

# The ratload Handbook

Jacob Hladky  
California Polytechnic State University  
San Luis Obispo, CA  
jhladky@calpoly.edu

April 22, 2015

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Requirements . . . . .	2
1.2	Project Manifest . . . . .	2
1.3	Bug Reporting . . . . .	2
1.4	Project Information . . . . .	2
<b>2</b>	<b>Adding the New RAT Wrapper</b>	<b>3</b>
2.1	Integration with RAT CPU . . . . .	3
2.2	Verification . . . . .	4
<b>3</b>	<b>Adding the New Prog_Rom Module</b>	<b>5</b>
<b>4</b>	<b>Installation and Use of the ratload Program</b>	<b>6</b>
4.1	Installing ratload . . . . .	6
4.2	Operational Overview . . . . .	6
4.3	Identifying Your Serial Device . . . . .	6
4.3.1	Nexys 2 . . . . .	6
4.3.2	Nexys 3 . . . . .	7
4.3.3	Finding the Port Name . . . . .	7
4.4	Ratload Procedure . . . . .	7
4.5	Interrupts and the ratload System . . . . .	7
<b>5</b>	<b>How It Works</b>	<b>8</b>
<b>6</b>	<b>Troubleshooting</b>	<b>10</b>
6.1	Checking Your Serial Cable with the Reference Bit File . . . . .	10
6.2	Troubleshooting the UART Module . . . . .	10
6.2.1	Obtaining a Serial Console . . . . .	10
6.2.2	Configuring the Proper Serial Console Settings . . . . .	11
6.2.3	Using the Serial Console to Troubleshoot the UART . . . . .	11
6.3	Troubleshooting the Ratload Program . . . . .	11

# 1 Introduction

Ratload is a system for loading new RAT assembly programs on to the RAT CPU without the need for resynthesis.

The system consists of two components: a computer program and a set of VHDL modules. This guide will detail how to install, use, and — if necessary — troubleshoot both. This guide is for Windows users, but `ratload` can also be run on GNU/Linux and OS X. Instructions for those platforms are in a separate document, called “`linux_and_osx.pdf`”.

## 1.1 Requirements

If you have a Nexys 2 board — You will need a serial cable. If your computer does not have a serial port, then you will need a USB-to-serial adapter. Here’s an example: <http://amzn.com/B00065H0QQ>. Any adapter will do as long as you have the right drivers for it.

If you have a Nexys 3 board — No serial cable is required, but you will need another micro-USB cable.

Do not integrate ratload into your RAT CPU until all the major components of the CPU are in place, and you have a good understanding of how they fit together. Understanding how the RAT architecture handles I/O and interrupts will make integration much easier.

## 1.2 Project Manifest

The following is a general overview of the files in the system.

- `handbook.pdf` This guide.
- `linux_and_osx.pdf` Instructions for using ratload on GNU/Linux and OS X platforms.
- `new_rat_wrapper/` VHDL files necessary to integrate the new `rat_wrapper`.
- `new_prog_rom/` VHDL files necessary to integrate the new `prog_rom`.
- `testing/` Files for testing the operation of the new system.
- `ratload_v<version>` The ratload program.

## 1.3 Bug Reporting

All programs have bugs. Ratload is beta software and is no exception. If you encounter a bug in the ratload program, please contact the author. It is also possible that you find a bug in the project VHDL modules. If you do please verify the bug via simulation and report it immediately so it can be fixed.

Since you are writing the mechanics of the CPU itself, it is possible for very odd bugs to happen, and for those bugs to interact with `ratload` in bizarre ways... Please note that `ratload` has been thoroughly tested and is free of any major bugs.

## 1.4 Project Information

If you have any questions about the project itself, or suggestions for improvement for this guide, please contact the author at [jhladky@calpoly.edu](mailto:jhladky@calpoly.edu). This project licensed under the MIT license and the complete source code — including the  $\text{\LaTeX}$  source for this guide — is available at <http://www.github.com/jhladky/ratload>. Ratload system releases are available at <http://homecrest.hladky.me/ratload/>.

This project uses a UART module authored by Peter A. Bennett and released under the LGPL. The source for that module is available at <https://github.com/pabennett/uart>.

## 2 Adding the New RAT Wrapper

Ratload requires a UART to communicate with the host computer, which is provided as part of a new RAT wrapper module, which will replace your current wrapper. Figure 1 shows how your RAT CPU will fit into the new wrapper.

Note that all the I/O devices, such as the 7-segment display and the buttons, have been moved into “input” and “output” modules. The new RAT wrapper provides access to the LEDs, switches, 7-segment display, buttons, and VGA. If you need to use some other I/O device, such as the mouse or keyboard, you will need to integrate it with the RAT wrapper yourself.

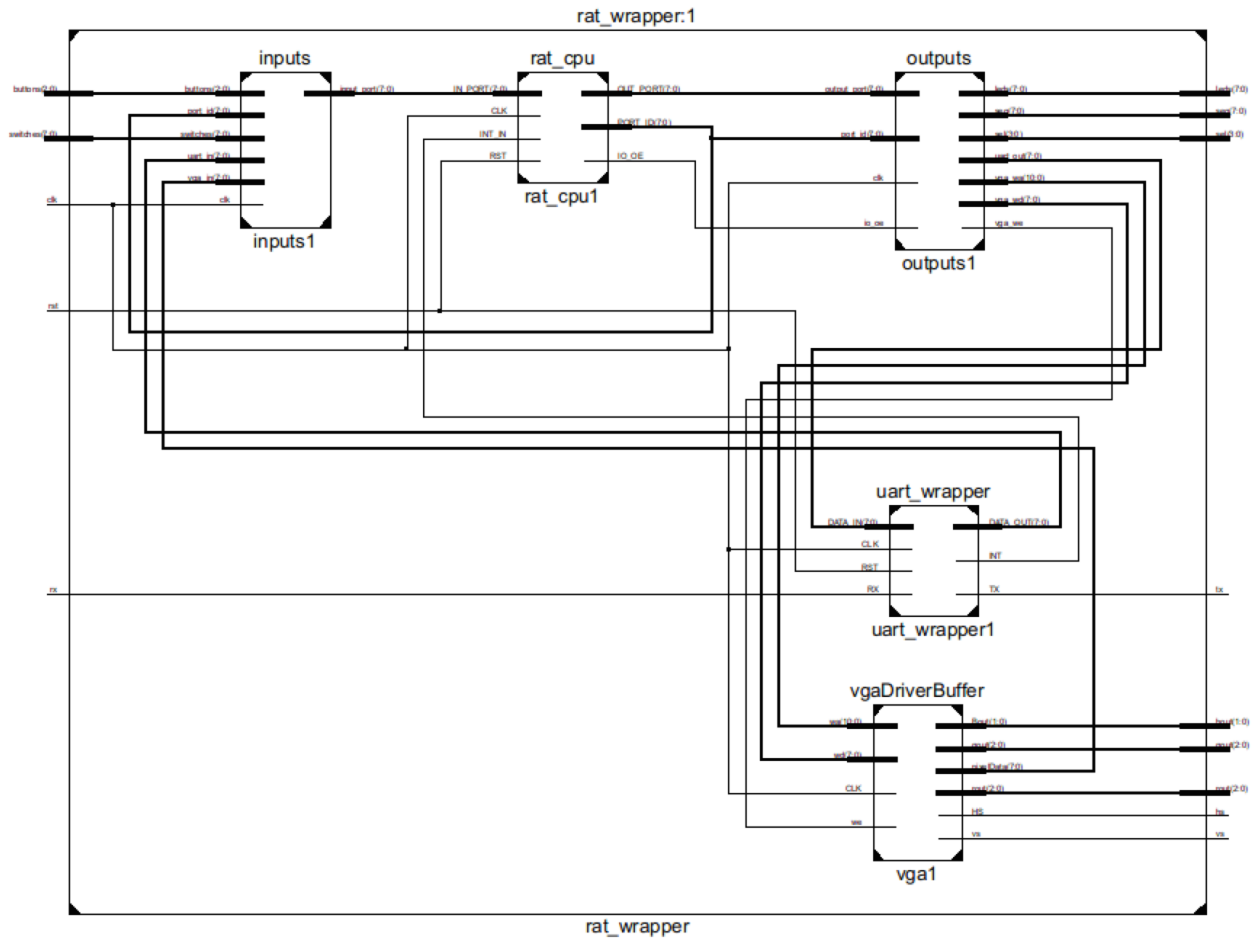


Figure 1: Overview of RAT Wrapper components

### 2.1 Integration with RAT CPU


1. Make a backup copy of your CPU right now! You are about to make some big changes to your project and you'll want something to fall back on in case something goes wrong.
2. Remove your RAT wrapper VHDL module, making your RAT CPU module the top-level module. Also remove your RAT wrapper ucf file.

 Double check you have backed up your RAT project! Otherwise you could lose everything!


3. Navigate to the `ratload` project directory (where this handbook is located), and then to the “new\_rat\_wrapper” directory. If you have a Nexys 2 board, delete the “rat\_wrapper\_nexys3.ucf” file and rename the “rat\_wrapper\_nexys2.ucf” file to “rat\_wrapper.ucf”. If you have a Nexys 3 board, do the opposite.
4. Go to **Project >Add Copy of Source**. Select all the files in the “new\_rat\_wrapper” directory and click **Open**. Another dialog box will pop up confirming you want to add these files. Click **OK** to confirm.
5. Go to **Project >Design Goals & Strategies**. Change the **Design Goal** to **Timing Performance**. Click **Apply** and then **OK**.
6. Go to **Project >Cleanup Project Files...** Click **OK**. This will “refresh” your project. Sometimes synthesis can fail for what seems like no reason, and this step is intended as a prevention of that. If you experience another such failure, repeat this step.
7. That is all that’s necessary to add the new RAT wrapper. However the signal names of your **RAT CPU module** may differ from those declared in the RAT wrapper. Change the signal names in the new RAT wrapper as necessary and make sure you can generate a bit file.

## 2.2 Verification

This section covers a quick test of the new RAT wrapper you just integrated into your RAT CPU. If the test in this section is not 100% successful **DO NOT CONTINUE**. Go back and double check that you have integrated the UART properly. If the UART continues not to work, see Section 6. `Ratload` **WILL NOT WORK** unless your UART behaves exactly as expected!

 UART stands for “Universal Asynchronous Receiver-Transmitter”, which you may recognize as another name for a serial port. The UART can be used outside of the `ratload` system as well. If you would like more info about using the UART in your final project, contact the author.

1. Delete your current `prog_rom` module from ISE by right-clicking on the file and clicking **Remove**. Then right-click again and select **Add Copy of Source**. Navigate to the “testing” directory and add the “serial\_test.vhd” file. This will replace your current `prog_rom` module with the `prog_rom` module in the “serial\_test.vhd” file.
2. Program your Nexys board with the new bit file.

 The next steps involve running the `ratload` program. For more information about installing and running the `ratload` program see Section 4.

3. Connect your serial cable between the Nexys board and the host computer. Open the `ratload` program and select the proper serial device from the dropdown menu. Ignore the “Choose File...” prompt. If you are having trouble identifying your serial device, see Section 4.3.
4. Go to **Menu >Run Serial Test**. The program will then attempt to communicate with the Nexys board via the serial cable.
5. If the test indicates “PASS”, proceed to the next section. Otherwise go straight to Section 6. Do not pass Go. Do not collect \$200.

### 3 Adding the New Prog\_Rom Module

Once you have added the UART and verified that it is functioning correctly, the remaining steps are simple.

1. Select the prog\_rom file in your RAT CPU and right-click **Remove**.
2. Go to **Project >Add Copy of Source**. Navigate to the “new\_prog\_rom” directory and select all the modules within. Click **Open**. Another dialog box will pop up confirming you want to add these files. Click **OK**. If it asks you about overwriting any files, click **OK**.
3. In the architecture section of the rat\_cpu module, edit the component declaration for the prog\_rom module. Add the following line:

```
TRISTATE_IN : in STD_LOGIC_VECTOR(7 downto 0)
```

4. In the same file, edit the port map declaration for the prog\_rom module. Map the signal for the RAT CPU’s tristate bus into to the prog\_rom module’s tristate bus. The RAT CPU’s “tristate\_bus” may be called the “MULTI\_BUS” in your CPU. (Regardless of its name, this is the bus that connects the ALU, the program counter, the scratch pad, and others.) That line will look similar to this:

```
tristate_in => tristate_bus_sig(7 downto 0),
```

5. Synthesize and generate a bit file for your newly integrated system. That’s it! Figure 2 shows what the updated prog\_rom module should look like. Ratload should work on your CPU now. The next sections cover how to set up the ratload program on your computer.

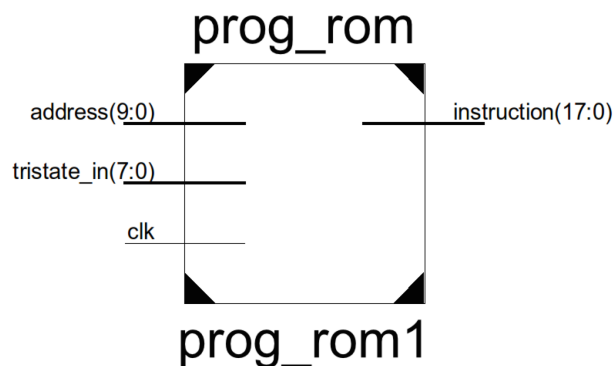


Figure 2: Updated prog\_rom Black Box Diagram

## 4 Installation and Use of the ratload Program

### 4.1 Installing ratload

Installation is really easy. The ratload program is located in the a directory called “ratload\_v<version>”, where <version> is replaced by whatever version the program happens to be in. Simply copy that directory to someplace useful to you. Run the program by running “app.exe”. There is no other installation step.

If a GUI is too fancy for you then you can also use the command line version of ratload. Instructions for doing so are available in the “linux\_and\_osx” PDF also included in this project.

### 4.2 Operational Overview

Up until now, you’ve probably followed this pattern when developing your assembly language programs:

1. Run your assembly repeatedly in the ratsim program until it produces a prog\_rom.vhd file and seems to be bug free.
2. Add the new prog\_rom.vhd file to your project, replacing the old one.
3. Resynthesize the entire project and reprogram the bit file onto your Nexys board.
4. Repeat ad infinitum.

The steps you have just followed to integrate the vhd files into your RAT CPU, and to install the ratload program onto your computer, will dramatically change this pattern. From now on you do not need to resynthesize your project when ratsim generates a new prog\_rom.vhd file, nor is it necessary to copy that new file into your project. In fact, **making any more changes to the prog\_rom.vhd file in your project directory will break ratload.**

This section contains instructions for identifying the serial device you need to use to communicate with the Nexys board, an overview of the `ratload` procedure, and instructions on using the ratload program.

### 4.3 Identifying Your Serial Device

How you identify your serial device will depend on what Nexys board you have as well as whether you are using a USB-serial adapter or not. Before you identify your serial device you need to make sure you have the correct drivers installed for it. This sounds a lot more complicated than it really is. Follow either of the following sections based on which Nexys board version you have to make sure you have the right drivers installed.

#### 4.3.1 Nexys 2

If you have a serial port on your computer, congratulations, you are using a computer from the 20th century! Windows probably already has installed drivers to use this port, so you don’t need to do anything else.

If you don’t have a serial port, then you need to obtain a USB-to-serial adapter cable. The manufacturer of the cable probably provided drivers for you to install along with the cable itself. Try to google around to try to find the right driver.

Once you think you’ve installed the driver then you simply need to verify that the OS can see it. Go to **Start** menu and **right-click Computer**, and then click **Manage**. Look for the serial device in the **Device Manager**.

### 4.3.2 Nexys 3

If you have a serial port on your computer, it doesn't matter, you can't use it! The Nexys 3 doesn't have a physical serial port, but instead emulates one. Those drivers should have been installed when you installed the Adept software, so you're good to go.

### 4.3.3 Finding the Port Name

Run the graphical ratload program. The main window of the program contains a dropdown which lists your available serial devices. If you are using a USB-to-serial adapter the easiest way to be totally sure of the correct COM port number is to unplug the adapter and note the adapters listed. Then plug in the adapter and click **File > Refresh Serial Devices**. The COM port that appears in the dropdown is the COM port of the adapter.

## 4.4 Ratload Procedure

1. Program your Nexys board with the generated bit file.
2. Without disconnecting anything else, connect the Nexys 2 board to your computer via the serial cable. If you have a Nexys 3 board, connect the second USB cable for the UART.
3. Start the ratload program. Select the proper serial device and prog\_rom.vhd file to read.
4. The program will then communicate with the Nexys board and send the prog\_rom.vhd to your RAT CPU via the serial connection. Once the program displays a success message, the serial cable can be disconnected and the board used normally. You can treat the program running on the board like it was synthesized with the system using the previous pattern, with one exception. For more information about this exception see Section 4.5.
5. To send a new prog\_rom.vhd file you must power cycle your Nexys board and repeat this procedure, starting with Step 1.

## 4.5 Interrupts and the ratload System

Ratload uses a UART to communicate with the host computer. The UART requires an interrupt, and the RAT CPU only has one interrupt. This creates a problem if you want to add I/O devices like a mouse, that also require an interrupt. A suggested solution is to OR the interrupts together.

Because the UART is only active during the programming stage, it will not trigger an ISR while your program is running. A more extensible solution would be to add multiple I/O interrupts, but for our purposes this is sufficient.

## 5 How It Works

The assembly programs that you create with RatSim and copy into your CPU are in fact RAMs that simply have the write strobe permanently tied low, hence the name `prog_rom`. When you synthesize your CPU with the `prog_rom` module in place, Xilinx is hard-coding your instructions into the “hardware” of the CPU. As you know very well by now, every time you want to modify your assembly program you have to re-synthesize the entire project.

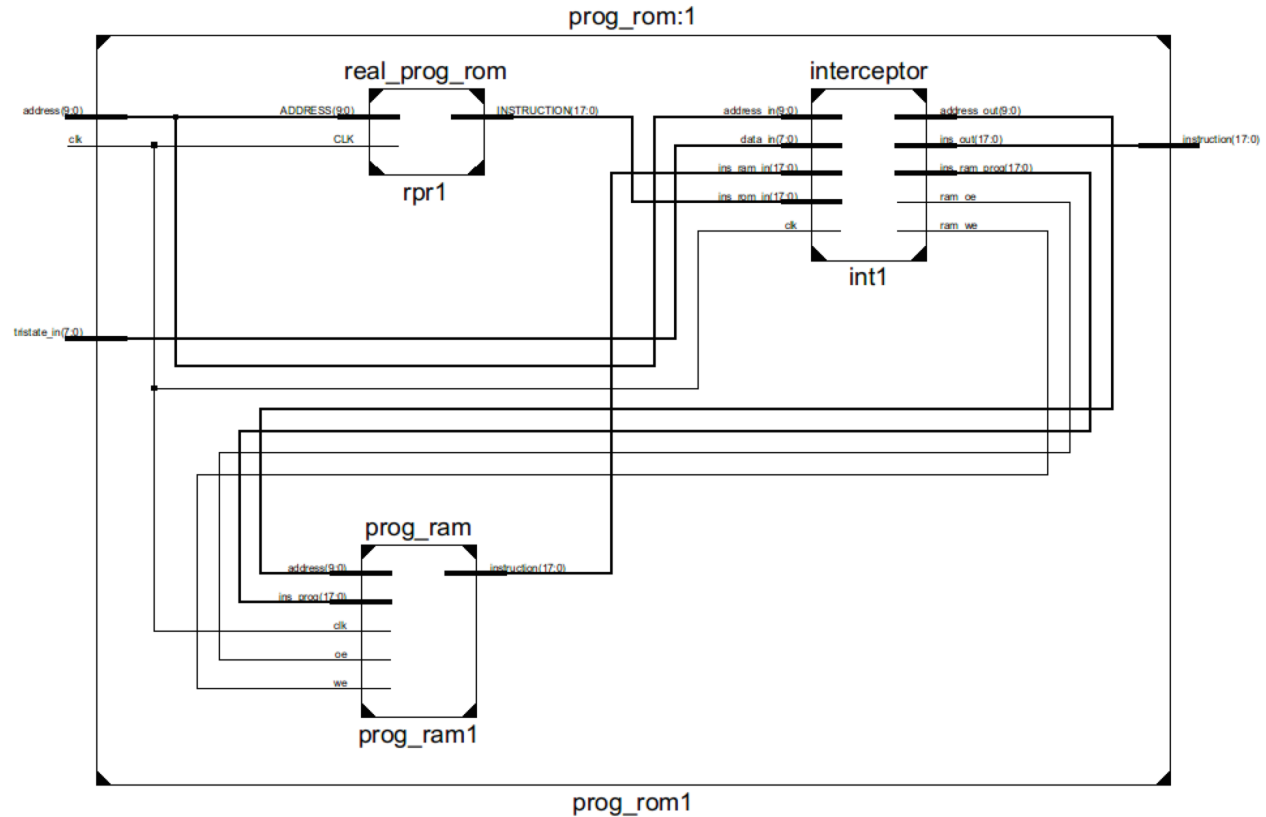


Figure 3: Prog\_rom Internal Operation

The solution to this problem presented by `ratload` uses a UART and a special `prog_rom` module. There are several other ways to reprogram the CPU without re-synthesis, but the UART+new `prog_rom` approach was chosen for two reasons. First, minimal modification of the CPU. The only change required to the RAT architecture is the addition of the tristate-, or multi-, bus to the `prog_rom` module. This way users already familiar with their CPUs will not be confused by architecture changes. Second, the simplicity of the UART. Other I/O options, such as USB or ethernet (available on the Nexys 3 board) require complex hardware drivers and have intricate handshaking procedures easy to mis-implement. UART is universal and dead simple.

Thus the project can be decomposed into two parts: the new `prog_rom` module and the UART. As mentioned, the UART is just another I/O device. The `prog_rom` contains the only complicated logic in the system. Figure 3 shows the internal operation of the new `prog_rom`. The updated `prog_rom` is just a wrapper — like the RAT wrapper — for three modules:

1. **real\_prog\_rom:** A “bootloader” assembly program. The structure of this module is identical to the `prog_rom` module you have been using so far.



2. **prog\_ram**: A RAM module.
3. **interceptor**: A state-machine that controls which module, either the prog\_ram or the real\_prog\_rom, feeds instructions to the CPU.

The CPU starts up and immediately starts running code in the real\_prog\_rom module. This code spin-waits until it receives an interrupt from the UART. The interrupt indicates that an instruction is available in the UART buffer. The “bootloader” program then issues a special assembly instruction called “LOADPR”. The interceptor sees this instruction and takes the data from the serial port buffer and writes it to the prog\_ram.

When 1024 instructions have been sent, the interceptor switches control of the CPU from the real\_prog\_rom to the prog\_ram. It resets the program counter and clears all interrupts and flags. Then execution of the new program starts.

## 6 Troubleshooting

### 6.1 Checking Your Serial Cable with the Reference Bit File

The following procedure will verify that your serial adapter is working and that the ratload program is working. Loosely speaking, this section tests the “host side” of the system, whereas subsequent sections test the “VHDL side” of the system.

1. Program your Nexys board with the “rat\_wrapper.bit” file included in the “testing” directory.
2. Connect your serial cable between the Nexys board and your computer.
3. Follow the steps in Section 2.2 starting from step 4.

### 6.2 Troubleshooting the UART Module

The easiest way to troubleshoot the UART module is to break out an actual serial console and see what it’s sending to the computer. So that’s what we’re going to do. You will need to know what your serial device is called on your computer. To do so see Section 4.3.

#### 6.2.1 Obtaining a Serial Console

A good serial console for Windows is RealTerm. Download and install it from here: <http://www.i2cchip.com/realterm/>

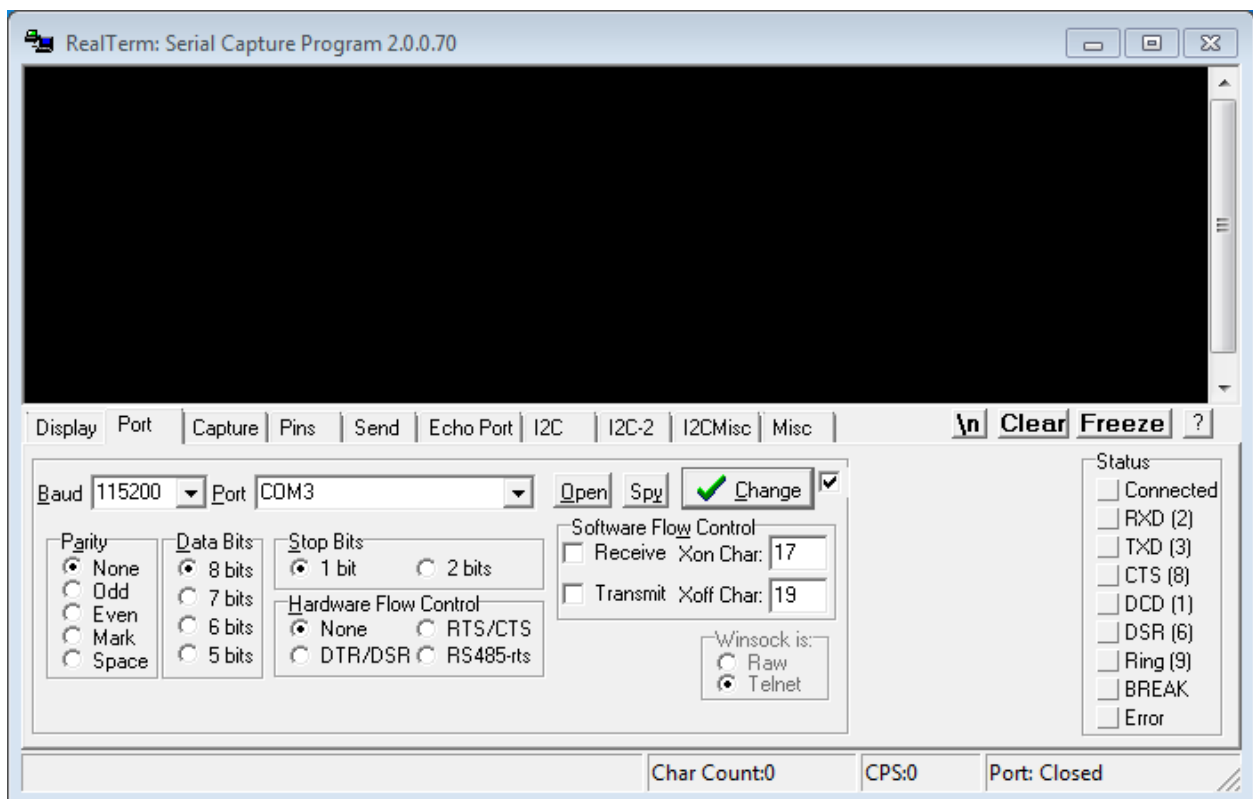


Figure 4: The correct settings to use with RealTerm

### 6.2.2 Configuring the Proper Serial Console Settings

Run RealTerm. Click on the **Port** tab. Make your RealTerm settings look like they do in Figure 4. Make sure to click **Change** to apply the settings. Remember that your COM port may differ.

### 6.2.3 Using the Serial Console to Troubleshoot the UART

Once you have your serial console up and running flash your Nexys board with your bit file that has the serial test program on it. (Make sure you've connected the Nexys board to your computer with the serial cable).

Send some data to the UART from the serial console. This is done by typing in the serial console. The behavior of the serial test program is as follows:

1. Get an ASCII-encoded byte from the host computer.
2. Convert that byte to binary
3. Display the byte on the LEDs on the Nexys board.
4. Re-encode the same byte to ASCII and send it back to the host computer.

You are looking for the following behavior when interacting with the serial test program:

1. Send 1 ASCII-encoded byte
2. Receive that same byte back.

If you receive two characters back, if you receive no characters back, if you receive a flood of characters back, or anything that is not that exactly what you typed, then you have integrated the RAT wrapper incorrectly.

## 6.3 Troubleshooting the Ratload Program

Ratload can fail in several different ways. This section lists all possible error messages, an explanation of each, and a suggestion on how to resolve the error.

- “File not found”, or similar: Ratload could not open your prog\_rom.vhd file. Perhaps it couldn't find it, or it didn't have permission.
- “Port not found”, or similar: Ratload could not open the serial device you specified. Perhaps you specified it incorrectly, or ratload does not have permission to access it.
- “Invalid prog\_rom.vhd file.”: Ratload was able to find and open the file you specified, but it couldn't parse it. Ratload expects the prog\_rom.vhd to be structured in a very specific way. Because this file is auto-generated by the ratsim program, this is not a problem. Make sure you are specifying the exact prog\_rom.vhd file that ratsim generates. If you continue to receive this error, try generating the prog\_rom.vhd file again.
- “Serial Configuration Failed”, or similar: Ratload was able to find and open the device you specified, but when it failed to configure it. It is possible but extremely unlikely that you have a serial device that does not support the proper settings. It is much more like that you specified a valid but incorrect device.
- “Handshake with Nexys board failed.”: Ratload was able to open initial communication with the Nexys board but received bad data. Get out the serial terminal and see what data the is actually being send out.
- “Received bad data from Nexys board.”: Ratload was able to open and parse the file you specified, and was able to open and configure the serial device you specified, but it received no or incorrect data from the Nexys board. Program the “reference\_rat\_wrapper.bit” file onto your Nexys board and try to send data to it with ratload. If that works, then you have misconfigured your RAT CPU, and you need to return to the integration section and make sure you followed those steps correctly.

- “Timeout.”: The serial port did not respond to the program in the time given. This is a very general error and could indicate a number of problems. The best course of action is to get out the serial terminal and try to communicate with the UART through there.