Basic / Advanced / Expert

Application note

How to control RS-232 enabled devices with SpaceLYnk

Including assignment to scenes



Safety Information

Important Information

Read these instructions carefully before trying to install, configure, or operate this software. The following special messages may appear throughout this bulletin or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of either symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠DANGER
DANGER indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.

⚠WARNING
WARNING indicates a potentially hazardous situation which, if not avoided, can result in death or serious injury.

⚠CAUTION
CAUTION indicates a potentially hazardous situation which, if not avoided, can result in minor or moderate injury.

NOTICE
NOTICE is used to address practices not related to physical injury. The safety alert symbol shall not be used with this signal word.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.

Safety Precautions

⚠WARNING

HAZARD OF INCORRECT INFORMATION

- Do not incorrectly configure the software, as this can lead to incorrect reports and/or data results.
- Do not base your maintenance or service actions solely on messages and information displayed by the software.
- Do not rely solely on software messages and reports to determine if the system is functioning correctly or meeting all applicable standards and requirements.
- Consider the implications of unanticipated transmission delays or failures of communications links.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please n otify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2017 Schneider Electric. All rights reserved

Table of Contents

1 Introduction 6

2 Design 8

2.1 Hardware configuration 8

3 Configuration 11

3.1 Communication configuration 11

3.2 LUA RS-232 commands 12

3.3 Including RS-232 commands into scenes 14

3.4 Connection setting and testing 17

4 Conclusion 18

5 Appendix 19

5.1 Glossary 19

1 Introduction

This application note describes how to set SpaceLYnk build inRS-232 interface for communication with 3rd party devices represented by the Barco projector in this application note. Wiring, initial setting and basic troubleshooting is described. Association with KNX objects and incorporation of controlled devices to the scenes is also explained.

Customer value proposition

The customer value propositions correspond to real use cases of SpaceLYnk RS-232 interface control of wide range of RS-232 enabled devices. It is still most preferred way of communication between AV and HVAC components for home automation.

Possibility to control:
· Projectors and displays
· Amplifiers, AV receivers and active speakers
· Air conditions, heat pumps and other HVAC technologies
· Meteorological stations and other precise sensors
· and many more

· Use case 1: Connection of common RS-232 devices trough most used connectors.
· Use case 2: Configuration of BARCO CRWQ-62B projector and SpaceLYnk for communication over RS-232.
· Use case 3: Simple command for switching of BARCO CRWQ-62B projector ON.
· Use case 4: Including control of of BARCO CRWQ-62B projector control into the scenes.

Competencies

There are few necessary competencies to follow procedures mentioned in this application note. It is mandatory to have knowledge of KNX commissioning trough KNX ETS software, basic SpaceLYnk configuration knowledge, fair knowledge or RS-232 interface and its settings and knowledge of Lua scripting language in the scope described in the Product manual.

System prerequisites

Software / Product name	version
SpaceLYnk	2.1
Hercules setup utility	3.2.6

Table 1: System prerequisites

Commercial number	Device ID	Description
LSS100100	SpaceLYnk	KNX IP logic controller
MNT684016	SCHNEIDER KNX power supply	Power supply for KNX components and SpaceLYnk
CRWQ-62B	BARCO CRWQ-62B	WQXGA, single chip DLP projector
UCAB232	RS-232 to USB converter	Converter cable

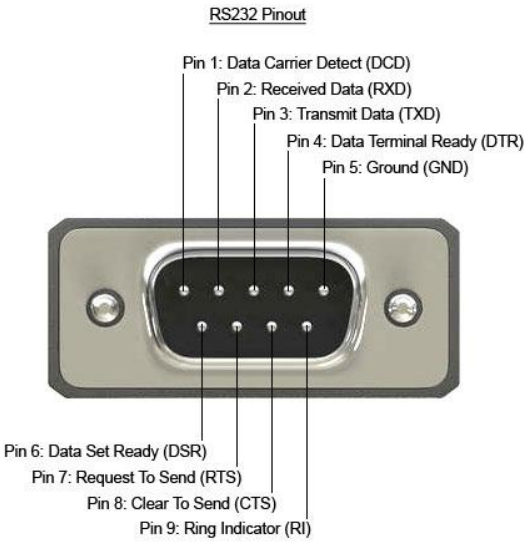
Table 2: List of used HW

2 Design

2.1 Hardware configuration

There are four most used connectors used for RS-232 control:

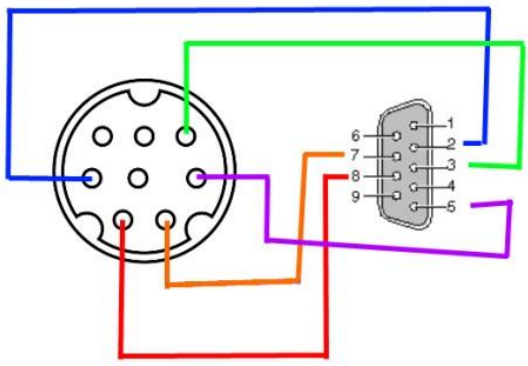
a) Classical 11 pins D-SUB connector commonly referred as CANON connector:



Picture 1 11-pins D-SUB connector

Please not that only RX, Tx and ground PINs are used with SpaceLYnk.

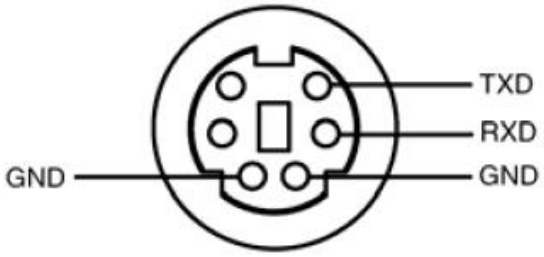
b) 8 pins mini DIN connector:



Picture 2 8–pins mini DIN connector

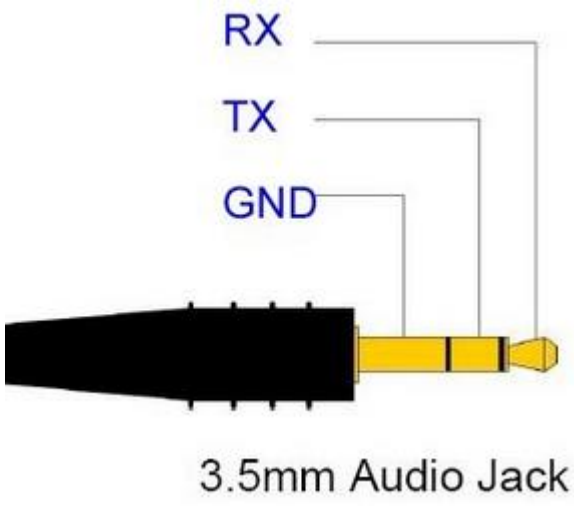
Please not that only RX, Tx and ground are used.

c) 6 pins mini DIN connector:



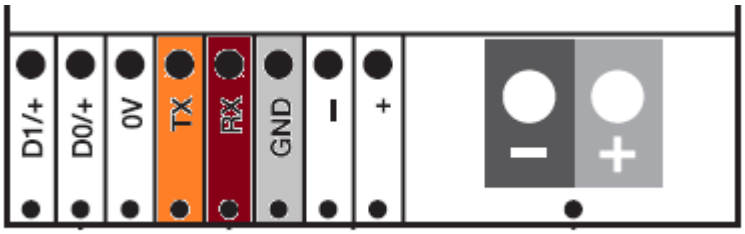
Picture 3 6-pins mini DIN connector

d) 3,5 mm Audio jack connector



Picture 4 3,5 mm audio jack connector

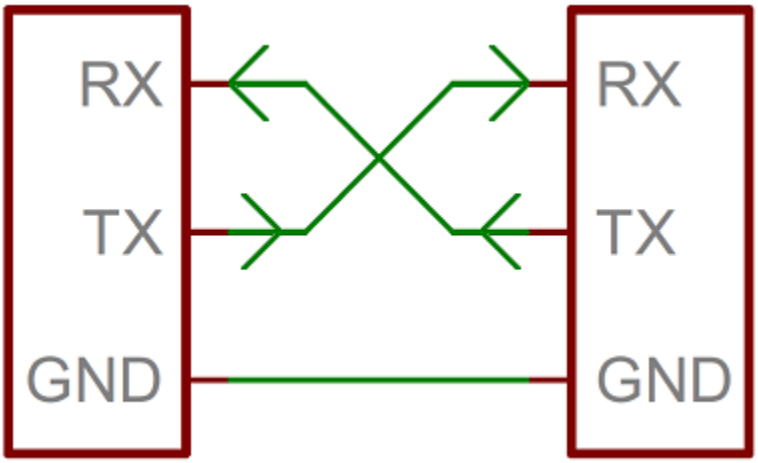
SpaceLYnk connection as follows:



Picture 5 SpaceLYnk terminals connection

Maximal recommended cable length is 5 m for 57600 bd, cable can be equally longer for lower speed e.g. 10 m for 28800 bd. By using low-capacitance cables, full speed communication can be maintained over larger distances up to about 1,000 feet (300 m).

It is mandatory to 'cross' the cable pins to ensure that Tx = transmitting pin will be connected to the Rx = receiving pin:



Picture 6 RS-232 'cross' connection

Please note: SpaceLYnk's is using standard RS-232C interface with signal level of ±12 Volts. If you want to use device which is using different voltage level e.g. RS-232TTL with signal level ±5V converter MUST be used to ensure proper function and prevention of damage of SpaceLYnk or connected device(s). Check manual of the device you want to connect first.

3 Configuration

3.1 Communication configuration

Setting of communication properties of the RS_232 as stated in the manual for controlled device. The only setting is on the SpaceLYnk side, RS-232 is enabled in the of BARCO CRWQ-62B projector by default. Communications settings must be included in the script addressing RS-232:

Connection Settings	Value
Baud Rate	19200 bps
Data Bits	8 bits
Parity	None
Stop Bits	1 bit
Flow control	None

Table 3 Connection parameters

Script for opening the RS-232 port, setting it's parameters and flushing it in case of any read/unsent leftovers:

```
-- Include library before calling serial functions
if not port then
require('serial')
  -- Setting port parameters and open serial port
port = serial.open('/dev/RS232', {
  baudrate = 19200,
  databits = 8,
  stopbits = 1,
  parity = 'none',
  duplex = 'full'
})
--  Flushes any read/unsent bytes
port:flush()
```

3.2 LUA RS-232 commands

Data can be sending to the RS-232 bus with commands in depending on used hardware. List of command is usually provided with the device. Sample from the of BARCO CRWQ-62B projector's manual:

Control Commands List

Command Description	Header (WORD)	Address Code (BYTE)	Size Of The Payload (WORD)	CRC16 For The Entire Packet (WORD)	MsG ID (WORD)	MsG Size (WORD)	Command Code (BYTE)	Value (BYTE)	Comment
Power On	0xefbe	0x10	0x0005	0xffc6	0x1111	0x0001	0x01		See Note 2
Menu	0xefbe	0x10	0x0005	0xbfc7	0x1111	0x0001	0x02		
Up	0xefbe	0x10	0x0005	0x7e07	0x1111	0x0001	0x03		
Down	0xefbe	0x10	0x0005	0x3fc5	0x1111	0x0001	0x04		
Left	0xefbe	0x10	0x0005	0xfe05	0x1111	0x0001	0x05		
Right	0xefbe	0x10	0x0005	0xbe04	0x1111	0x0001	0x06		

Table 4 sample of control codes list

Samples of possible formats:

a) Binary code for devices controlled by binary code:

```
-- Write data to serial port
port:write('0000001')
```

b) ASCII string format for command directly typed in human readable form

```
-- Write data to serial port
port:write('Powr___1')
```

c) Hexadecimal raw format (prefix 0x is needed for number to be recognized as a hexadecimal)

```
-- Write data to serial port
port:write('0xbe 0xef 0x10 0x05 0x00 0xc6 0xff 0x11 0x11 0x01 0x00 0x01')
```

c) String

```
-- Write data to serial port

msg = string.char(0x49, 0x52, 0x47, 0x42, 0x5F, 0x5F, 0x5F, 0x31)
port:write(msg)
```

Data can be also read from the RS-232 bus. Sample of pop-up message in the SpaceLYnk visualization when confirmation from projector is received:

```
-- Read data from serial port
if port:read() == 'Powr___1'
-- Send alert when condition matched
then alert('Projector is ON')
end
```

Port should be closed when not needed anymore:

```
-- Closing serial port
port:close()
```

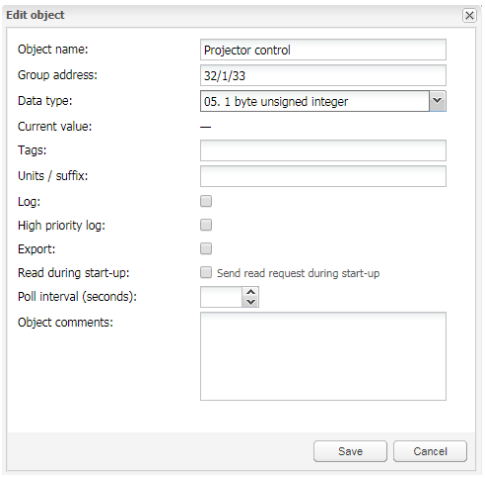
Final sample of complete code for switching of BARCO CRWQ-62B projector ON:

```
-- Include library before calling serial functions
if not port then
require('serial')
  -- Setting port parameters and open serial port
port = serial.open('/dev/RS232', {
  baudrate = 19200,
  databits = 8,
  stopbits = 1,
  parity = 'none',
  duplex = 'full'
})
--  Flushes any read/unsent bytes
port:flush()
-- Write data to serial port
msg = string.char(0x7E, 0x30, 0x30, 0x30, 0x20, 0x31, 0x0D)
port:write(msg)
-- Closing serial port
port:close()
end
```

3.3 Including RS-232 commands into scenes

SpaceLYnk offer creation of scenes, which are easy to create and edit. Following sample displays how to include command for switch ON/OFF the of BARCO CRWQ-62B projector when certain scene is activated.

Object for control of scenes needs to be created, data type need to be set to 1 byte unsigned integer. Object can be created as standard and or virtual if used only for control trough visualization:



Picture 7 Scene object properties

Scenes can be link directly as 'Event script':



Picture 8 Object link to the event script

Script with 3 scene samples recalled by the value of 'New scene' object::

If 'New scene' value will be 2 = 'going out' all light will be switched off, blinds lowered and of BARCO CRWQ-62B projector switched OFF for leaving the house.

If 'New scene value' will be 3 ='movie' lights will be set to low level, blinds lowered and of BARCO CRWQ-62B projector switched ON for movie watching.

Event-based: Event for 1/4/3 (1/4/3 New scene)

```
-- object mapped to this event must have its data type set
value = event.getvalue()

if (value == 0) then
  --scene 1 Welcome

  -- write value (e.g. boolean 'true') to object with group address 1/1/1, datatype must
be set for this object
  grp.write('1/0/10', true)  --corridor on
  grp.write('1/1/20', 0)     --blinds up
  grp.write('1/2/0', 1)   --night area mode to comfort
  grp.write('1/2/20', 1)  --day area mode to comfort


elseif (value == 1) then
  --scene 2 going out
  grp.write('1/0/21', false)  --all lights off
  grp.write ('1/0/26', false) --RGB cycling stop
  grp.write('1/1/20', 100)     --blinds down
  grp.write('1/0/28', 0)       --Red off
  grp.write('1/0/29', 0)       --green off
  grp.write('1/0/30', 0)       --blue off
  grp.write('1/2/0', 2)   --night area mode to stand by
  grp.write('1/2/20', 2)  --day area mode to stand by

  -- Include library before calling serial functions
    if not port then
    require('serial')
  -- Setting port parameters and open serial port
    port = serial.open('/dev/RS232', {
  baudrate = 19200,
  databits = 8,
  stopbits = 1,
  parity = 'none',
  duplex = 'full'
    })
    --  Flushes any read/unsent bytes
    port:flush()
    -- Write data to serial port Projector OFF
    msg = string.char(0x7E, 0x30, 0x30, 0x30, 0x30, 0x20, 0x30, 0x0D)
    port:write(msg)
    port:close()
    end


elseif (value == 2) then
  --scene 3 movie
  grp.write('1/0/6', true)  --kitchen lights on
  grp.write('1/0/19', 10)   --Living light 10%
  grp.write('1/1/14', 100)  --blinds living down
  grp.write('1/1/18', 100)  --blinds dinning down
  grp.write('1/0/28', 0)       --Red off
  grp.write('1/0/29', 0)       --green off
  grp.write('1/0/30', 20)       --blue 20%

    -- Include library before calling serial functions
    if not port then
```

```
  require('serial')
-- Setting port parameters and open serial port
  port = serial.open('/dev/RS232', {
baudrate = 19200,
databits = 8,
stopbits = 1,
parity = 'none',
duplex = 'full'
  })
  --  Flushes any read/unsent bytes
  port:flush()
  -- Write data to serial port Projector ON
  msg = string.char(0x7E, 0x30, 0x30, 0x30, 0x30, 0x20, 0x31, 0x0D)
  port:write(msg)
  port:close()
  end
```

Scene can be also easily called directly from the KNX buttons or visualization just by setting the object's value to desired scene number:



Picture 9 Recalling the scene from visualization

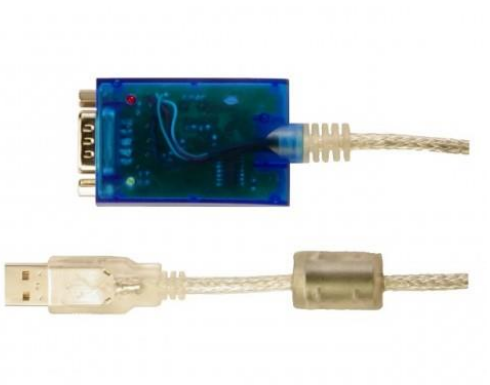
3.4 Connection setting and testing

In case of need of connection setting/testing, number of utilities can be used.

Free Hercules setup utility is one of them:

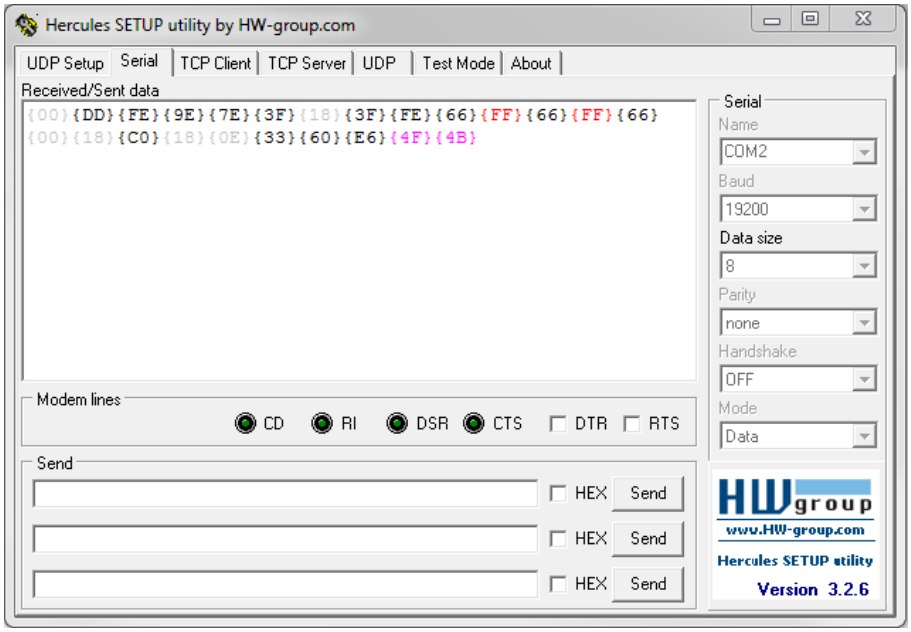
http://www.hw-group.com/products/hercules/index_en.html

RS-232 to USB cable is needed. ASIX UCAB232 cable was used in this case. This cable has two LEDs which are clearly signalling incoming/outgoing communication.



Picture 10 ASIX UCAB232 cable

When connected and properly set it can both receive commands from SpaceLYnk and sending commands to controlled device for testing it. Sample of sent control code with confirmation from the projector – hexadecimal code 4F 4B at the end = OK in ASCII.



Picture 10 Hercules utility environment with received code

4 Conclusion

SpaceLYnk can be used as gate for control of RS-232 devices. Proper settings and connection is mandatory for correct function. It is always important to get valid set of codes from the manufacturer of device.

5 Appendix

5.1 Glossary

The following table describes the acronyms and defines the specific terms used in this document.

Abbreviation	Description
RS-232	Telecommunications standard for binary serial communications between devices
bd	Baud = unit for transmitted symbols per seconds
hexadecimal	Positional number system with base of 16
ASCII	American Standard code for Information Interchange

Schneider Electric Industries SAS

Head Office
35, rue Joseph Monier
92506 Rueil-Malmaison Cedex
FRANCE

www.schneider-electric.com