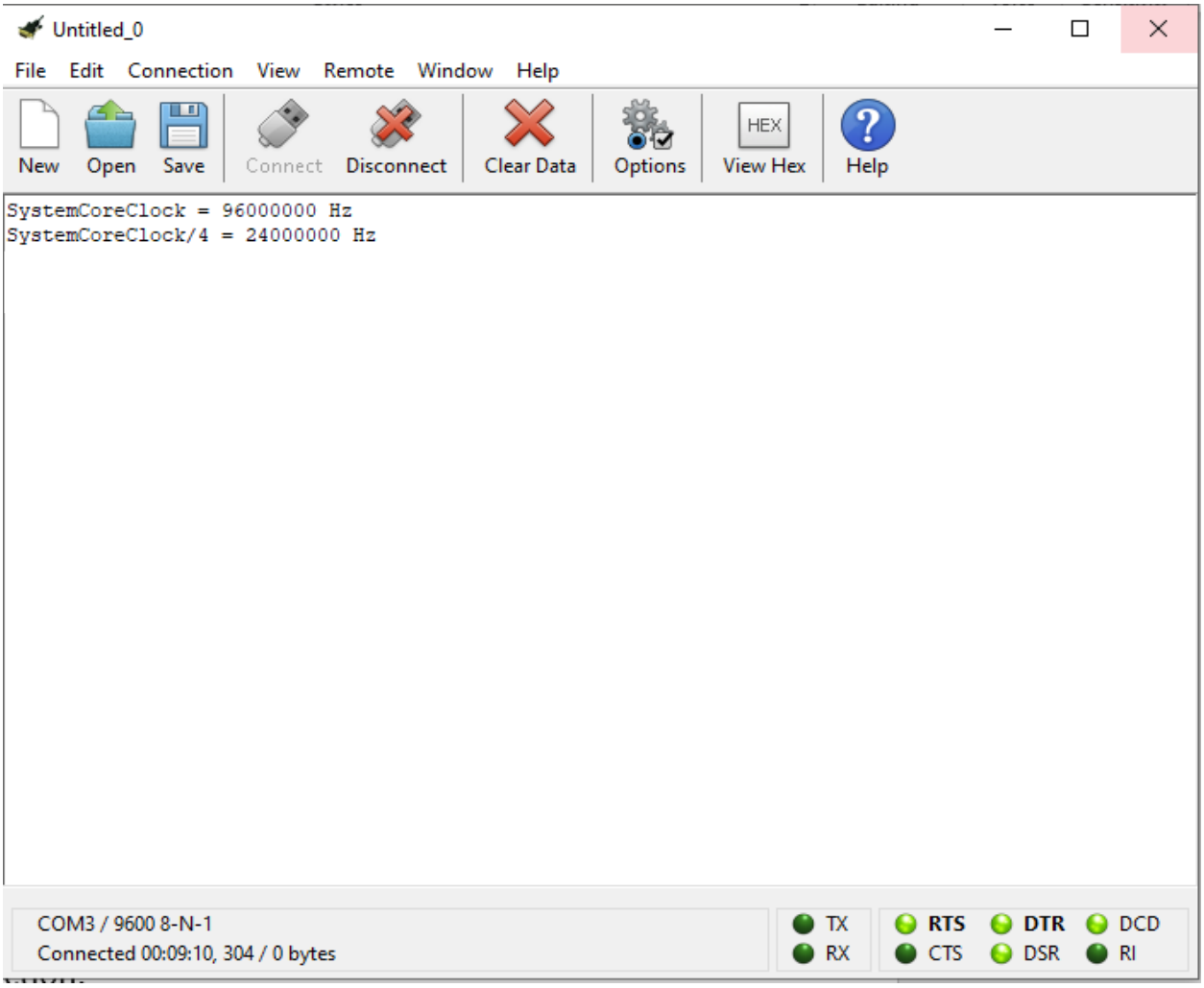
Baud rate calculation = $\frac{PCLK}{16 \times Baudrate}$

In **Table 40**,
PCLK_UART0 is at Bit 7:6 in **PCLKSEL0**.
In **Table 42**,
00 will result in **PCLK** = $\frac{CCLK}{4} = \frac{96MHz}{4} = 24MHz$

In order to find **CCLK**, the following code was implemented and using **CoolTerm** to detect the output.

```
#include "mbed.h"
int main() {
    int x = SystemCoreClock;
    printf("SystemCoreClock = %d Hz\n", x);
    x = SystemCoreClock/4;
    printf("SystemCoreClock/4 = %d Hz\n", x);
}
```

The output:



To configure **UART0**:

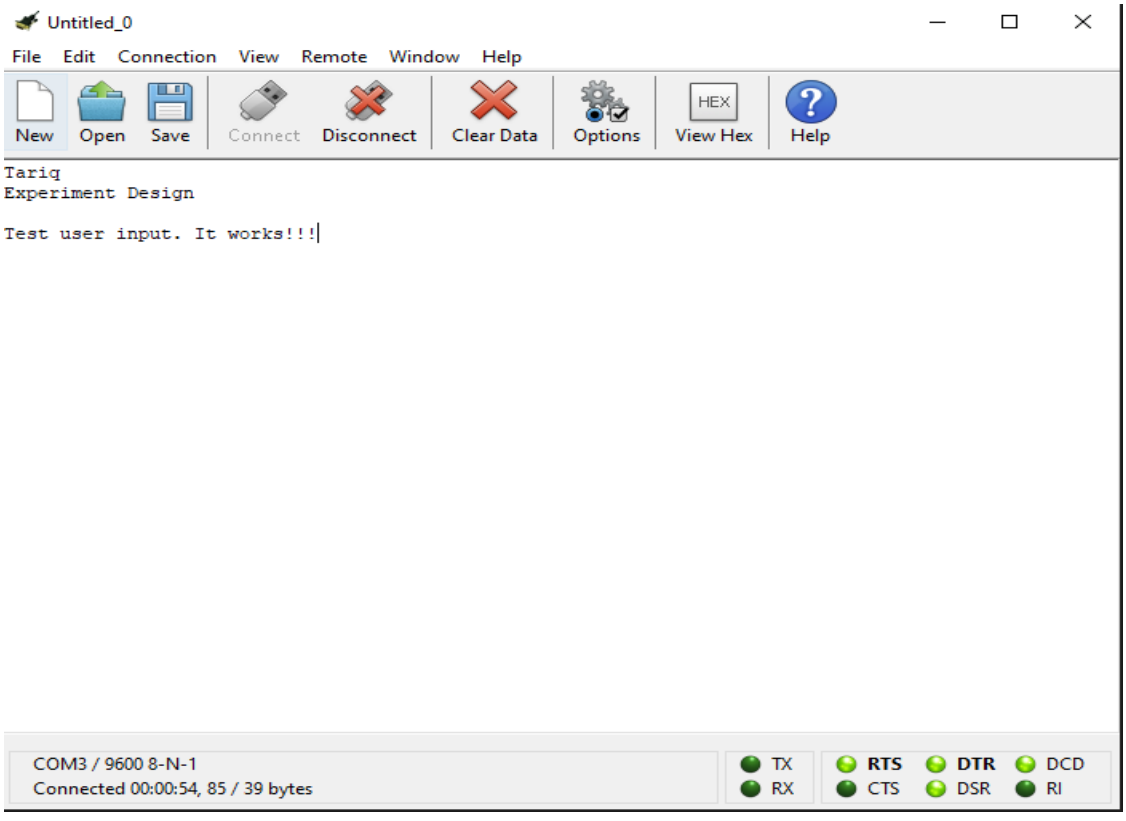
- Use **PINSELn** to configure the **GPIO** pin function.
- Configure **FCR** register by enabling **FIFO** and resetting **RX** and **TX**.
- Configure **LCR** register by determining the word length, Stop bit and enable the access to Divisor Latches.
- Determine **PCLK** from **Table 42,** in this case **00** to divide **CCLK** by **4**.
- Calculate the Baud Rate calculations. The Baud Rate, data bits, parity bit and Stop bit must be in sync with the software that is used to read the output, i.e. **CoolTerm** or **PuTTy**.
- Update **DLL** and **DLM** with their values from the Baud Rate calculations.
- Disable the access to Divisor Latches.

In the coming code **UART0** was implemented and configured to print an array of type 'char' and to print single characters one by one. Also, the code allows the user to write an input using the keyboard which will be read in the 'while' loop using 'ReceiveRx' function. Further explanation is in the code using comments.

```
#include "mbed.h"


void UARTInit(){

    LPC_PINCON -> PINSEL0 |= (1 << 4) | (1 << 6); // Enable TXD0 and RXD0.

    LPC_UART0 -> FCR = (1<<0) | (1<<1) | (1<<2); // Enable FIFO, reset RX
and TX buffers.
    LPC_UART0 -> LCR = (1<<0) | (1<<1) | (1<<7); // 8bit char length, 1 Stop
bit, Enable access to Divisor Latches.

    int BaudRateCalc = ( 24000000 / (16 * 9600 )); // 24MHz.
    LPC_UART0 -> DLL =  BaudRateCalc & 0xFF; // Take only first 8bits.
    LPC_UART0 -> DLM = (BaudRateCalc >> 0x08) & 0xFF; // Delete lower 8 bits
then right shift the 8 higher bits.

    LPC_UART0 -> LCR &= ~(1 << 7); //Disable access to Divisor Latches.
}


//Transmitting here.
void TransmitTx(char input){
    while( (LPC_UART0 -> LSR & (1 << 5)) >> 5 != 1 ); // Wait till previous
data is transmitted.
    LPC_UART0 -> THR = input; //THR have the data to be transmitted.
}
```

```c
//Receving here.
char ReceiveRx(){
    char input;
    while( (LPC_UART0 -> LSR & 1) != 1 );  // Wait for data to be received.
    input = LPC_UART0 -> RBR; //RBR have the recently recevied data.
    return input;
}


int main(){
    SystemInit();
    UARTInit();  // Initialize UART.

    TransmitTx('T'); //Transmitting character by character.
    TransmitTx('a');
    TransmitTx('r');
    TransmitTx('i');
    TransmitTx('q');

    char arr[]="\nExperiment Design";
    for(int i=0; arr[i]; i++){  //Transmitting an array.
        TransmitTx(arr[i]);
        }

    char input;
    while(1)
    {
        input = ReceiveRx(); //Recieve input from the user.
        TransmitTx(input);   //Transmit the received input.
    }
}
```

Output:



Another method to configure UART communications, is by using MBED libraries that will be easier rather than writing the whole process.

```
#include "mbed.h"

Serial pc(USBTX, USBRX); // This is for Tx and Rx USB here indicates that
the input and the output will be read and sent from and to the USB.

int main() {
    pc.printf("Hello World!\n");
    while(1) {
        pc.putc(pc.getc());// This will read the user's input and print it.
    }
}
```

Output: