**ESE381 Embedded Microprocessor Systems Design II**

Spring 2024, K. Short revised February 24, 2024 4:47 pm

**PRELIMINARY PRELIMINARY PRELIMINARY PRELIMINARY PRELIMINARY**

## Laboratory 05: AVR128DB48 USART Module in Asynchronous Serial (RS232) Mode and Saleae Logic Analyzer

To be performed the week starting February 25th.

### Prerequisite Reading

1. Lecture 09: AVR128DB48 USART in Asynchronous Serial (RS232) Mode
2. AVR128DB48 Data Sheet, Section 27: USART - Universal Synchronous and Asynchronous Receiver and Transmitter.
3. Analog Devices ADM3202 Low Power, 3.3V, RS232 Line Drivers/Receivers Data Sheet.

### Overview

When an external peripheral is added to a system, the microcontroller must be able to communicate with it. To minimize the number of microcontroller and external peripheral pins needed for this communication, most recent peripheral ICs have serial interfaces. There are several popular serial interfaces, including RS232, SPI, and I2C.

RS232 is a standard for transmitting serial data. It was originally created to transfer data between systems a large physical distance apart. When used in those types of applications the logic voltage levels for the transmitted and received signals are different from the signals in the systems themselves. This is done in order to provide greater noise immunity. The specification of these signal voltage values is part of the RS232 standard.

Serial data transfer requires the receiver to able to sample the incoming data in the center of each bit received. For synchronous serial data transfer a sample clock signal between the transmitter and receiver makes this simple. In asynchronous serial data transfer a sample clock signal between the two devices is not used. The receiver must derive a sampling clock from the received data itself. This approach is called “clock recovery”.

The serial protocol used to transmit serial data can be implemented in software or hardware. If done in software, as in Laboratory 04, the microcontroller CPU is dedicated to implementing the protocol when a transfer is in progress. If hardware is used instead, the protocol implementation is accomplished by the hardware and once a transfer is initiated, the microcontroller’s CPU can do some other tasks.

Early integrated circuit implementations of receivers and transmitters for the asynchronous serial protocol used in RS232 were called **universal asynchronous receiver/transmitters (UARTs)**.

Later versions of these ICs could also support synchronous serial data transfer. These **universal synchronous/asynchronous receiver/transmitters are called USARTs**.

Using an on-chip UART we can eliminate the need to execute a large number of instructions to read or write data bits from or to a compatible device. For maximum flexibility and wide applicability, most microcontroller's have on-chip serial data transfer modules that are configurable to meet the RS232 serial protocol. Thus, they contain one or more control registers that are written to specify the module's configuration.

While in operation, an on-chip serial module has one or more status registers that are used to provide information about the current state of the module. On-chip modules also have one or more data registers. Writing the transmit data register initiates a serial data transmission. Reading the receive data register returns received data.

When asynchronous serial is used to communicate with an external peripheral IC or system, the microcontroller's serial module must be configured to be compatible with the device it communicates with. This compatibility includes format and baud rate.

The AVR128DB48 has five USART modules for serial asynchronous and synchronous communication. These USARTs can be configured to achieve serial communications using the asynchronous (RS232) protocol. The lecture notes and the assigned reading from the AVR128DB48 Data Sheet provide details on the AVR128DB48 USART's asynchronous serial operation.

Major objectives of this laboratory are reinforcing your understanding of the asynchronous serial protocol and familiarizing you with the use of an AVR128DB48 USART module to implement asynchronous serial communications.

An additional important objective is for you to become familiar with the use of a "protocol aware" logic analyzer or oscilloscope to observe and verify serial transmissions. These instruments can be used to view serial data being transmitted and received by a microcontroller. Such an instrument has significant advantage over use of an oscilloscope that does not have "protocol aware" capabilities. Because it is protocol aware, it can decode the data being transmitted and received and display it in decoded form. You will use Saleae's 16-bit logic analyzer for this purpose. This device is easy to use and its operation is described in the Saleae Logic Software Users Guide.

You should download the Saleae software for this logic analyzer from https://www.saleae.com/downloads/. This software has a fully-featured demo mode that allows you to learn the operation of the logic analyzer without having the actual logic analyzer hardware. You should familiarize yourself with the operation of this logic analyzer outside of the laboratory before your laboratory session. In the laboratory, endeavor to take advantage of any extra time available, after you have completed your required tasks, to become more proficient in the use this logic analyzer. Proficiency in its use will be invaluable to you when debugging RS232 and other serial communications protocols later in this course.

**Design Tasks**
The designs in this laboratory involve writing simple programs to configure **USART3** to transmit and receive data using the asynchronous serial protocol. You will verify the results of your work using the oscilloscope and Saleae logic analyzer.

***Design Task 1: Using USART3 to Transmit a Single Character and Using the Saleae Logic Analyzer to Verify the Transmission***
When using any type of serial communication, a good place to start is to see if you can transmit a single character in the desired protocol. Write a program named `USART3_asynch_test`. This program transmits a single character each time it is run. The character transmitted is observed using both the oscilloscope and Saleae logic analyzer.

You are not to receive any characters, just send them. You must transmit the character at 9600 baud with 8 data bits, no parity, and a single stop bit (8N1). PB0 of the Curiosity Nano board, is the Tx output of USART3. PB1 (pin 17) of the Curiosity Nano board is the Rx input of USART3.

At home you can compile the program and single step through its operation. Since you probably don't have an oscilloscope or logic analyzer at home, you want to primarily check whether your program actually configures the USART3 module as required. You can do this by looking at the registers in the USART3 module in the I/O window.

If you have Tera Term or some other terminal emulator on your own PC you can view the characters sent by your program in the Tera Term monitor window.

**Submit your C source file for this program as part of your prelab.**

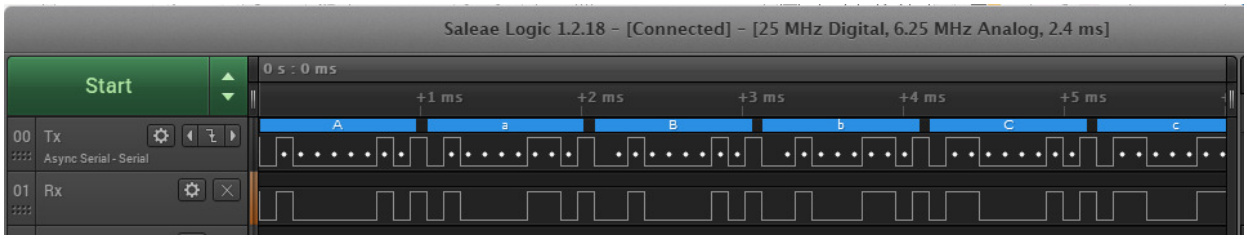
***Design Task 2: Using USART3 to Transmit Characters A to Z via Asynchronous Serial.***
Write a program named `A_to_Z_async_Tx` that uses USART3 to continuously transmit the sequence of capital letters A through Z.

**Submit your source file for this program as part of your prelab.**

***Design Task 3: Loopback of Tx to Rx***
One simple way to check whether a serial communications design works is to connect the transmitting line directly to the receiving line. This is called a **loopback test**. Normally, you would simply connect the USART's Tx output pin to its Rx input pin. That way any character sent by your program should be received by your program. However, since we are using USART3 and its RX and TX pins are also connected to the Curiosity's Virtual Serial Port (CDC) we cannot do this. Instead you must use the **Loop-back Mode Enable (LBME)** feature of the UART module that enables an internal connection between the TXD pin and the USART's receiver and the RX input pin of the USART receiver is disconnected. The LBME bit is in CTRLA.

Write a program named `USART3_loopback`. This program continually transmits an uppercase character in the sequence from A to Z. After it transmits a character and before it transmits the next uppercase character, it takes the character it received and converts it to lower case and then transmits it. Then the program transmits the next upper case character in sequence and converts the character received to lower case and transmits it.

The output you want to see on the logic analyzer is:




Your transmitted output must have each upper case letter followed by the lower case version of the received character.

**Submit your C source file for this program as part of your prelab.**

***Design Task 4: Echo of Characters Sent by a PC Using a USB Port***

In the previous task we've verified that our system (the Curiosity Nano board) can asynchronously transmit characters and can receive characters transmitted by itself. Now we want to have our system communicate with another system, which would normally be the objective. In many applications this other system is an external peripheral IC with an asynchronous interface. In that case, the logic level voltages used would be those of the ICs and not the voltages specified by the RS232 standard.

In other cases we might want to connect to a PC. In connecting to a PC there are two options. Some PCs have a DB9 RS232 connector which uses RS232 standard compatible voltages. Most desktop computers have this connector. But, for portable computers this connector is often not available and a USB connector must be used instead.

For this task you will use the Curiosity connection to the Tera Term via its CDC TX pin and its USB connection as you did in Task 1 of Laboratory 04.

We want to transmit a character from the PC and have the Curiosity Nano board echo it back. To transmit the character we will use the Tera Term terminal emulator program running on the PC.

Write a program named `USART3_echo_usb`. This program simply receives a character and writes it back.

**Submit your C source file for this program as part of your prelab.**

***Design Task 5: Echo of Characters Sent by a PC Using an Actual RS232 Port***

The laboratory computers have an actual RS232 port, COM1, on the back of the computer. This port is connected to using a DB9 connector, also specified in the RS232 standard, and uses valid RS232 voltage levels.

You must use an Analog Devices ADM3202 Low Power, 3.3V, RS232 Line Drivers/Receivers IC to translate between COM1's RS232 logic levels and the Curiosity Nano board's 3.3V logic levels.

Draw a schematic showing the connection of the ADM3202 to the signals of the Curiosity Nano board. Use driver T1 and receiver R1 in your design. Don't forget to include the capacitors required for the ADM3202 charge pump's operation. Write a program named `USART3_echo_rs232`. This program is the same as `USART3_echo_usb`.

**Submit your schematic file for this program as part of your prelab.**

**Laboratory Activity**

***Laboratory Task 1: Using USART3 to Transmit a Single Character and Using the Saleae Logic Analyzer to Verify the Transmission***

Create a project named `USART3_asynch_test` using the program `USART3_asynch_test` that you wrote. Set the compiler optimization to -Og. Build the program. Place any variables in a watch window to observe their values. Single step through each instruction in the program. Prior to single-stepping each instruction, determine what changes you expect in the values of the port and USART3 registers and any variables you created.

*(a) Verification with Oscilloscope*
Use the oscilloscope with Single selected as its Run Control to capture your program's output. This output must be the same as in the figure in Design Task 1. Use the oscilloscope's trace and its measurement capability to verify the Baud rate.

**When your program is working correctly and generates the required trace, have a TA verify that your program performs as required. Get the TA's signature.**

*(b) Verification with Saleae Logic Analyzer*

Connect the Saleae logic analyzer's USB connector to a super speed USB connector on the PC. Connect the Saleae logic analyzer's lead 0 and associated ground to the Curiosity Nano board's Tx signal and ground, respectively.

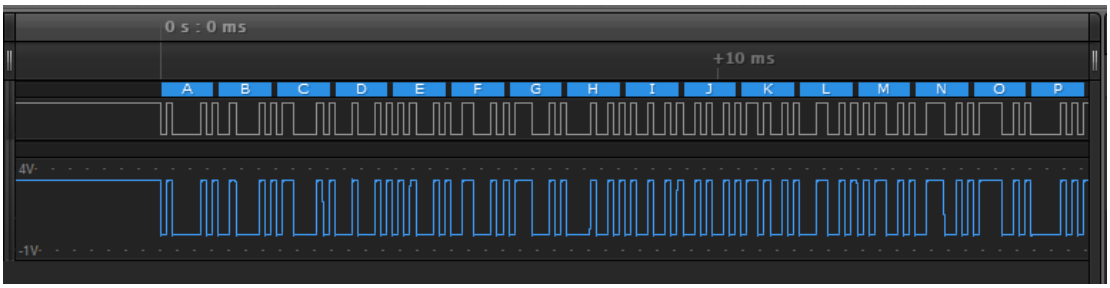
**When your program is working correctly and generates the required trace on the Saleae, have a TA verify that your program performs as required. Get the TA's signature.**

***Laboratory Task 2: Using USART3 to Transmit Characters A to Z via Asynchronous Serial.***
Create a project named `A_to_Z_async_Tx` using your program `A_to_Z_async_Tx`. Set the compiler optimization to -Og. Build the program. Place any variables in a watch window to observe their values. Single step through the program. When you get to a function call of a function you already verified in the previous task, you can step over it (F10) instead of stepping into it (F11). Stepping over a function still executes the function. Only execution doesn't stop until the function completes. Thus, you have executed the entire function as one step. Prior to single-stepping each instruction, determine what changes you expect in the values of the port registers and any variables you created.

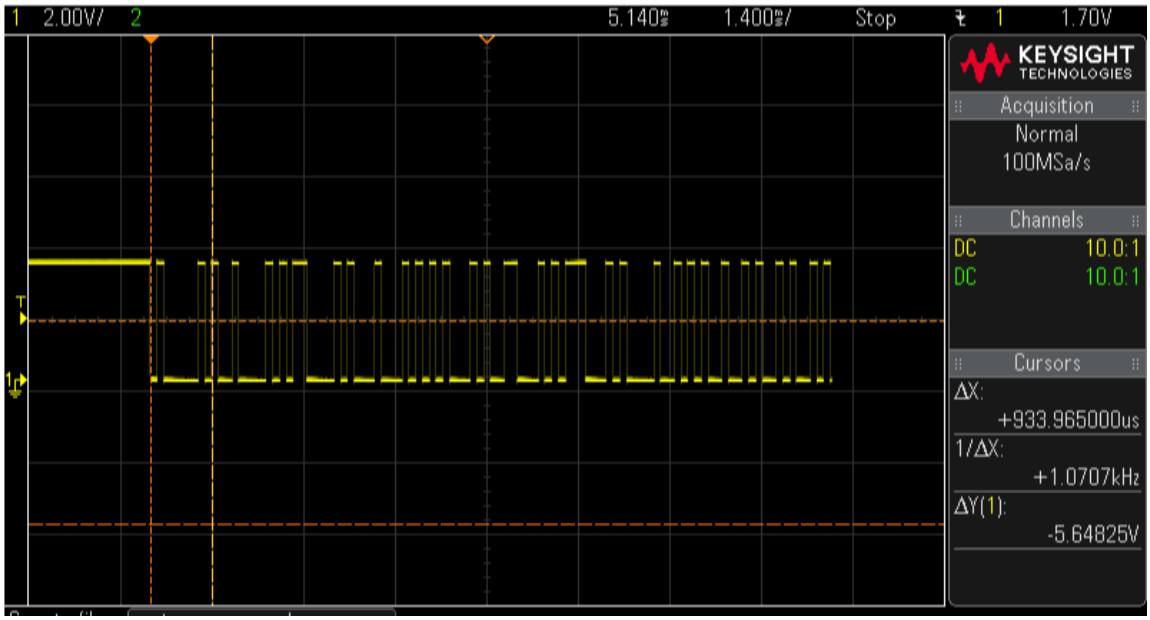
Configure the oscilloscope as in Task1. Configure the Saleae logic analyzer as in Task1, except set the Display Radix to Ascii only. Run and debug your program. Use a breakpoint to send just one full sequence of the characters A through Z.

Comparing the oscilloscope's output with that of the logic analyzer, it is clear that if you were transmitting or receiving ASCII data it would be much better to have an instrument that is protocol aware and decodes and displays each character for you as in the top waveform in the follow-

ing figure. The bottom waveform is the analog version of the signal without protocol aware decoding




The oscilloscope you are using has the capability for protocol aware decoding as an option that we don't have installed. So, you would be left trying to figure out whether the following waveform is correct.



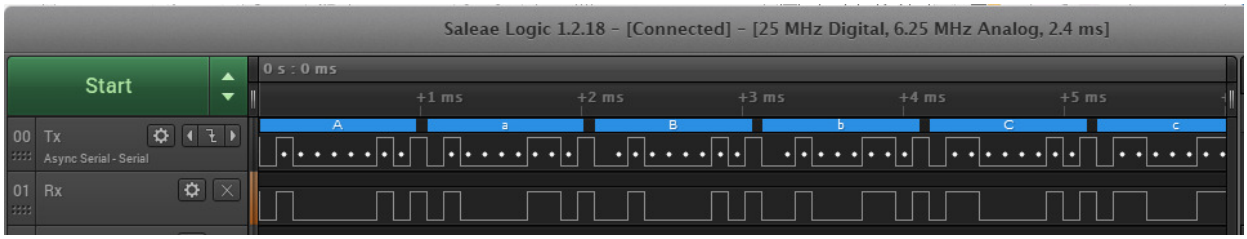

**When your program is working correctly and you have the correct waveforms, have a TA verify that your program performs as required. Get the TA's signature.**

***Laboratory Task 3: Loopback of Tx to Rx***

Create a project named `USART3_loopback` using the program `USART3_loopback` that you wrote. Set the compiler optimization to -Og. Build the program. Place any variables in a watch window to observe their values. Single step through the program. Prior to single-stepping each instruction, determine what changes you expect in the values of the port registers and any variables you created.

Use the logic analyzer to capture the your program's output. You will need to configure Channel 1

for Rx, so that you can see both the transmitted and received characters. They should be the same since Tx and Rx are connected together. You objective is the following output.



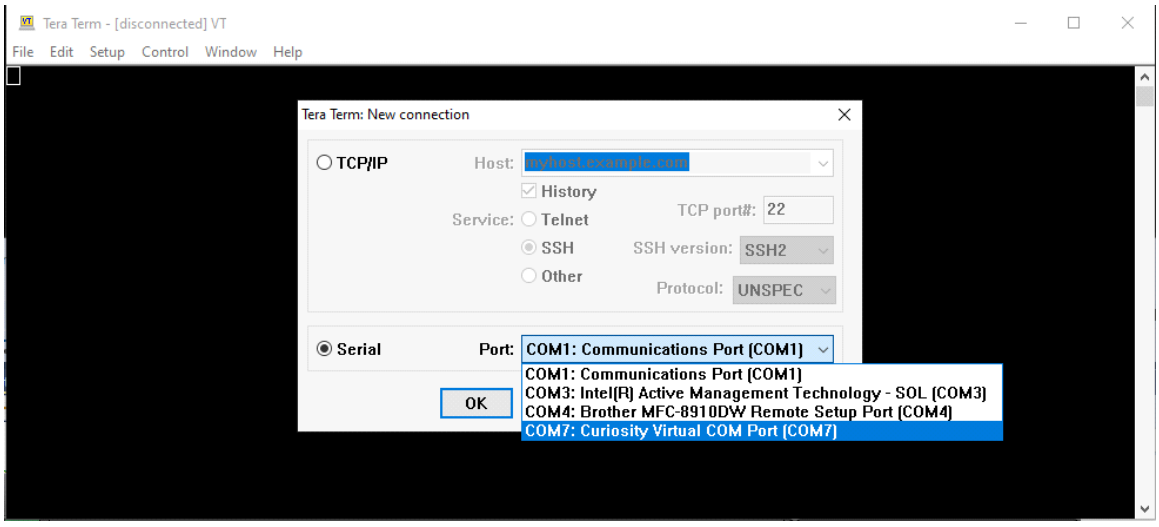

**When your program is working correctly and generates the required trace, have a TA verify that your program performs as required. Get the TA's signature.**

***Laboratory Task 4: Echo of Characters Sent by a PC Using USB Port***

Use the Tera Term terminal emulation application on the PC to emulate a simple terminal.

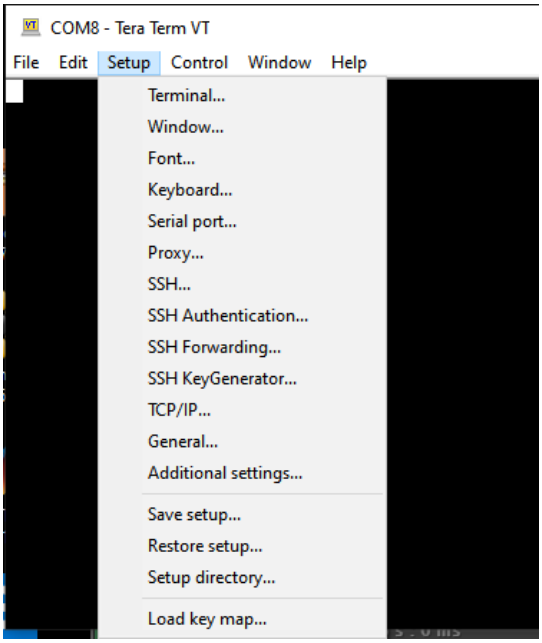
Click on the Tera Term icon.


Go to File > New Connection.

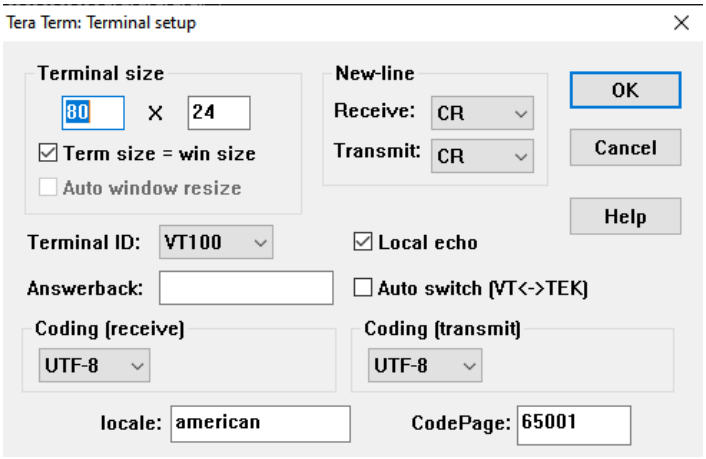



Click on the Port drop down menu. Look for the COM port that is associated with Curiosity Virtual COM Port and select it.

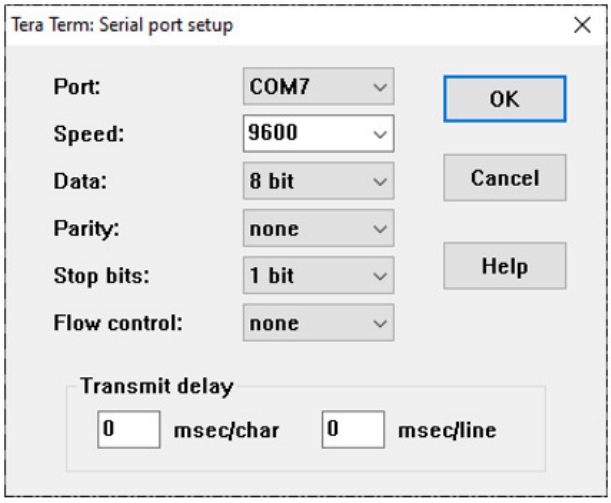
Click on the Setup tab and select Terminal



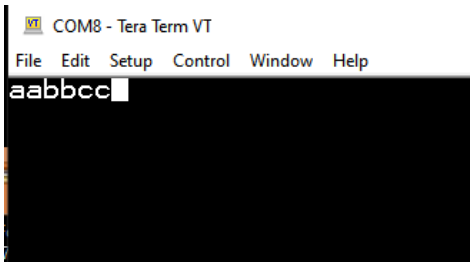

Make sure the settings are as shown below, including having Local echo checked.




Next in Setup select Serial port. Make sure that Port is the port that you selected. Make sure the other settings are as shown below. Since we are communicating between two different systems we have to make sure that the are both configured for the same speed and format, 9600 Baud, 8N1 and no flow control.




When you type a character into the terminal window it is transmitted to the Curiosity Nano board by the PC and also displayed on the PC Monitor (because Local echo is selected). The Curiosity Nano board receives the character and sends it back to the PC where Tera Term displays it. Thus, you have two copies of the character you typed displayed in the terminal window.




If you deselect Local echo you will only see one character displayed for each character you type. That is the character received by the PC from your microcontroller.

**When your program is working correctly and generates the required data in the terminal window, have a TA verify that your program performs as required. Get the TA's signature.**

***Laboratory Task 5: Echo of Characters Sent by a PC Using an Actual RS232 Port***

Wire your ADM3202 circuit for the RS232 driver IC. Wire the RS232 level Tx, Rx, and GND signals to a place on the breadboard where you can plug in the 3-pin connector of the RS232 cable. The red wire of the RS232 cable is the Tx wire from the PC. The clear wire is the RX wire to the PC. The black wire is GND. Have a TA check your wiring before proceeding.

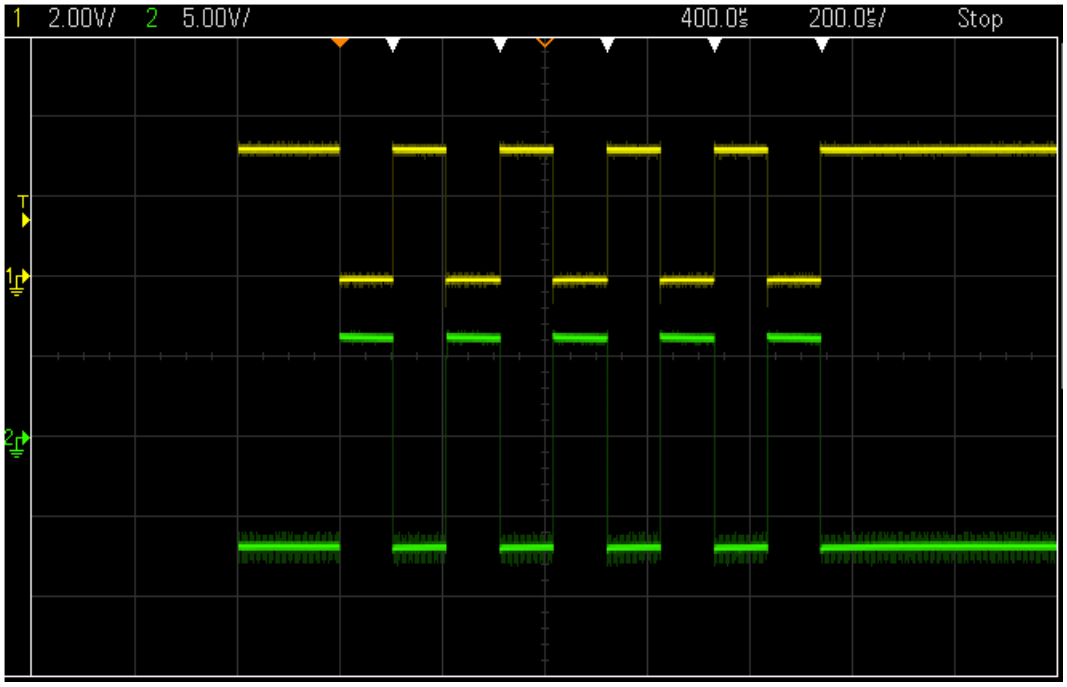
# Do not connect the Saleae logic analyzer to any of the RS232 signals. Their voltage levels are beyond the safe range for the Saleae logic analyzer!!!!

Connect the Curiosity Nano board to its USB connection to the PC. Using the **oscilloscope**, measure the voltage at V+, pin 2 and V-, pin 6 of the ADM3202. These are the positive and negative rails being generated by the ADM3202's charge pump. Measure the voltage on pin 14 of the ADM3202. This is the Tx output of the ADM3202 driver and represents the unloaded RS232 logic 1 level generated by the ADM3202 while the transmit line is marking. Record these values.

Connect the RS232 cable to the COM1 port at the back of the receiver. Note the change in the voltage on pin 14, now that the driver is loaded.

Using the oscilloscope, measure the voltages from the RS232 cable with respect the black ground wire of the cable. The red wire, which is the Tx wire from the PC, should be about -10V. The clear wire, which is the RX wire to the PC should be approximately 0V.

You will need to show the TA the following trace of the TX input to the ADM3202 and the TX output from the ADM3202 on your oscilloscope.




Set up Tera Term to communicate with the Curiosity Nano board. You will have to select COM1 and the serial port on the PC. The other Tera Term settings should be as in Task 4. When you type a character into the terminal window it is transmitted to the Curiosity Nano board by the PC and also displayed on the PC Monitor (because Local echo is selected).

**When your system is working correctly and generates the required data in the terminal window, have a TA verify that your program performs as required. Get the TA's signature.**

**Leave the circuit you have constructed on your breadboard insert, it may be used in later laboratories.**

**Problems**

1.