

RS232 File Sender and Receiver for Commodore Plus/4 by GMOLN

Introduction

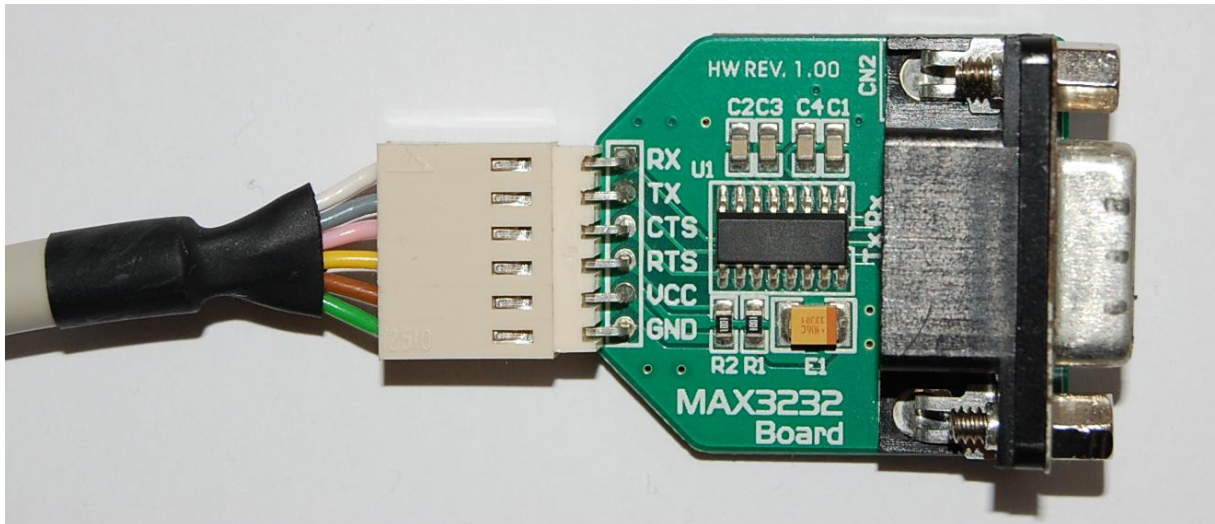
The purpose of these applications is to send files from Java capable computers with serial port to Commodore Plus/4 via standard RS232 link. The main goal was to provide a GUI based, up to date and easy to use toolset without the use of any special and often overcomplicated hardware or cable.

Hardware

The required user port / RS232 cable is as simple as possible.



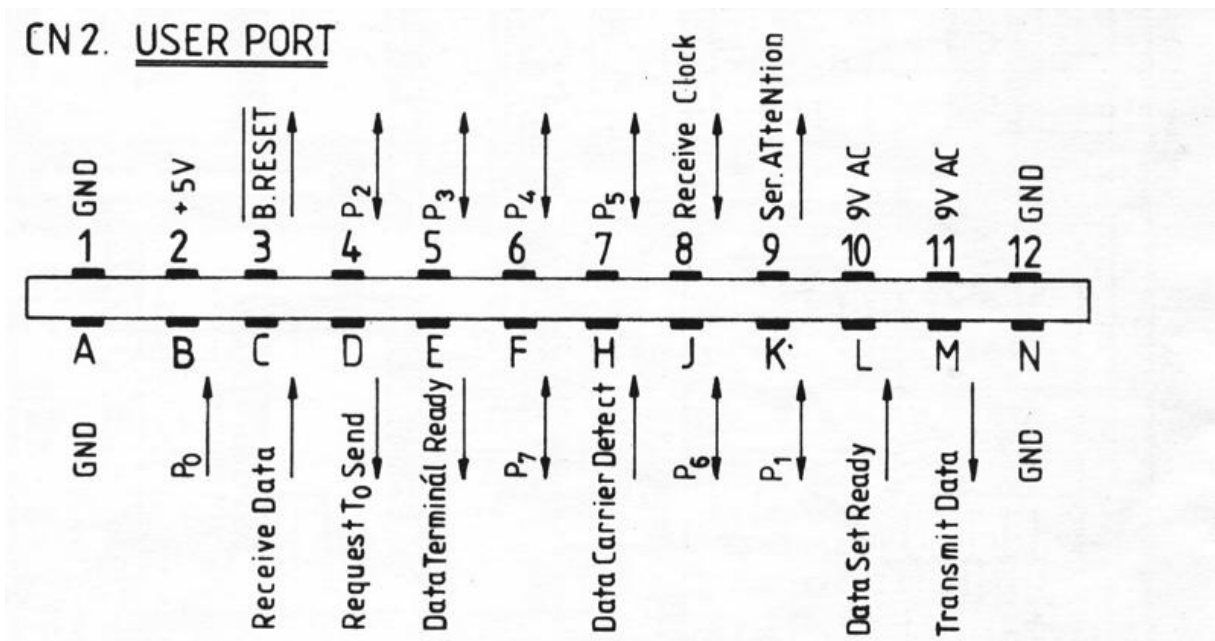
It is based on a 24 pin Commodore User Port connector which can be purchased from various sources on eBay. Because the **6551A ACIA** in the Commodore Plus/4 is working with **TTL** levels, a further RS232 level shifter is required.

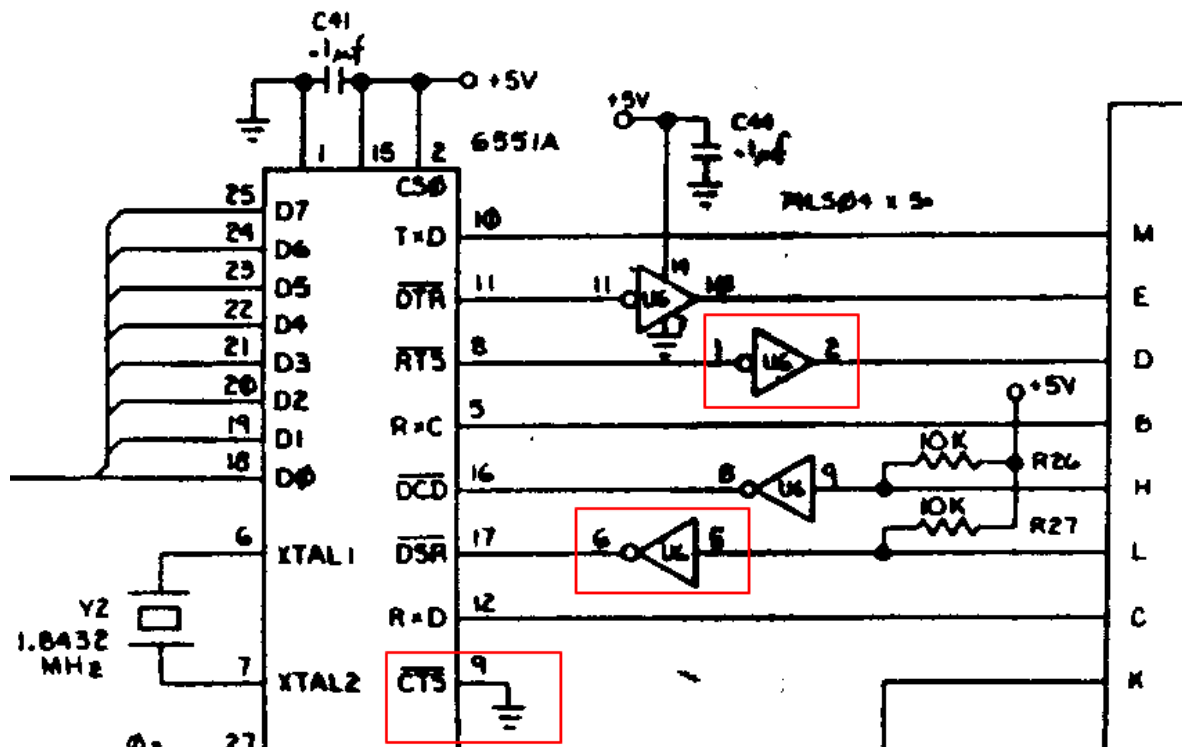


In this example, a [MAX3232](#) board is used from **Mikroelektronika**, but any other RS232 level shifter with **hardware flow control** capabilities can be used with **MAX (3)232** chips. The connections between the **User Port** connector and the **MAX3232** board should be the following:

User Port pin	MAX3232 board pin
2	VCC (+5V)
A	GND
C	RX
D	RTS
L	CTS
M	TX

User port pin out can be found below:





Because of the strange hardware implementation of the serial port in the Commodore Plus/4, the **DSR** pin of the **6551A** is used to handle **CTS** signal. Furthermore, all hardware flow control signals are inverted, but the serial port driver of the receiver program on the Commodore Plus/4 can also handle this.

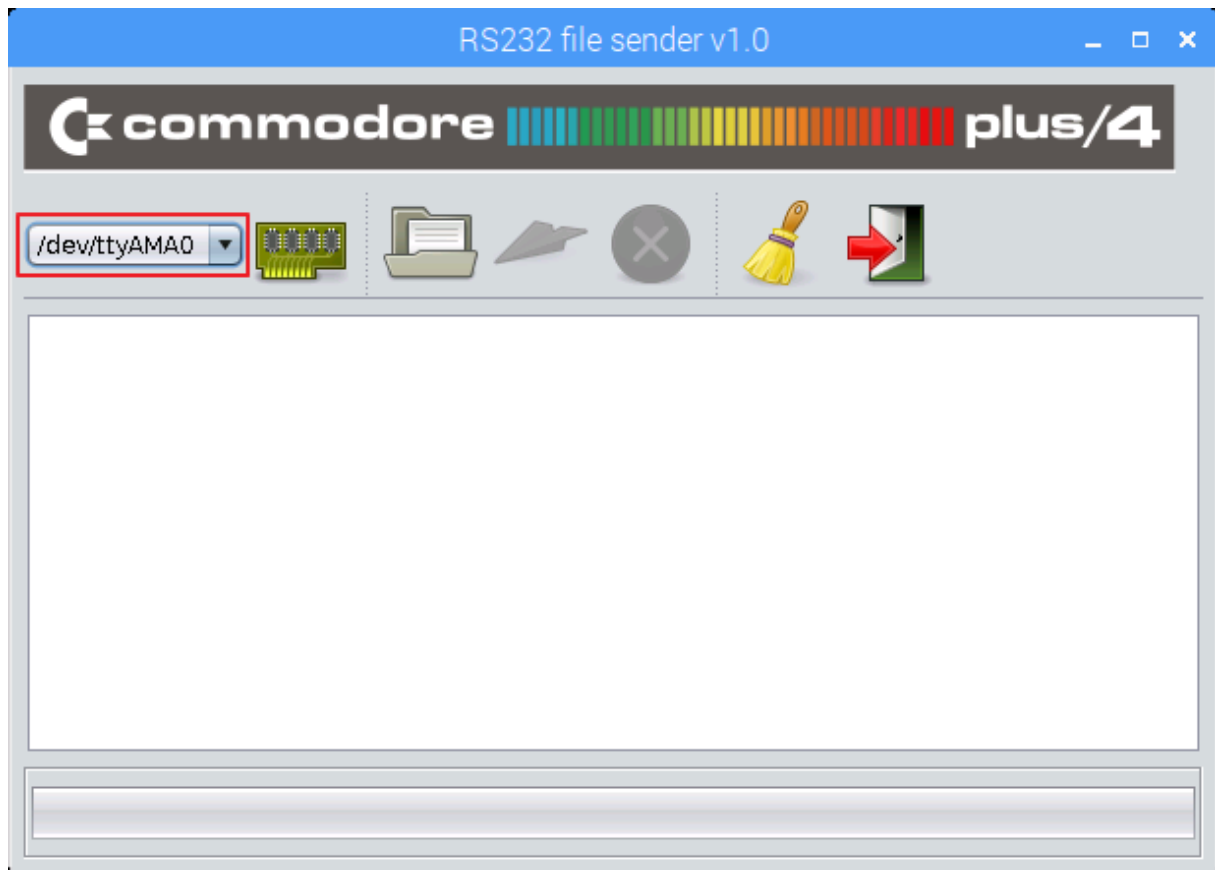
!!!IMPOTANT!!! The RS232 cable between the two computers must have proper connections for the hardware flow control (CTS/RTS) lines! Otherwise this toolset will not work! Basic RS232 cables, where only the TX, RX and GND pins are connected not suitable for this purpose!

RS232 File Sender v1.1

This application is intended to select and send files to Commodore Plus/4 via the serial port. It is a multi threaded, pure Java application, requires **JRE 1.8** to run. The embedded **jSSC** Java serial port library can run on Windows, Linux (x86 and ARM), Mac OSX operating systems. Nevertheless, it was tested only on **Windows 7 & 10** with **Oracle JRE 1.8** and on **Raspbian** (Jessie on Raspberry Pi 2) also with **Oracle JRE 1.8**.

Special instructions for Raspbian (Raspberry Pi)

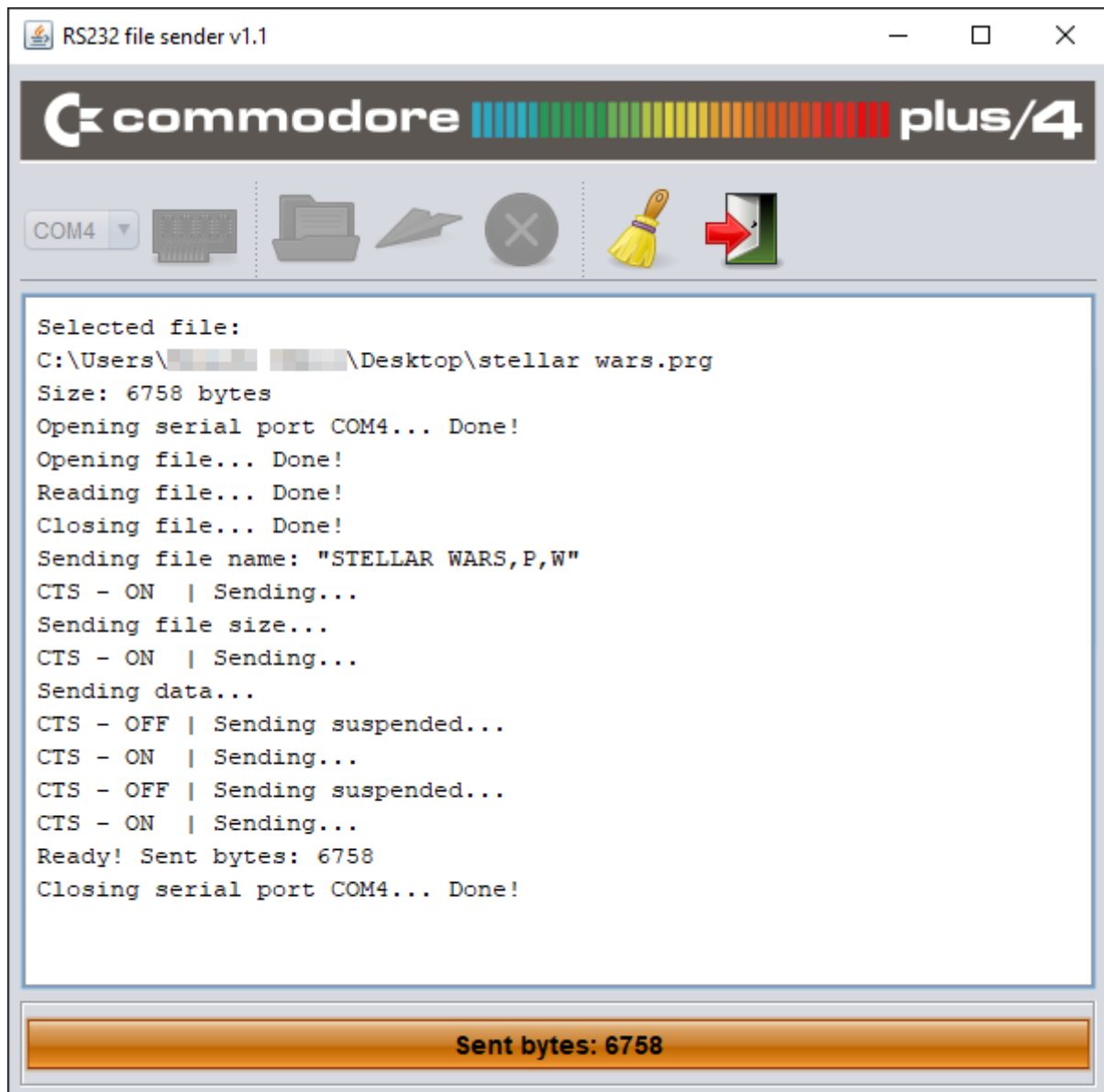
In case of **Raspbian**, the default serial console on the built in serial port needs to be disabled first with **raspi-config** utility. Then the **GPIO** pins need to be reconfigured with **rpiwtscts** utility to provide **RTS/CTS** hardware flow control lines. Finally, the serial port needs to be forced to use hardware flow control with command **sudo stty -F /dev/ttyAMA0 crtscts**. The Java application must be started as **root**, otherwise it is not able to access the **UART** of the Raspberry Pi. In case of any X host related starting issues, the **xhost +** command might be issued before the starting of the File Sender application.



It sounds complicated, but a small start script can be created with the above steps. The red marked serial port is the built in serial port of the Raspberry Pi and its pins are available on the GPIO header. Please don't forget the additional RS232 level shifter for the Raspberry Pi in this case. It works with **3.3V** signal levels, therefore only the **MAX3232** chip based level shifters can be used here.

How to use File Sender application

The usage of the file sender application is quite simple and straightforward.



Available serial ports can be selected from the drop down list. If an USB serial port is attached after the start of the application, then the button next to the drop down list can be used to rescan for serial ports.

In the first step, the file which needs to be sent should be selected. The application will automatically convert the file name to upper case. It will use only the first 12 characters and put **,P,W** at the end. The file receiver on the Commodore Plus/4 will save the file as **PRG** file. Please do not use any special characters in the file name. The **ASCII** to **PETSCII** conversion is limited only to numbers and letters.

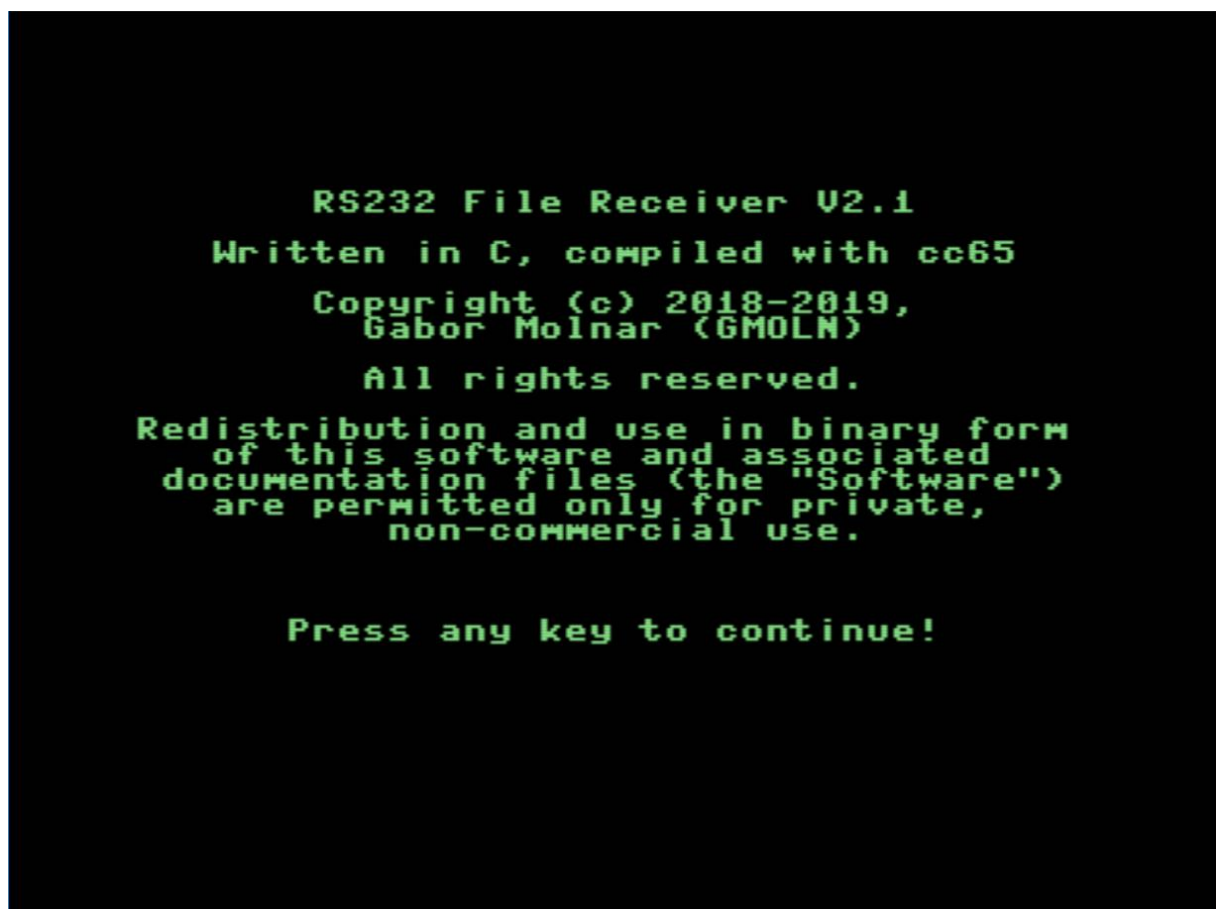
The file transfer can be started with the send button. Please note that the **File Receiver** on the Commodore Plus/4 must be started and ready before sending files to it. The application will open the serial port, open and read the whole file into the memory. In the subsequent steps, it will send the file name, file size in bytes and finally the binary file content to the receiver on Commodore Plus/4. When all data were sent, it will close the serial port. New transmission can be initiated with the reset button on the toolbar. The current file transfer can be stopped with the cancel button, but it will not stop the receiver on the Commodore Plus/4. It needs to be stopped manually on that machine. The File Sender doesn't check the file size regarding to the free space. It is limited only by the free space available on device 8 on the Commodore Plus/4.

If the File Sender application stuck at file name sending, then it is an indication about issues with the hardware flow control lines. In this case all of the above connections and the serial cable need to be double checked.

For technical details about the RS232 settings and used protocol, please refer to the next chapter.

RS232 File Receiver V2.1

This program is intended to receive files, sent by the **File Sender** application, on the Commodore Plus/4 via the built in serial port. The program was written in **C** and compiled with **cc65**. The custom **6551** driver with full-blown hardware flow control capabilities was written in assembly and complied with **ca65**. The whole program is compressed with Time Crunch V5.0.



Technical informations

RS232 settings:

- 9600 baud, 8 data bits, No parity
- 1 stop bit, CTS/RTS HW Flow Control

Data structure:

- The first max. 16 bytes contain the PETSCII encoded file name closed with 0x00
- The next 4 bytes contain the file size (big endian byte order)
- Following bytes contain the data

Application decodes the file name and creates it on device 8. Receiving of data stops automatically when no more bytes remained to receive. Internal RX buffer size is 4096 bytes. Application switches off the screen during RX, to be able to continuously process the incoming data with 9600 bps.

Press any key to continue!

Please select:

- 1 - Receive file
- 2 - Quit
- 3 - Reset

Receiving file can be started with pressing button one.

Serial port settings:

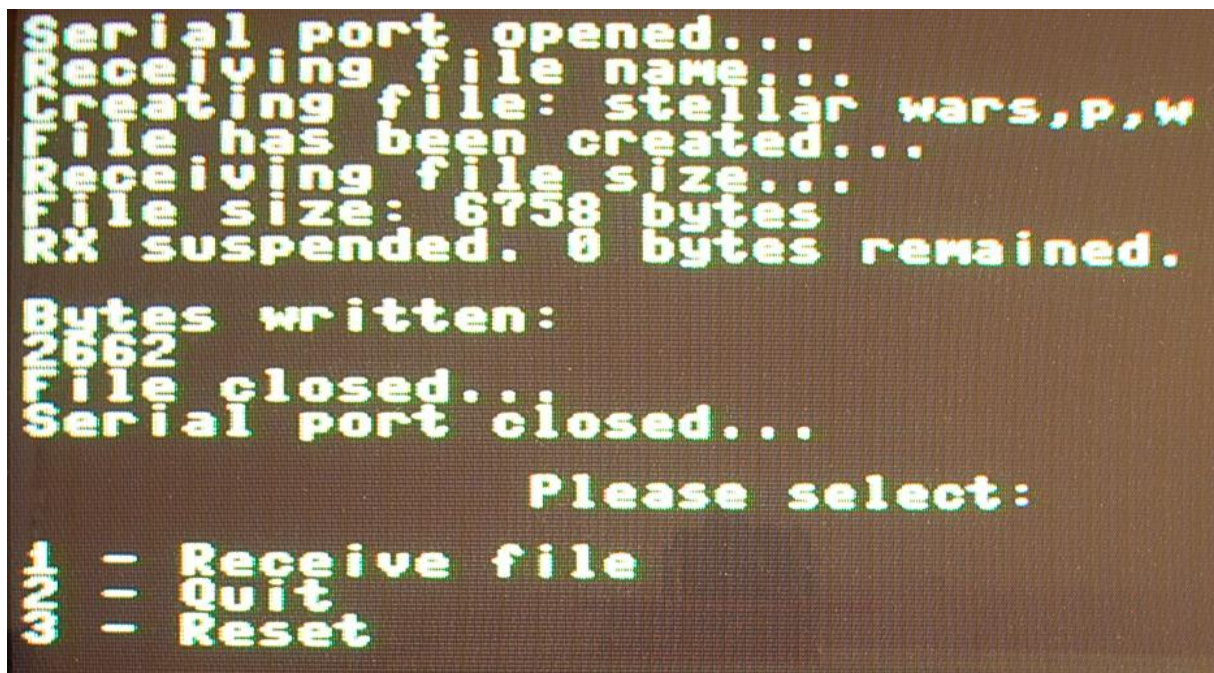
- 9600 baud
- 8 data bits
- No parity
- 1 stop bit
- CTS/RTS HW Flow Control

These settings are “hard wired” into both sender and receiver.

Protocol:

- First maximum 16 bytes contain the PETSCII encoded file name closed with 0x00
- Next 4 bytes contain the file size in big endian byte order
- Following bytes contain the binary data

The program decodes the file name and creates it on **device 8** regardless of the free space or file size. It switches off the screen during **RX**, to be able to continuously copy the incoming data into the **RX** buffer with **9600 bit per second**. Internal **RX** buffer size is **4096 bytes**. When it gets full, then the program stops the sender via the **RTS** signal (which is connected to the **CTS** pin on the other side) and switches on the screen. The buffer content will be written into the file with **KERNAL CHROUT** routine, which is quite slow. When the buffer gets empty, the program starts the sender via the **RTS** line and switches off the screen. Receiving of data stops automatically when no more bytes remained to receive and all received bytes were written into the file.



```
Serial port opened...
Receiving file name...
Creating file: stellar wars,p,w
File has been created...
Receiving file size...
File size: 6758 bytes
RX suspended. 0 bytes remained.

Bytes written:
2662
File closed...
Serial port closed...

Please select:
1 - Receive file
2 - Quit
3 - Reset
```

In the current version, the file transmission can be aborted only with the Reset button of the Commodore Plus/4 with its consequences, like not properly closed file, etc.

Copyright

Copyright (c) 2018-2019, Gabor Molnar (GMOLN)

All rights reserved.

Redistribution and use in binary forms of these softwares and associated documentation files (the "Software") are permitted only for private, non-commercial use.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.