

This is a basic how-to on setting up an Internet-connected BBS using GBBS Pro and either a real or emulated Apple IIe or IIgs computer. For AppleWin users, the minimum required version is 1.26.3.0 – this version has an ACIA behavior fix that's needed for GBBS Pro (or any other BBS package) to operate properly. You can find the latest version of AppleWin here: <https://github.com/AppleWin/AppleWin>

In order to connect your real or virtual Apple IIe to the Internet, you're going to need to set up a "modem emulator". There's a number of different choices out there, but for this how-to I'm going to cover one called tcpser. What tcpser does is act as a bridge between the Internet and your Apple IIe (or other vintage computer). As far as your Apple IIe is concerned, it's talking to a real modem.

Setting up a host for tcpser is pretty simple. I'm going to cover two methods that I've used myself. Note that tcpser can be built for Linux, Windows, and pretty much any BSD-derived operating system (OpenBSD, FreeBSD, NetBSD, Darwin, etc.)

First will be a Raspberry Pi configuration and the second will be a Windows configuration.

For the Raspberry Pi option, you'll need the following items:

- Raspberry Pi 3
- USB to Serial adapter – units that use the Prolific PL-2303 chipset are preferred and are known to work well. You can search Amazon for "Prolific PL2303" to see some good examples from the likes of Sabrent and Trendnet.
- A powered USB 2.0 hub. The Raspberry Pi cannot provide the power that the USB to Serial adapter needs, so you'll have to plug it into the hub. The advantage here is that you can also power the Raspberry Pi from the HUB as well.
- A special cable called a "Null Modem". This is a special serial cable that is used to connect your Apple IIe to the USB to Serial adapter. I'll cover this in detail later.
- The tcpser software.

If you're using a Windows PC as the host, you can skip the Pi and the USB hub (unless of course you're out of USB ports on your PC!)

The Null Modem Cable

In order to connect your Apple IIe to the host running tcpser, you're going to need a null modem cable. This is a special cable that is designed to allow two computers to communicate over a serial link. This special cable is needed because the connection on both the host serial end and Apple IIe end is known as "DTE" or Data Terminal Equipment. When you're trying to get a serial link going DTE to DTE, you've got to swap some wires around in the cable. This is not the case when going from a DTE device, to a DCE (Data Communication Equipment) device like a modem.

The serial connector on the Apple end is typically a 25 pin connector and the connector on the PC or Pi end is often a 9 pin connector. The diagram on the right shows how the wires are connected between the two to make a null modem cable. You can use the chart on the right to build your own cable if you like.

DE9 to DB25 Null Modem Cable			
DE9 Pin #	Name	Name	DB25 Pin #
8	CTS	RTS	4
7	RTS	CTS	5
3	TXD	RXD	3
2	RXD	TXD	2
4	DTR	DCD	8
1	DCD	DTR	20
5	GND	GND	7

Cables of this type are easily found at online retailers such as Amazon (StarTech SCNM925FM). Make sure that the cable you're buying is a "full" null modem cable with a pinout that matches the chart on the previous page, otherwise you could have issues with it working properly.

Configuring the Super Serial Card

On the Apple II side, you should set up the Super Serial Card like so:

Set the jumper block with the triangle on it so that the tip of the triangle points towards "MODEM".

SW1 is the first switch block, SW2 is the second.

Set SW1-5 and SW1-6 to on for "communications mode".

Set SW1, switches 1-4 to: off on off on - this sets 2400 baud.

Set SW2-1 to on for 1 stop bit

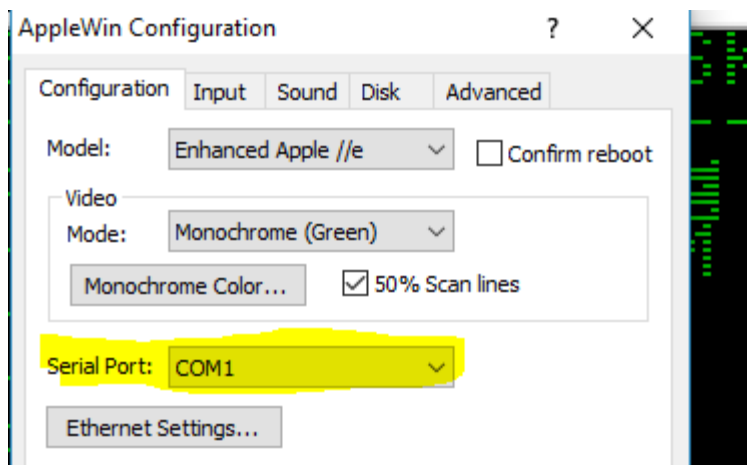
Set SW2-2 to on for 8 bits

Set SW2-4 to on for no parity.

Set SW2-6 to on to enable interrupts (set to off if you're using a II or II+)

Configuring AppleWin

In order to use tcpser with AppleWin, you'll need to update the AppleWin configuration with the ID of the serial port you're going to be using.



You'll also need to start AppleWin with the "-modem" command line switch in order to enable proper behavior of the ACIA on the emulated Super Serial Card within AppleWin.

Getting and Installing tcpser for Windows

For those that plan on using Windows as their tcpser host, I've created a pre-compiled binary that you can download here:

http://www.geneb.org/tcpser/tcpser_bin.zip

The zip file contains the tcpser.exe program as well as a pair of DLLs that are required for it to function. Make sure that you keep the two included DLLs in the same directory that tcpser.exe is in, otherwise the program may fail to work.

If you're going the Linux or Raspberry Pi route, read on!

Getting and Installing tcpser for the Raspberry Pi

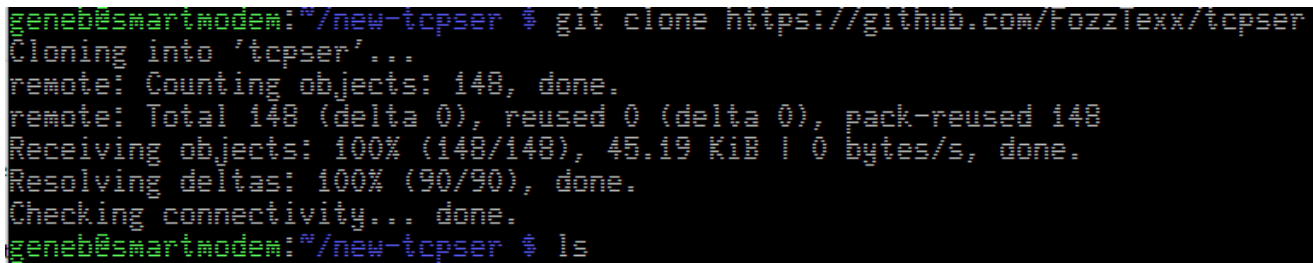
In order to get tcpser installed on your Raspberry Pi, you're going to have to grab the source code out of the Git repository where it lives. If you don't have Git or the C compiler installed on your Raspberry Pi, you'll need to get those installed first – Google is your friend here since that's way outside the scope of this document.

Obtaining the source code for tcpser is very simple – you simply “clone” into the Git repository where it's stored. The Git repository you're going to clone into is here:

<https://github.com/FoizzTexx/tcpser>

The command to obtain the software is:

```
git clone http://github.com/FoizzTexx/tcpser
```



```
geneb@smartmodem:~/new-tcpser $ git clone https://github.com/FoizzTexx/tcpser
Cloning into 'tcpser'...
remote: Counting objects: 148, done.
remote: Total 148 (delta 0), reused 0 (delta 0), pack-reused 148
Receiving objects: 100% (148/148), 45.19 KiB | 0 bytes/s, done.
Resolving deltas: 100% (90/90), done.
Checking connectivity... done.
geneb@smartmodem:~/new-tcpser $ ls
```

Obtaining tcpser via git.

Once the clone is finished, you can change to the tcpser directory and type “make” to compile the program. When the compile finishes, you're ready to go!

Running tcpser for your BBS

You'll need to identify what serial device you want tcpser to talk to while it's running. On a Raspberry Pi with a single USB serial adapter, it will be called “/dev/ttyUSB0”. On a Windows system with a serial port at COM1, it will be called “/dev/ttyS0”. Serial ports referenced by tcpser are counted at from zero on Windows, so /dev/ttyS1 corresponds to COM2, /dev/ttyS2 corresponds to COM3, etc.

Decide what telnet port you want tcpser to watch for connections. 23 is the standard telnet port, but I recommend using something like 2300 or 6502. The reason for this is that if your BBS is on a non-standard port, it will prevent (most) people running port scanners and disturbing your BBS.

Chose a baud rate for your BBS that's the fastest that your BBS software can reliably support. In the case of GBBS Pro v1.3 and earlier, the driver appears to only support up to 2400 baud. This is unsurprising considering what modem the average user had available to them in 1986. GBBS Pro 2.2n (and possibly others after 1.3j) support baud rates up to 38,400. Because tcpser runs at a fixed baud rate, you'll need to configure the BBS to run at the rate you've chosen for tcpser. If you notice characters are being dropped at 38,400, slow it down and test again.

Available baud rates for tcpser are 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200.

Create a short text file that will be sent to a caller when they try to connect and someone is already on the BBS. This is known as the "busy" file and should let the user know that the system is busy and that they should try connecting again at a later time.

Once you've gotten tcpser built or downloaded, you can run it with the following options:

```
tcpser -d /dev/ttyUSB0 -s 2400 -p 2300 -B busy-msg.txt -i "X3S0=1&C1&D2S2=128"
```

- d /dev/ttyUSB0 tells tcpser what device it's going to use.
- s 2400 tells tcpser to set the port baud rate to 2400.
- p 2300 tells tcpser to listen on port 2300 for incoming connections.
- B busy-msg.txt tells tcpser what the name of the "busy file" is.
- i "X3S0=1&C1&D2S2=128" configures the virtual modem inside of tcpser:
 - X3 - enables busy signal detection
 - S0=1 - Set to auto answer
 - &C1 - Enable DCD after carrier detected
 - &D2 - Dropping DTR causes connection to be closed.
 - S2=128 - Sets the modem "escape" character to ASCII 128.

On the GBBS side, you'll configure the system for a Hayes 2400 baud modem. Note that if your version of GBBS can support baud rates higher than 2400, there's no reason you can't use those higher rates with tcpser. Just specify that faster rate the same way you did with 2400 baud.

The simplest way to run tcpser is via a script (Linux/Unix) or batch file (Windows).

For Linux:

```
/path/to/tcpser -d /dev/ttyUSB0 -s 2400 -p 2300 -B busy-msg.txt  
-i "X3S0=1&C1&D2S2=128"
```

For Windows:

```
X:\path\to\tcpser -d /dev/ttyS0 -s 2400 -p 2300 -B busy-msg.txt  
-i "X3S0=1&C1&D2S2=128"
```

Save the relevant command line to a file – there's your script or batch file! Keep in mind that under Linux, you'll need to start the script with "sudo" in order for tcpser to have permission to access the serial device on your system. ex.:

```
"sudo runbbs.sh"
```

Note that you'll need to mark the script as executable by issuing "chmod +x script-name.sh". This is not necessary in Windows.