CECS-412-01

Summer 2014

Project #

Copyright 2017, Jack Dorne, Sam DuPlessis, Matt Elmlinger, Aryan Ghazipour, Logan Schaufler

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Report  (20 Points) | Demo  (15 Points) | Quiz  (5 Points) |
| Group 8  Jack Dorne, Sam DuPlessis, Matt Elmlinger, Aryan Ghazipour, Logan Schaufler |  |  |  |

LAB 3

Group 8

CECS-412-01

Abstract

I/O communications through implemented drivers written in C are studied in this lab. This device onboard the A3BU that is used to accomplish this is through the Universal Synchronous Asynchronous Receiver Transmitted (USART), an I/O Peripheral device that permits serial I/O Communication. First, an RS232C serial interface circuit was built which allowed for asynchronous serial communication between the A3BU and a PC running terminal communication software, and an example driver was flashed onto the board to initialize the USART. If done correctly, ASCII characters would be communicated from the PC terminal to the A3BU, and then echoed back to be displayed on the terminal. Next, LCD synchronous communications were investigated through the PORT D Serial Peripheral Interface (SPI) and the LCD ST7565R MCU. Another example driver was flashed onto the A3BU to initialize the port. This time, lines were first drawn across the screen and then the screen was rolled. Finally, the two example drivers were integrated into one driver called SYSTEM, and a MAIN was written to receive ASCII characters from the keyboard to be displayed on the LCD. Through the successful completion of this lab, basic communication between an MCU and external hardware is understood.

Body

# Part 1

The Tera Term software was downloaded for aid in every portion of the lab. Tera Term was used as the chosen terminal program to display procedures that would be carried out to the ATxMEGA256A3BU Xplained board. To interface the A3BU board, a RS232C serial interface circuit (Schematic located in “Schematics” Section) was implemented. The circuit was then assembled and inspected. Once the circuit was set up, the program USART\_EXAMPLE was built and stored on the A3BU board. The function of USART was to communicate to the A3BU board by way of TxD and RxD asynchronously (of header J1). Pin PC3 served as the TxD pin and PC2 served as the RxD pin on the A3BU board.

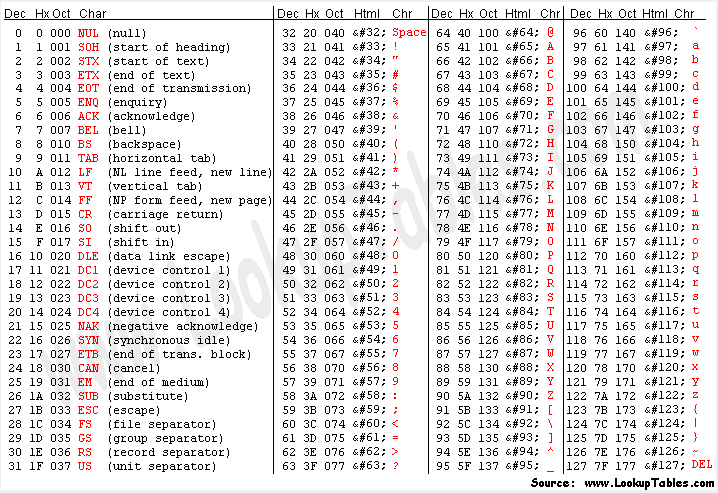
Serial port configuration settings next had to be configured. The number of data bits was set to 8, BAUD RATE to 9600, one stop bit used, and no parity bit was necessary in the C program (Tara Term would also follow these settings).

The program outputted a “Hello World” message on the Tera Term screen. This was possible by the translation of ASCII characters from the terminal back to the terminal by way of Tera Term.

The code was then re-written. The new program allowed for two single digit numbers to be added together. Again, ASCII values were translated, this time to single digit values. Since 0 has an ASCII value of 48, 48 had to be subtracted from the result of the addition of the two digits (any addition would result in a “c” output rather than the true sum value). This displays the true value of the new digit. This is an example of when digits “3” and a “6” are inputted through Tera Term, which results in their sum, 9.



An ASCII table was necessary for reference in part 1:



The rate at which bits are communicated between the PC and the A3BU is called “BAUD rate.” The RS232 aids by allowing both the PC and A3BU to know when bits are being communicated, and subsequently when this processes is stopping. A higher BAUD rate allows for data to transferred faster, however this comes with risk as data can be misinterpreted. A lower BAUD rate may run at a slower rate, but data is rarely, if ever, misinterpreted.

A chart of the USART registers that were involved in the program:

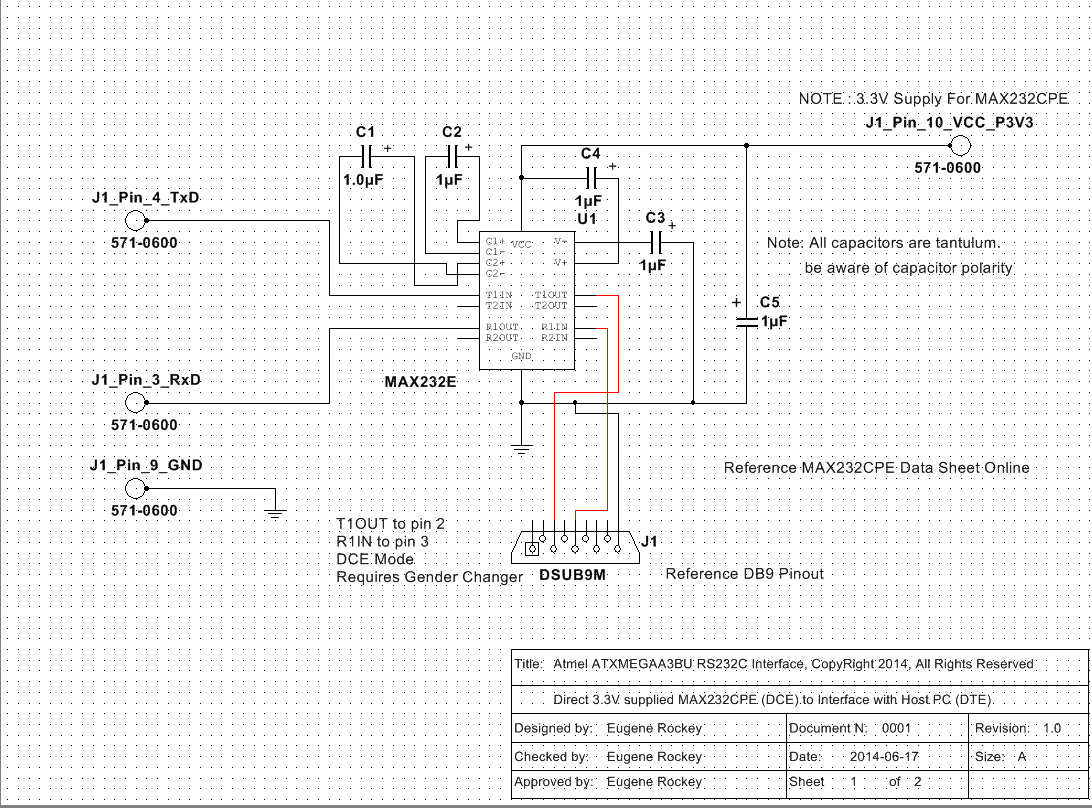
|  |  |
| --- | --- |
| Register | Purpose |
| UDR | 16-bit USART Data Register. The Transmit Data Buffer Register (TXB) is where UDR received data is stored. |
| UBRRH | Stands for USART Baud Rate Register High. The first 8 bits of the 16-bit BAUD rate register are stored here. |
| UBRRL | Stands for USART Baud Rate Register Low. The last 8 bits of the 16-bit BAUD rate register are stored here. |
| UCSRA | 8-bit USART Control and Status Register A. The flag bit is set by the CPU when unread data is in the Receive Buffer, and then cleared by the CPU when the Buffer is empty. |
| UCSRB | 8-bit USART Control and Status Register B. The UCSRB enables interrupts, receiver, transmitter, and third bit character size. |
| UCSRC | 8-bit USART Control and Status register C. Allows for stop bits, Asynchronous or Synchronous mode, parity type, selection of registers, and selection of two bits of character size. |

(The USART of the AVR » MaxEmbedded)

# Part 2

Source Code (Software)

Schematics (Hardware)



Analysis

Conclusion

References

1. "The USART of the AVR » MaxEmbedded." *MaxEmbedded*. N.p., 04 Nov. 2015. Web. 23 June 2017.