Debugging with UART

# Project Description

This project will serve as an introduction to the Universal Asynchronous Receiver/Transmitter (UART), which can be used to interface with the microcontroller.

# Project Requirements

At the end of this project, you should be able to use a serial terminal to control the LEDs on the board. The terminal will use UART to communicate with the MSP.

## Universal Asynchronous Receiver/Transmitter

UART is a hardware device that allows for serial communication between two computers. The “Receiver/Transmitter” aspect is known as **full-duplex**, meaning it can both send and receive simultaneously. This can happen because of the dedicated transmit and receive buffers and pins, as shown in Figure 1. We see that, so long as the UCLISTEN bit is reset (set to 0), the receive shift register will store input obtained serially from the UC0RX pin, and send it to the 8-bit UC0RXBUF register. Likewise, an 8-bit value stored in UC0TXBUF will be transmitted serially out the UC0TX pin. Note that both the RX and TX operations are controlled by the same clock. As a result, in order to receive data, one must send data as well. This can be done by placing all zeros in the transmit buffer UC0TXBUF. As stated in the user guide, writing data to the TX buffer immediately begins the transmission (p. 419).

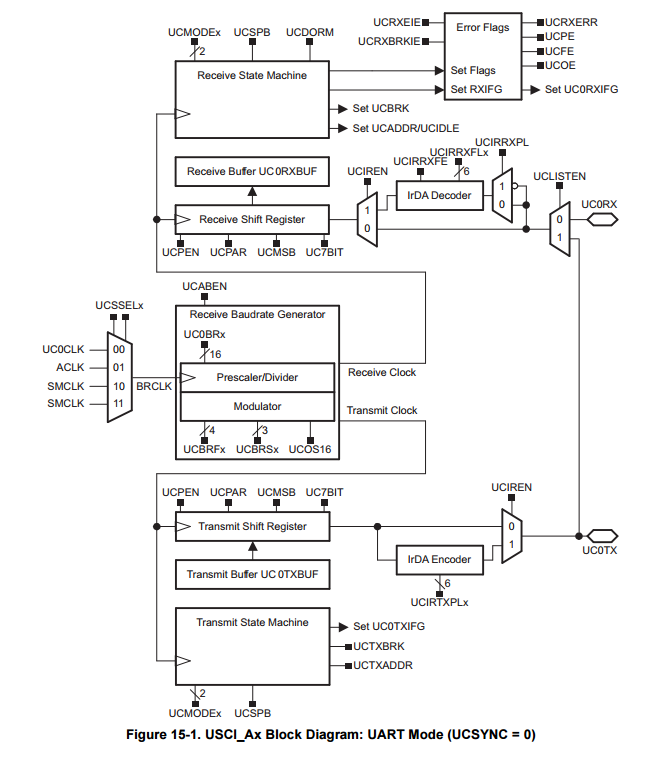


Figure : Schematic of UART Hardware in MSP430 Microcontroller.

## Selecting a Baud Rate

A commonly-used term in data transmission is the **Baud rate**. It is distinguished from the bit rate of a data line in that it relates directly to the capabilities of the hardware. Baud rate describes the number of changes in the signal per second, rather than the number of bits themselves that are present. For example, a computer may be able to output 10,000 bits per second to a controller, but if the microcontroller can support only 9600 baud, then in one second, it can only read 9600 changes in the signal. This means that although it is possible to not have lost any data, this will depend specifically on the data being sent.

9600 baud is generally used for most applications involving UART and serial communication because it is fast enough. Transmitting characters to a serial terminal window and displaying them need not be any faster, because the user would not be able to tell the difference.

## The Interface

You are tasked with implementing the interface depicted by the flowchart in Figure 2. This is a simplified flowchart in which white boxes are text output to the terminal and blue boxes represent some action taken by the software. The input can be dealt with by a switch-case statement, the default handling erroneous entries.

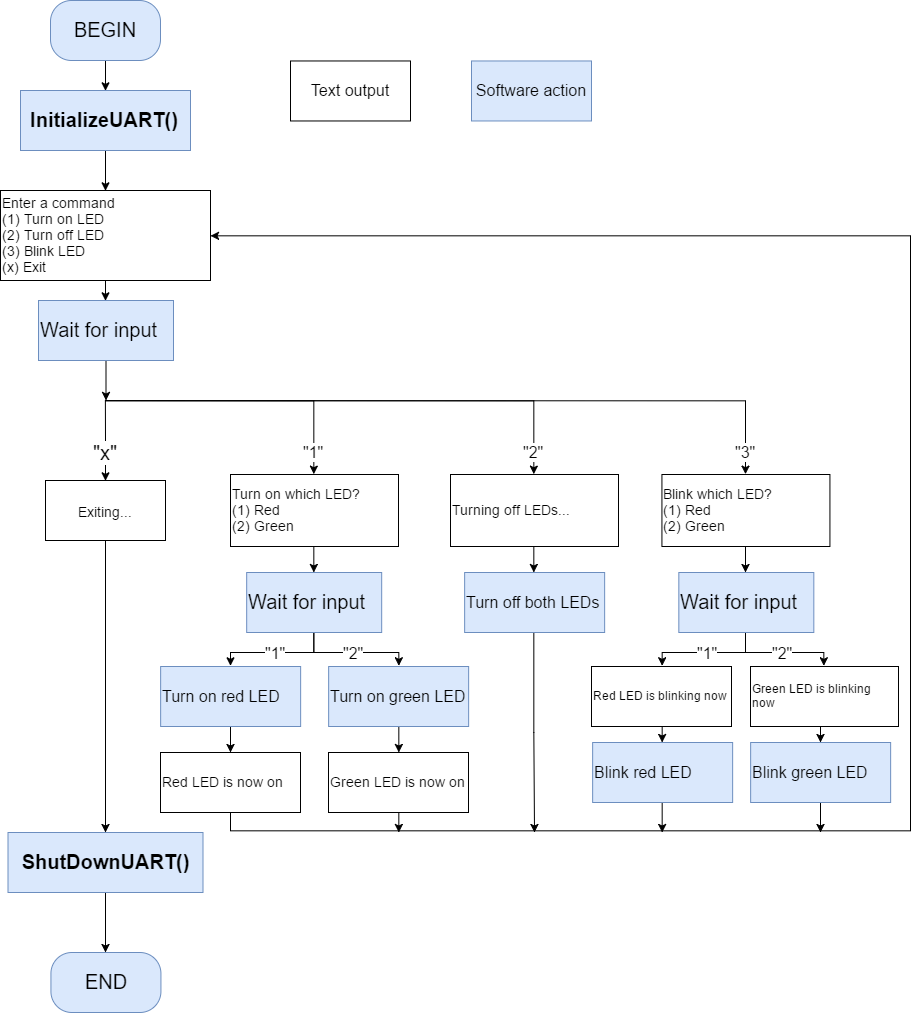


Figure 2: Flowchart for UART Interface.

## Opening a Serial Terminal

In order to view serial data, one needs to open a serial terminal. Although CCS does have this capability, we suggest making use of PuTTY or RealTerm. These are programs that emulate a variety of terminals, and are commonly used outside the classroom. RealTerm has a relatively intuitive interface, and PuTTY is the lightest-weight. Choose whichever suits your needs. See the slides for details on how to open a serial terminal using each.

To open the terminal on the correct port, you must know to which COM port your MSP is connected. This can be done in the UNIX command line on MAC OS or in the Device Manager in Windows.

# Project Grading

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Point Value** | **Points Earned** |
| Interface with the serial terminal | 10 |  |
| Control the LEDs | 5 |  |
| Verification | 5 |  |
|  |  |  |
| Total points | 20 |  |