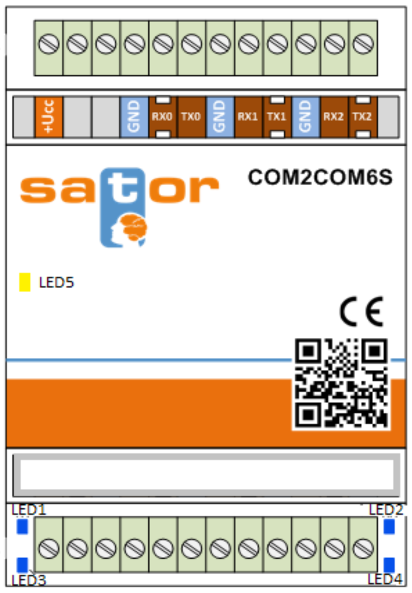
# Per-RS232 translator (hw COM2COM6) [ENG]

RS232-RS232 interface for controlling a "new" device with "old" RS232 codes - translates codes between old and new devices.



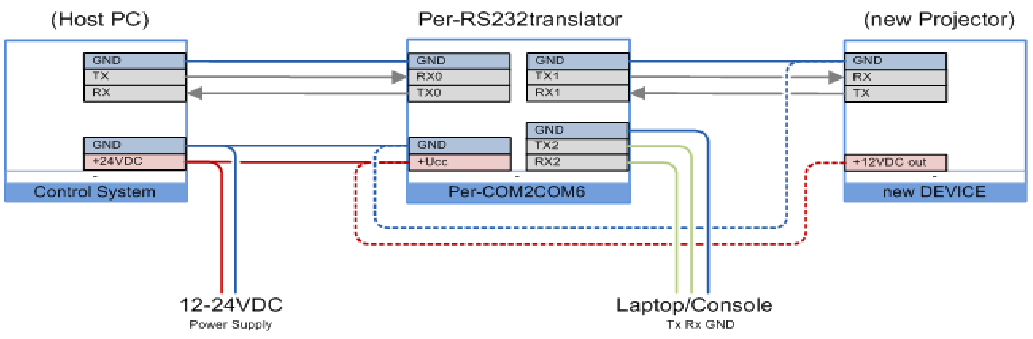
RS232-RS232 interface for controlling a "new" device with "old" RS232 codes; it performs a translation using a user-entered table of RS232 codes for the **original device (PORT0)** and RS232 codes for the **new device (PORT1)**, including parameters (baud rate / parity) of both ports. User can enter up to 8 different control codes for each port with a maximum length of 55 characters, both for the Rx (receiving control commands) and for Tx (sending control commands or confirmation responses).
The Interface can be also used as a Baudrate Converter.
The **system port (PORT2)** is used to set up the interface and to test communication to the connected devices, using common ASCII text commands and messages sent from a hyperterminal or similar command-line application.
Incoming, outgoing and system communication is indicated by status LEDs. The data are stored at internal EEPROM.

A typical example of using the interface is the replacement of a projector connected via RS232 to an (unknown) control system, with a new projector using another control codes - without a need to modify the original program or control system settings. Therefore, the control system (HostPC) sends commands to the original device, the interface receives theese commands and sends the relevant translated commands, stored in the control table (CtrlTable) to the new projector.
Interface also allows you to capture acknowledgment responses and send it to the control system if required by its program, or to simulate these acknowledgment responses (Autoresponse mode). The same apply for a new device, including simulating responses that the new device may require, but the control system does not send them in its current configuration.

Passthru mode (ver. 1.4>) determines whether commands coming to a port that are not defined in CtrlTable will be discarded (pass = 0, default setting), or whether they will be forwarded unchanged to the other device (pass = 1). Incoming commands that match the stored strings in CtrlTable are always translated and sent to the other device. Passthru setting applies for both directions of communication. This mode may be suited for eg. upgrading a projector of the same manufacturer, which has the same commands as the original, with only few new inputs added. At this case is sufficient to activate Passthru mode (pass = 1) and write into CtrlTable only cmds for new inputs (typically HDMI1 instead of VGA1). Other commands (ON, OFF, MENU, ASPECT, etc.) in this mode will pass to the new device unchanged.
From a practical point of view, if CtrlTable is empty (can be deleted with the !init command) and Passthru mode is active (pass = 1), the interface behaves like a usual Baudrate-Converter.

**Specifications/HW:**
PCB: COM2COM6S V1 R003
Dimensions: 105x53x60mm, DIN module, 3MOD width.
Power Supply: 12-24VDC / typ. 50mA, pins +Ucc, GND
RS232-PORT0 (ControlSystem / HostPC): Rx0,Tx0,GND (no RTS/CTS support); baudrate 9600/19200/38400/57600/115200,
parity No/Even/Odd, 8-dataBit, 1-stopBit, (default 9600,N,8,1), timeout 200ms
RS232-PORT1 (new controlled DEVICE): Rx1,Tx1,GND (no RTS/CTS support); baudrate 9600/19200/38400/57600/115200,
parity No/Even/Odd, 8-dataBit, 1-stopBit, (default 9600,N,8,1), timeout 200ms
RS232-PORT2 (system settings PORT): Rx2,Tx2,GND (no RTS/CTS support); always 9600,N,8,1, entry timeout 4s
LED1-4: activity on Rx0 (LED1), activity on Tx1 (LED2), activity on Rx1 (LED3), activity on Tx0 (LED4)
LED-indication: short blink: ctrl command found at CtrlTable (incomming string at Rx), or translated command sent (outcomming string at Tx); bold blink: ctrl string not found at CtrlTable, end of reception by timeout
LED5: on - entering user data or commands on the syst. port (PORT2), goes out when the entry is completed or by timeout

**Typical HW connections:**
HostPC or ControlSystem is typically connected to PORT0 of the interface, the new controlled device (DEVICE) to PORT1:



The TX-HostPC (original ctrl commands) goes to Rx0; RX-HostPC receives data (original ack messages) from Tx0. RX-DEVICE input receives data (new ctrl commands) from Tx1, TX-DEVICE data (new ack  messages) goes do Rx1.
Service PC/Laptop with text console is connected to the system port PORT2 of the interface.

**Functionality and system settings:**
Settings and communication are performed by text commands sent, for example from the PC application as Hyperterminal, HerculesSETUP, etc. to PORT2, which is permanently set with parameters 9600, N, 8,1 (no RTS / CTS :-)
After the interface is switched on or reset, LEDs light up for 1 sec, and if ServicePC is connected, the initial message with port settings, reading data from EEPROM to CtrlTable status (....) and prompt (>) is displayed in the hyperterminal window:

```
port0: 19200,E,8,1
port1: 9600,N,8,1
. . . .
>help
>
```

Afterwards, the interface receives and evaluates incoming commands (ctrl strings) on Rx0 or Rx1, and sends translated strings to Tx1 or Tx0.
The interface separates incoming commands to Rx0 / Rx1 after 200ms (timeout), ie. HostPC should not send commands faster in a row. However, if the commands are valid (stored in CtrlTable), then they are translated and sent immediately, thus sending could be done at this case several times within 200ms.
The evaluation of the ctrl string is performed with the stored commands in CtrlTable for a left-most match of characters; for example if the command "HELLO" is stored in CtrlTable, then there is a match (and translation with possible sending of the related string to DEVICE) both if HostPC sends "HELLO" or "HELLO123".

By displaying the ">" (prompt) character, the interface is ready for entering system commands from ServicePC. All commands on the service PORT2 are terminated by pressing the <Ent> key (ie. the double character \x0D\x0A, or only one character \x0D is ok). Entering is also terminated if delay between characters is greater than 4 sec.

**System commands:**
Interface settings are performed using system commands entered from the hyperterminal (or similar) of ServicePC connected to PORT2. Status LED5 lights up during entry:

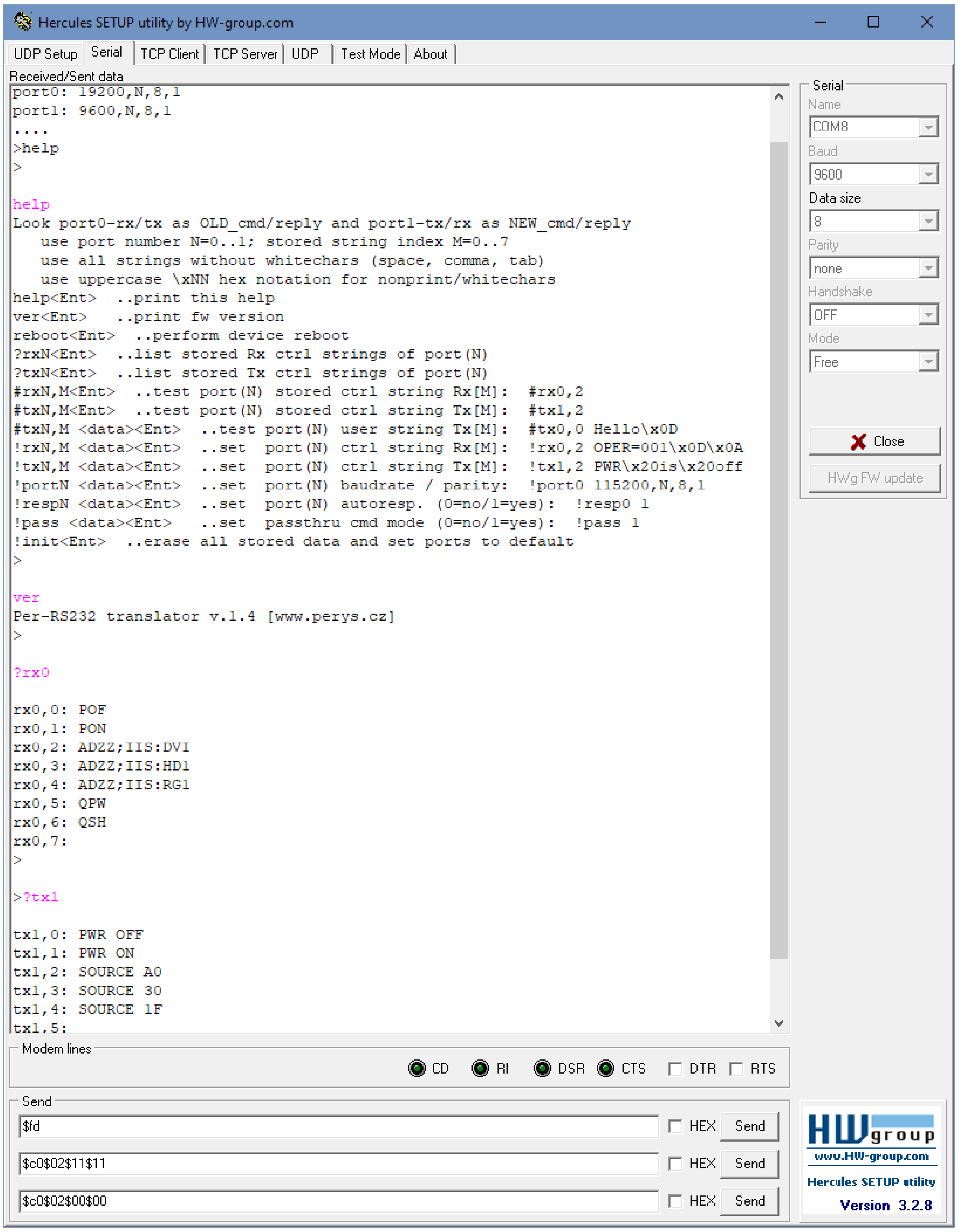
***help<Ent>***
displays a list of interface commands.
***ver<Ent>***
displays the interface firmware version.
***reboot<Ent>***
reboots the interface. Displays the current settings for PORT0 and PORT1, and all data is read from the EEPROM.
***!init<Ent>***
clears all user data in CtrlTable and EEPROM, sets Port0 and Port1 to default and reboots.
***!pass<Ent>***
lists the Passthru mode status.
*!pass<Ent> ... lists the current status of Passthru mode (0 = inactive, 1 = active)*
***!pass <data><Ent>***
reads and saves the value for the Passthru mode status (0 or 1) into EEPROM.
*!pass 1<Ent> ... activates Passthru mode.*
***!portN<Ent>***
lists the communication parameters for the given port; N is the port number (0 or 1).
*!port0<Ent> ... lists the communication parameters of PORT0*
***!portN  <data><Ent>***
reads and saves communication parameters for a given port; N is the port number (0 or 1) and displays the current port setting. the parameters entered in the <data> field must be one of the supported baud rates and the parity specified as N, E or O; other parameters are not supported and are permanently set to 8-dataBit, 1-stopBit.
*!port0  19200,E,8,1<Ent> ... set parameters of PORT0 to 19200,E,8,1*
*!port1  19200,O,7,2 <Ent> ... set parameters of PORT1 to 19200,O,8,1*
*!port0  38400,N <Ent> ... set parameters of PORT0 to 38400,N,8,1*
*!port0  9600 <Ent> ... changes baudrate of PORT0 to 9600, others remains unchanged*
*!port0  0,N <Ent> ... changes parity of PORT0 to N, others remains unchanged*

***?rxN<Ent>***
lists all ctrl strings stored in the CtrlTable for the Rx direction of the given port; N is the port number (0 or 1).
*?rx0<Ent> ... lists ctrl strings for Rx0 (PORT0)*
***?txN<Ent>***
lists all ctrl strings stored in the CtrlTable for the Tx direction of the given port; N is the port number (0 or 1).
*?tx1<Ent> ... lists ctrl strings for Tx1 (PORT1)*
***!rxN,M<Ent>***
prints ctrl string read and stored in CtrlTable; N is the port number (0 or 1), M is the index of the read ctrl string (0 to 7).
*!rx0,0<Ent> ... ctrl string for Rx0 (PORT0), position 0 v CtrlTable*
*!rx1,7<Ent> ... ctrl string for Rx1 (PORT1), position 7 v CtrlTable*
***!rxN,M <data><Ent>***
reads and stores ctrl string specified in the <data> field to CtrlTable; N is the port number (0 or 1), M is the index/posit. (0 to 7); the specified (user's) string can have up to 55 characters, must be entered as a set of ASCII characters without whitechars and can only contain printable characters; ASCII "upercase hex \xNN notatoin" must be used for non-printable characters, spaces and control characters.
*!rx0,0 Hello123<Ent> ... ctrl string for Rx0 (PORT0), position 0, will contain: Hello123*
*!rx0,1 Hello123\x0D\x0A<Ent> ... ctrl string for Rx0 (PORT0), position 1, will contain: Hello123<0Dh><0Ah>*
*!rx0,2 \x01\x30\x31\x00\x0D<Ent> ... ctrl string for Rx0 (PORT0), position 2, will contain: <01h>01<00h><0Dh>*
*!rx1,7 Hello\x20Hello\x20123<Ent> ... ctrl string for Rx1 (PORT1), position 7, will contain: Hello Hello 123*
***!txN,M<Ent>***
the same as *!rxN,M<Ent>* ... related to Tx.
***!txN,M <data><Ent>***
the same as*!rxN,M <data><Ent>* ... related to Tx.
***!respN <data><Ent>***
sets and prints the Autoresponse mode status for ctrl strings in the CtrlTable for the given port; N is the port number (0 or 1). the value in the <data> field is either 1 or 0 (yes/no); if Autoresponse mode for eg. PORT0 is on, then when receiving the ctrl string read in CtrlTable eg. at Rx0,7, the relevant string is sent back (with the same index) stored here at Tx0,7 (or empty string /no reply/ to Tx0 if no string is stored at that position).
*!resp0 1<Ent> ... switches Autoresponse mode on for PORT0*
*!resp1 0<Ent> ... switches Autoresponse mode off for PORT1*
***#rxN,M<Ent>***
prints the ctrl string read and stored in the CtrlTable for the given port and sends the relevant (with the same index) ctrl string to the Tx pin of the opposite port, for testing the response of connected device (or to check the ctrl string syntax); if the Autoresponse mode is active for that port, stored relevant string is also sent to the Tx pin of the same port. When this command is entered, the interface behaves as if it receives a valid (stored) ctrl string from the connected device on the given Rx pin.. N is the port number (0 or 1), M is the index/position of the read ctrl string (0 to 7).
*#rx0,2<Ent> ... ctrl string for Rx0 (PORT0), position 2 at CtrlTable is translated and sent the string stored at tx1,2 to pin Tx1 (PORT1); if the Autoresponse mode of the port0 is on, then the string tx0,2 is being sent to Tx0 (PORT0) as well.*
***#txN,M<Ent>***
prints the ctrl string read and stored in the CtrlTable for the given port and sends it to Tx pin, for testing a response of connected device (or to check the ctrl string syntax). N is the port number (0 or 1), M is the index of the read ctrl string (0 to 7).
*#tx1,2<Ent> ... ctrl string for Tx1 (PORT1), position 2 v CtrlTable is sent to Tx1 (PORT1)*
***#txN,M <data><Ent>***
allows to enter a user string for a given port and sends it directly as a ctrl string to the Tx pin of the given port, for testing the response of connected device (or to check the ctrl string syntax). N is the port number (0 or 1), M is the index, but it is irrelevant in this command. The specified (user's) string can have up to 55 characters, must be entered as a set of ASCII characters without whitechars and can only contain printable characters; ASCII "upercase hex \xNN notatoin" must be used for non-printable characters, spaces and control characters.
*#tx1,0 Hello\x20Hello\x0D<Ent> ... ctrl string sent to Tx1 (PORT1) will contain: Hello Hello<0Dh>*

For other special HW requirements than the interface configurations mentioned here, or for initial control strings entry to the ControlTable and test, please ask for thos before the order.

Preview of system communication could be seen at the figure below.

5/2019
www.perys.cz



Received/Sent data

```
port0: 19200,N,8,1
port1: 9600,N,8,1
....
>help
>

help
Look port0-rx/tx as OLD_cmd/reply and port1-tx/rx as NEW_cmd/reply
   use port number N=0..1; stored string index M=0..7
   use all strings without whitechars (space, comma, tab)
   use uppercase \xNN hex notation for nonprint/whitechars
help<Ent>  ..print this help
ver<Ent>   ..print fw version
reboot<Ent>  ..perform device reboot
?rxN<Ent>  ..list stored Rx ctrl strings of port(N)
?txN<Ent>  ..list stored Tx ctrl strings of port(N)
#rxN,M<Ent>  ..test port(N) stored ctrl string Rx[M]:  #rx0,2
#txN,M<Ent>  ..test port(N) stored ctrl string Tx[M]:  #tx1,2
#txN,M <data><Ent>  ..test port(N) user string Tx[M]:  #tx0,0 Hello\x0D
!rxN,M <data><Ent>  ..set  port(N) ctrl string Rx[M]:  !rx0,2 OPER=001\x0D\x0A
!txN,M <data><Ent>  ..set  port(N) ctrl string Tx[M]:  !tx1,2 PWR\x20is\x20off
!portN <data><Ent>  ..set  port(N) baudrate / parity:  !port0 115200,N,8,1
!respN <data><Ent>  ..set  port(N) autoresp. (0=no/1=yes):  !resp0 1
!pass <data><Ent>   ..set  passthru cmd mode (0=no/1=yes):  !pass 1
!init<Ent>  ..erase all stored data and set ports to default
>

ver
Per-RS232 translator v.1.4 [www.perys.cz]
>

?rx0

rx0,0: POF
rx0,1: PON
rx0,2: ADZZ;IIS:DVI
rx0,3: ADZZ;IIS:HD1
rx0,4: ADZZ;IIS:RG1
rx0,5: QPW
rx0,6: QSH
rx0,7:
>

>?tx1

tx1,0: PWR OFF
tx1,1: PWR ON
tx1,2: SOURCE A0
tx1,3: SOURCE 30
tx1,4: SOURCE 1F
tx1,5:
```

Serial
Name: COM8
Baud: 9600
Data size: 8
Parity: none
Handshake: OFF
Mode: Free

Modem lines
● CD ● RI ● DSR ● CTS ☐ DTR ☐ RTS

Send
$fd ☐ HEX Send
$c0$02$11$11 ☐ HEX Send
$c0$02$00$00 ☐ HEX Send