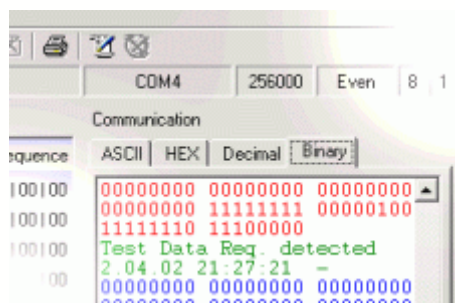




Docklight User Manual

Copyright 2002 Flachmann und Heggelbacher

Table of Contents

1 Copyright	1
2 Introduction	1
1 Typical Applications	2
2 System Requirements	3
3 User Interface	3
1 Main Window	3
4 Features and Functions	4
1 How Serial Data is Processed and Displayed	4
2 Editing and Managing Sequences	4
5 Working with Docklight	5
1 Testing a serial device or a protocol implementation	5
2 Simulating a serial device	6
3 Monitoring the serial communication between two devices	8
4 Logging and analyzing a test	9
5 Saving and loading your project data	9
6 Reference	10
1 Menu and Toolbar	10
2 Options	11
7 Appendix	12
1 ASCII Character Set Tables	12
2 RS232 Connectors / Pinout	14
3 Standard RS232 Cables	16
4 RS232 SUB D9 Monitoring Cable for Docklight	18
5 Sample Project: ModemDiagnostics.ptp	18
8 Glossary / Terms Used	19
1 Character	19
2 Sequence	19
3 Send Sequence	19
4 Receive Sequence	20
5 Action	20
6 RS232	20
7 RS422	20
8 RS485	21

9 CAN	21
10 DTE	21
11 DCE	21

1 Copyright

Issue 04/2002

Copyright 2002 Flachmann und Heggelbacher

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Trademarks

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

Disclaimer

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Contact

E-Mail Support: docklight@fuh-edv.de

Flachmann und Heggelbacher
Mittelstr. 82
D-90425 Nuernberg
Germany
<http://www.fuh-edv.de>

2 Introduction

Docklight is a test, analysis and simulation tool for serial communication protocols. It allows you to monitor the communication between two serial devices or to test the serial communication of a single device. Docklight is easy to use and runs on almost any standard PC using MS Windows 95/98/NT 4/2000/XP operating system.

Docklight can send out user-defined sequences according to the protocol used and it can react to incoming sequences. This makes it possible to simulate the behaviour of a serial communication device, which is particularly useful for generating test conditions that are hard to reproduce with the original device (e.g. problem conditions).

Docklight will work with the COM communication ports provided by your operating system. Physically, these ports will be **RS232** SUB D9 interfaces in most cases. However, it is also possible to use Docklight for other communication standards such as **RS485** and **RS422**, which have a different electrical design to RS232 but follow the RS232 communication mechanism. If you're developing systems which provide a **CAN** bus interface, then Docklight may also be used to test the CAN interface. You will just need a CAN to RS232 converter for your PC.

This manual will only refer to RS232 connections in detail, since this is the basis for the other standards mentioned above.

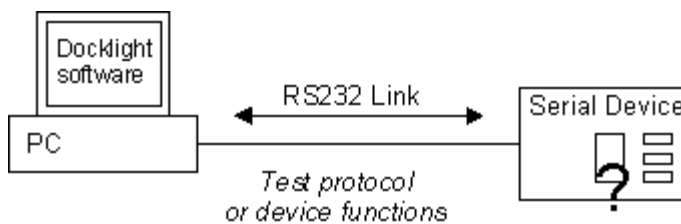
NOTE: If you have a modem available, try our [sample project](#) on modem diagnostics for getting started.

2.1 Typical Applications

Docklight is the ideal tool to support your development and testing process for serial communication devices. Docklight may be used to

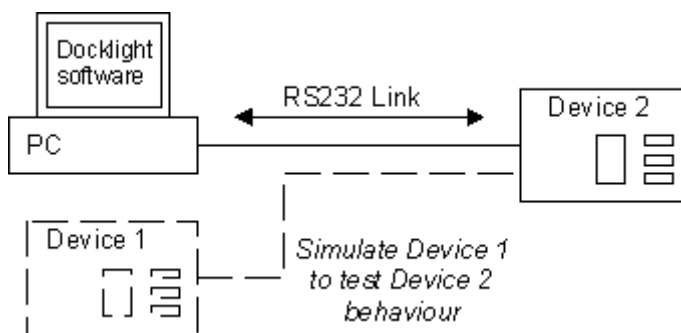
- [Test the functionality or the protocol implementation of a serial device](#) .

You may define control sequences recognized by your device, send them, log and analyze the responses and test the device reaction.



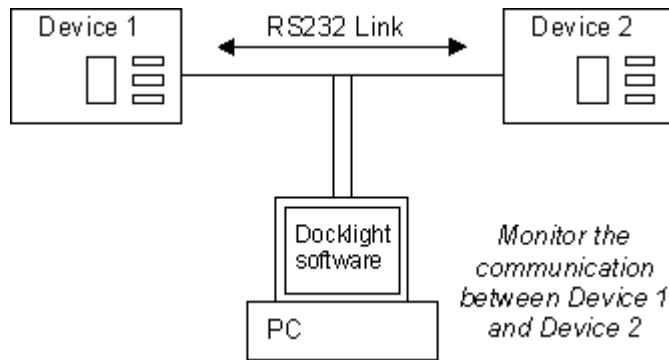
- [Simulate a serial device](#) .

Although rare, the possibility of a hardware fault must be considered in most systems. Imagine you have a device that will send an error message in case of a hardware fault, and a second device, which should receive this error message and perform some kind of reaction. Using Docklight you can easily simulate the error message to be sent and test the second device's reaction.



- [Monitor the communication between two devices](#) .

Insert Docklight into the communication link between two serial devices. Monitor and log the communication in both directions. Detect faulty communication sequences or special error conditions within the monitored communication.



2.2 System Requirements

Operating system:

- MS Windows 95/98/NT/2000/XP

Hardware requirements:

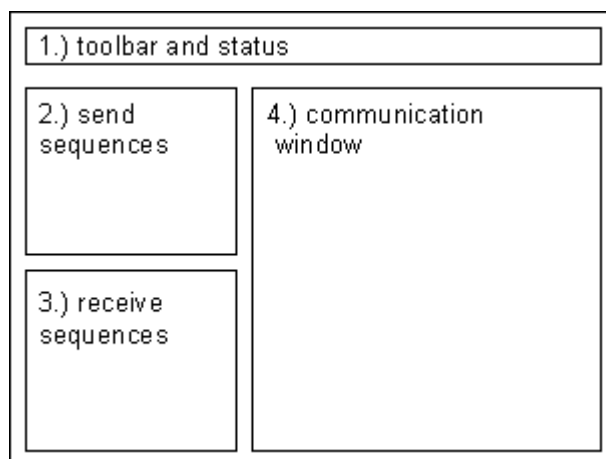
- Pentium processor 200 MHz or faster
- Minimum 32 MB RAM
- Minimum one COM port available. Two COM ports for monitoring communication between two serial devices.

Additional cables may be required for connecting the equipment to be tested. See sections [Standard RS232 Cables](#) and [RS232 SUB D9 Monitoring Cable for Docklight](#).

3 User Interface

3.1 Main Window

The main window of Docklight is divided into four sections:



1. Toolbar and status

All main Docklight functions may be selected from the toolbar. Additional information about the serial communication settings is displayed in the status line below.

2. Send sequences

Define, edit and manage your [send sequences](#) here. Using the arrow symbol, the selected

sequence will be sent out immediately.

3. Receive sequences

Define, edit and manage your [receive sequences](#) here.

4. Communication

Displays the outgoing and incoming communication from the serial port. Various display options are available for the communication data, including ASCII / HEX / decimal display, time stamps, highlighting.

4 Features and Functions

4.1 How Serial Data is Processed and Displayed

Docklight handles all serial data in an 8 bit-oriented way. Every [sequence](#) of serial data consists of one or more 8 bit [characters](#). All communication characters sent or received from the serial port are logged with a time stamp.

Docklight allows you to

- display the serial data in either ASCII, HEX, decimal or binary format
- copy serial data to the clipboard and paste it into a standard text file
- print out serial data, user comments and other extra information

Docklight will log all incoming and outgoing serial data for you. The communication window of Docklight will always show you the current communication on the serial port. Incoming and outgoing data are displayed using different colors or fonts, and additional date/time information may be included. The logged data may be copied to the clipboard, printed or stored to a standard ASCII text file.

4.2 Editing and Managing Sequences

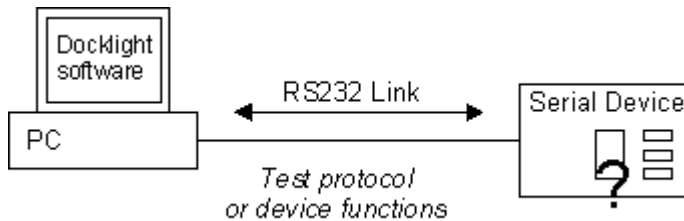
A Docklight project mainly consists of user-defined sequences. These may be either [receive sequences](#) which are used to detect a special message within the incoming serial data, or [send sequences](#) which may be transmitted by Docklight itself.

Sequences are defined using the **Edit Send Sequence** or **Edit Receive Sequence** dialog window. This dialog window is opened

1. when clicking on an existing sequence within the send sequence or receive sequence list.
2. when creating a new sequence by clicking on the blank field at the end of a list.

5 Working with Docklight

5.1 Testing a serial device or a protocol implementation



Preconditions

- You need the specification of the protocol to test, e.g. in written form.
- The serial device to test should be connected to one of the PC's COM ports. See section [Standard RS232 Cables](#) for details on how to connect two serial devices.
- The serial device must be ready to operate.

Performing the Test

A) Creating a new project

Create a new Docklight project by selecting the menu **File-->New Project**

B) Setting the Communication Options

1. Choose the menu **Tools-->Options...** and select the **Communication** tab.
2. Choose communication mode **Send/Receive**
3. At **Send/Receive on COM port**, set the COM port number where your serial device is connected.
4. Set the baud rate and all other **COM Port Settings** required.
5. Confirm the settings and close the dialog by clicking the **OK** button.

C) Defining the send sequences used

You will probably test your serial device by sending specific sequences, according to the protocol of the device, and observe the device's reaction. Perform the following steps to create your list of sequences.

1. Click on the last line of the **Send Sequences** table. The dialog **Edit Send Sequence** is displayed.
2. Enter a **Name** for the sequence. The sequence name should be unique for every send sequence defined.
3. Enter the **Sequence** itself. You may enter the sequence either in ASCII, hex, decimal or binary format. Switching between the different formats is possible at any time using the **Edit Mode** radio buttons.
4. After clicking the **OK** button the new sequence will be added to the send sequence lists.

Repeat steps 1.-4. to define other send sequences needed to perform your test.

D) Defining the receive sequences used

If you want Docklight to react when receiving specific sequences, you have to define a list of receive sequences.

1. Click on the last line of the **Receive Sequences** table. The Dialog **Edit Receive Sequence** is displayed. The dialog consist of three parts: **Name** field, **Sequence** field, and **Action** field.

2. Edit the **Name** and **Sequence** fields. These fields work exactly like the corresponding fields of the **Edit Send Sequence** Dialog.
3. Specify an **Action** to perform after the sequence has been received by Docklight. There are two types of actions possible:
Answer - After receiving the sequence, transmit one of the send sequences.
Comment - After receiving the sequence, insert a user-defined comment into the communication window (and log file, if available).
4. Click the **OK** button to add the new sequence to the list.

Repeat steps 1.-4. to define other receive sequences you need to perform your test.

E) Storing the project

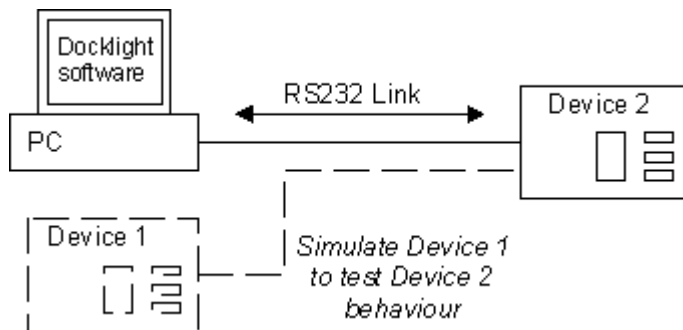
Before running the actual test, it is recommended to store the communication settings and sequences defined. This is done using the menu **File-->Save Project**

F) Running the test

Start Docklight by pressing the  **Start Communication** button in the toolbar.

Docklight will open a serial connection according to the parameters specified. It will now display all incoming and outgoing communication in the communication window. Use the  **Send** button to send one of the sequences defined to the serial device. The on-screen display of all data transfer allows you to check the device's behaviour. All protocol information can be logged in a text file for further analysis, see section [Logging and analyzing a test](#).

5.2 Simulating a serial device



Preconditions

- You need the specification of the behaviour of your serial device you want to simulate, e.g. what kind of information is sent back after receiving a certain command.
- A second device is connected to a PC COM port, which will communicate with your simulator.

This second device and its behaviour is the actual object of interest. An example could be a device that periodically checks the status of an UPS (Uninterruptible Power Supply) using a serial communication protocol. You could use Docklight to simulate basic UPS behaviour and certain UPS problem cases. This is very useful when testing the other device, because it can be quite difficult to reproduce a problem case (like a bad battery) at the real UPS.

NOTE: The second device may also be a PC software. It is possible to run both Docklight and the PC software on the same machine. Simply use a different COM port for each of the two applications and connect the two COM ports using a standard null modem cable (see section [standard RS232 cables](#)).

Performing the Test

A) Creating a new project

Create a new Docklight project by selecting the menu **File-->New Project**

B) Setting the Communication Options

1. Choose the menu **Tools-->Options...** and select the **Communication** tab.
2. Choose communication mode **Send/Receive**
3. At **Send/Receive on COM port**, set the COM port number where your serial device is connected.
4. Set the baud rate and all other **COM Port Settings** required.
5. Confirm the settings and close the dialog by clicking the **OK** button.

C) Defining the send sequences used

Define all the responses of your simulator. Think of responses when the simulated device is in normal conditions, as well as responses when in fault condition. For the UPS example mentioned above, a battery failure would be such a problem case that is hard to reproduce with the original equipment. To test how other equipment reacts to a battery failure, define the appropriate response sequence your UPS would send in this case.

The response sequence of your protocol will probably be sent out only after receiving a specific input sequence. The received sequences are defined in the next step and linked to the send sequences defined here.

D) Defining the receive sequences used

In most cases, your simulated device will not send the data unrequested, but will be polled from the other device. The other device will use a set of predefined command sequences to request different type of information. Define the command sequences that must be interpreted by your simulator here.

For every command sequence defined, specify **Answer** as an action. Choose one of the sequences defined in C). If you want to use two or more alternative response sequences, make several copies of the same receive sequence, give them a different name (e.g. "status cmd - answer ok", "status cmd - answer battery failure", "status cmd - answer mains failure") and assign different send sequences as an action. After that you will have three elements in the receive sequences list that would respond to the same command with three different answers. During the test you may decide which answer should be sent by checking or unchecking one of these list elements (**Active** column).

E) Storing the project

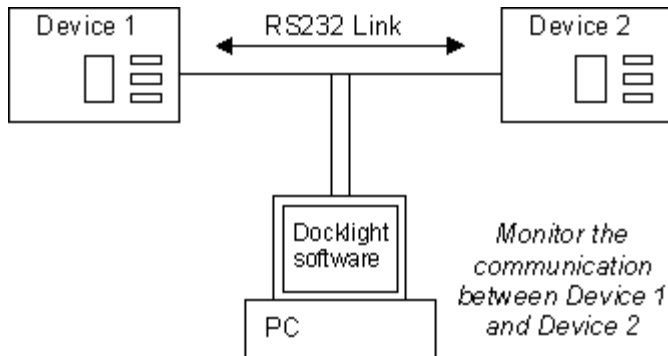
Before running the actual test, it is recommended to store the communication settings and sequences defined. This is done using the menu **File-->Save Project**

F) Running the test

Start Docklight by pressing the  **Start Communication** button in the toolbar.

Docklight will now respond to all commands received from the serial device connected. The on-screen display of all data transfer allows you to monitor the communication flow. All protocol information can be logged in a text file for further analysis, see section [Logging and analyzing a test](#).

5.3 Monitoring the serial communication between two devices



Preconditions

- You need the specification of the communication protocol that is used by the two serial devices.
- A special [RS232 monitoring cable](#) is needed which connects to the RS232 sender of both serial devices and feeds them into Docklight, while not interfering with the communication signals between the two devices.
- Two COM ports must be available for monitoring. Each port will receive the data from one of the two serial devices to monitor.
- The serial devices are ready to operate and the monitoring cable is connected.

NOTE: Make sure your PC COM port works properly for the baud rate and communication settings used between Device 1 and Device 2. If Device 1 and Device 2 use a high speed data transfer protocol that includes a hardware handshaking mechanism to prevent the loss of data, the PC COM interface might be too slow to receive every data byte properly.

Performing the Test

A) Creating a new project

Create a new Docklight project by selecting the menu **File-->New Project**

B) Setting the Communication Options

1. Choose the menu **Tools-->Options...** and select the **Communication** tab.
2. Choose communication mode **Monitoring**
3. At **Receive channel 1 data from COM port**, set the COM port number where the monitoring signal from serial device 1 is received. At **Receive channel 2 data from COM port**, set the COM port for the second device.
4. Set the baud rate and all other communication parameters of the protocol used.
5. Confirm the settings and close the dialog by clicking the **OK** button.

C) Defining the receive sequences used

Define receive sequences which should be marked in the test protocol or which should trigger an action within Docklight. Docklight checks for receive sequence on both monitoring channels, i.e. it does not matter whether the sequences come from serial device 1 or serial device 2.

NOTE: Since a special monitoring cable is used for this test, all communication between serial device 1 and serial device 2 will remain unbiased and no additional delays will be introduced by Docklight itself. This is particularly important when using Docklight for tracking down timing problems. This means on the other hand, that there is no way to influence the serial communication between the two devices. While communication mode **Monitoring** is selected, it is therefore impossible to use send sequences.

D) Storing the project

Before running the actual test, it is recommended to store the communication settings and sequences defined. This is done using the menu **File-->Save Project**

F) Running the test

Start the protocol tester by pressing the  **Start Communication** button in the toolbar. Then activate the serial devices 1 and 2 and perform a test run. Docklight will now display all communication between serial device 1 and serial device 2. Docklight uses different colors and font types to make it easy to distinguish between data transmitted by device 1 or device 2. The colors and font types may be chosen in the [Display-1](#) tab of the Options dialog.

5.4 Logging and analyzing a test

Preconditions

- Docklight is ready to run a test as described in the previous use cases:
[Testing a serial device or a protocol implementation](#)

Logging the Test

A) Choosing the log file format

1. Choose the menu **Tools-->Options...** and select the **Logging** tab.
2. Choose which representation of the communication data to be stored in a log file (ASCII, HEX, decimal or binary representation).

B) Turning on the communication log

Click on the  **Start Logging** button of the main toolbar. A dialog window is opened to select the path and file name for the log file(s).

For each representation (ASCII, HEX, ...), a separate log file will be created. The log files will have a ".txt" file extension and may be opened by any type of text editor. Docklight additionally adds the representation type to the file name to distinguish the different log files. E.g. if the user specifies "Test1" as the log file name, the ASCII log file will be named "Test1_asc.txt", whereas the HEX log file will be named "Test1_hex.txt".

To close the log file(s), click the  **Stop Logging** button of the main toolbar. Unless the log file(s) have been closed, it is not possible to view them using a text editor.

5.5 Saving and loading your project data

All project data may be saved in a Docklight project file using the menu **File-->Save Project** or **File-->Save Project As...**

The project data includes

- send sequences defined
- receive sequences defined

It is recommended to save your current project before starting a test run. Please note the difference between storing the project settings as described here and logging the communication during a test (see section [Logging and Analyzing a test](#)).

NOTE: Docklight will not save the current COM port and display settings as project data.

Loading a project is done using the **File-->Open Project...** menu.

6 Reference

6.1 Menu and Toolbar

File Menu



New Project

Close the current Docklight project and create a new one.



Open Project ...

Close the current Docklight project and open another project.



Save Project / Save Project As ...

Save the current Docklight project.

Print Project ...

Print the project data, i.e. the list of send sequences and receive sequences defined. The sequences are printed in the same representation (ASCII, HEX, decimal or binary) that is used for the Docklight main window. The representation may be chosen at the [Options](#) dialog window.



Print Communication ...

Print the contents of the communication window. The communication data is printed in the same representation that is currently visible in the communication window.

Exit

Exit the Docklight application.

Tools Menu



Options...

Choose general communication and display settings.

Additional Toolbar Buttons



Start communication

Open the communication ports and enable serial data transfer.



Stop communication

Stop serial data transfer and close the communication ports.



Clean Communication Window

Delete the contents of the communications window. This applies to all four representations (ASCII, HEX, Decimal, Binary) of the communication window.



Start Communication Logging

Create new log file(s) and start logging the incoming/outgoing serial data. See [logging and analyzing a test](#).



Stop Communication Logging

stop logging and close the currently opened log file(s) . See [logging and analyzing a test](#).

6.2 Options

Communication

Basic communication mode

- Send/Receive:
Docklight acts both as transmitter and receiver of serial data. This mode is applied when [Testing the functionality or the protocol implementation of a serial device](#) or [simulating a serial device](#).
Naming conventions: The received data will be displayed and processed as "Channel 1", the transmitted data will be displayed as "Channel 2".
- Monitoring:
Docklight will receive serial data on two different COM ports. This applies to: [Monitoring the communication between two devices](#). Docklight will not interfere with the serial communication (entirely passive monitoring).
Naming conventions: The serial data from device 1 will be handled as "Channel 1", the data from device 2 is "Channel 2" respectively.

Display-1

Color and text attributes

Set the appearance of the Docklight communication window. The two different serial data streams "Channel 1" and "Channel 2" may be displayed using different color and font types. The standard setting is using different colors for the two channels, but using different font styles (e.g. Italics for "Channel 2") is also possible.

Date and time stamp

Docklight adds a date/time stamp to every [character](#) transmitted or received. You may choose here if this date/time stamp is visible within the communication window.

Display-2

Sequence Windows Display Options

Choose the representation (ASCII, HEX, Decimal or Binary) to use for the send sequences and receive sequences lists. The selected representation will also be applied when printing the project data.

7 Appendix

7.1 ASCII Character Set Tables

Control Characters

Dec	Hex	ASCII Char.	Meaning	Keyboard Entry
0	00	NUL	Null	Ctrl-@
1	01	SOH	Start of heading	Ctrl-A
2	02	STX	Start of text	Ctrl-B
3	03	ETX	Break/end of text	Ctrl-C
4	04	EOT	End of transmission	Ctrl-D
5	05	ENQ	Enquiry	Ctrl-E
6	06	ACK	Positive acknowledgment	Ctrl-F
7	07	BEL	Bell	Ctrl-G
8	08	BS	Backspace	Ctrl-H
9	09	HT	Horizontal tab	Ctrl-I
10	0A	LF	Line feed	Ctrl-J
11	0B	VT	Vertical tab	Ctrl-K
12	0C	FF	Form feed	Ctrl-L
13	0D	CR	Carriage return	Ctrl-M
14	0E	SO	Shift out	Ctrl-N
15	0F	SI	Shift in/XON (resume output)	Ctrl-O
16	10	DLE	Data link escape	Ctrl-P
17	11	DC1	Device control character 1	Ctrl-Q
18	12	DC2	Device control character 2	Ctrl-R
19	13	DC3	Device control character 3	Ctrl-S
20	14	DC4	Device control character 4	Ctrl-T
21	15	NAK	Negative Acknowledgment	Ctrl-U
22	16	SYN	Synchronous idle	Ctrl-V
23	17	ETB	End of transmission block	Ctrl-W
24	18	CAN	Cancel	Ctrl-X
25	19	EM	End of medium	Ctrl-Y
26	1A	SUB	substitute/end of file	Ctrl-Z
27	1B	ESC	Escape	Ctrl-[
28	1C	FS	File separator	Ctrl-\
29	1D	GS	Group separator	Ctrl-]
30	1E	RS	Record separator	Ctrl-^
31	1F	US	Unit separator	Ctrl-_

Printing Characters

Dec	Hex	ASCII Char.	Meaning	Keyboard Entry
32	20	SP	Space	Space
33	21	!	!	!
34	22	"	"	"
35	23	#	#	#
36	24	\$	\$	\$
37	25	%	%	%
38	26	&	&	&
39	27	'	'	'
40	28	(	(	(
41	29	)	)	)
42	2A	*	*	*
43	2B	+	+	+
44	2C	,	,	,
45	2D	-	-	-

46	2E	.	.	.
47	2F	/	/	/
48	30	0	Zero	0
49	31	1	One	1
50	32	2	Two	2
51	33	3	Three	3
52	34	4	Four	4
53	35	5	Five	5
54	36	6	Six	6
55	37	7	Seven	7
56	38	8	Eight	8
57	39	9	Nine	9
58	3A	:	:	:
59	3B	;	;	;
60	3C	<	<	<
61	3D	=	=	=
62	3E	>	>	>
63	3F	?	?	?
64	40	@	@	@
65	41	A	A	A
66	42	B	B	B
67	43	C	C	C
68	44	D	D	D
69	45	E	E	E
70	46	F	F	F
71	47	G	G	G
72	48	H	H	H
73	49	I	I	I
74	4A	J	J	J
75	4B	K	K	K
76	4C	L	L	L
77	4D	M	M	M
78	4E	N	N	N
79	4F	O	O	O
80	50	P	P	P
81	51	Q	Q	Q
82	52	R	R	R
83	53	S	S	S
84	54	T	T	T
85	55	U	U	U
86	56	V	V	V
87	57	W	W	W
88	58	X	X	X
89	59	Y	Y	Y
90	5A	Z	Z	Z
91	5B	[	[	[
92	5C	\	\	\
93	5D	]	]	]
94	5E	^	^	^
95	5F	_	_	_
96	60	`	`	`
97	61	a	a	a
98	62	b	b	b
99	63	c	c	c
100	64	d	d	d
101	65	e	e	e
102	66	f	f	f
103	67	g	g	g
104	68	h	h	h
105	69	i	i	i
106	6A	j	j	j
107	6B	k	k	k
108	6C	l	l	l

109	6D	m	m	m
110	6E	n	n	n
111	6F	o	o	o
112	70	p	p	p
113	71	q	q	q
114	72	r	r	r
115	73	s	s	s
116	74	t	t	t
117	75	u	u	u
118	76	v	v	v
119	77	w	w	w
120	78	x	x	x
121	79	y	y	y
122	7A	z	z	z
123	7B	{	{	{
124	7C			
125	7D	}	}	}
126	7E	~	Tilde	~
127	7F	DEL	Delete	Del

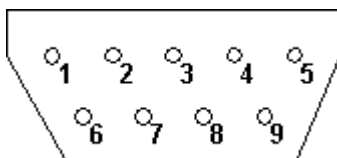
7.2 RS232 Connectors / Pinout

The most common connectors for RS232 communication are

- 9-pole SUB D9 (EIA/TIA 574 standard). Introduced by IBM and widely used. See below.
- [25-pole SUB D25](#) (RS232-C). This is the original connector introduced for the RS232 standard. It provides a secondary communication channel.
- [8-pole RJ45](#) (RS232-D, according to EIA/TIA-561 standard).

RS232 SUB D9 Pinout

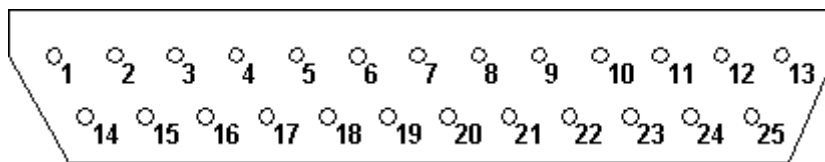
View: [DTE](#) perspective, looking into the male connector



Pin No.	Signal Name	Description
1	DCD	Data Carrier Detect
2	RxD	Receive Data
3	TxD	Transmit Data
4	DTR	Data Terminal Ready
5	SGND	Signal Ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indicator

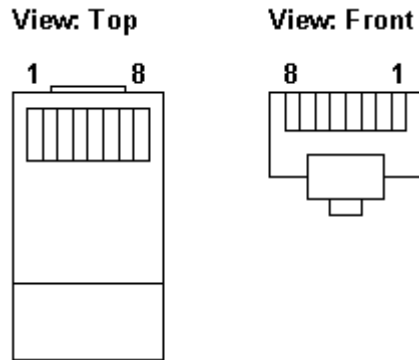
RS232 SUB D25 Pinout

View: [DTE](#) perspective, looking into the male connector



Pin No.	Signal Name	Description
1	-	Protective/Shielding Ground
2	TxD	Transmit Data
3	RxD	Receive Data
4	RTS	Request To Send
5	CTS	Clear To Send
6	DSR	Data Set Ready
7	SGND	Signal Ground
8	DCD	Data Carrier Detect
9	-	Reserved
10	-	Reserved
11	-	Unassigned
12	SDCD	Secondary Data Carrier Detect
13	SCTS	Secondary Clear To Send
14	STxD	Secondary Transmit Data
15	TxCLK	Transmit Clock
16	SRxD	Secondary Receive Data
17	RxCLK	Receive Clock
18	LL	Local Loopback
19	SRTS	Secondary Request To Send
20	DTR	Data Terminal Ready
21	RL/SQ	Remote Loopback / Signal Qualify Detector
22	RI	Ring Indicator
23	CH/CI	Signal Rate Selector
24	ACLK	Auxiliary Clock
25	-	Unassigned

RS232-D, RJ45 pinout



Pin No.	Signal Name	Description
1	DSR / RI	Data Set Ready / Ring Indicator
2	DCD	Data Carrier Detect
3	DTR	Data Terminal Ready
4	SGND	Signal Ground
5	RxD	Receive Data
6	TxD	Transmit Data
7	CTS	Clear To Send
8	RTS	Request To Send

7.3 Standard RS232 Cables

RS232 Connections

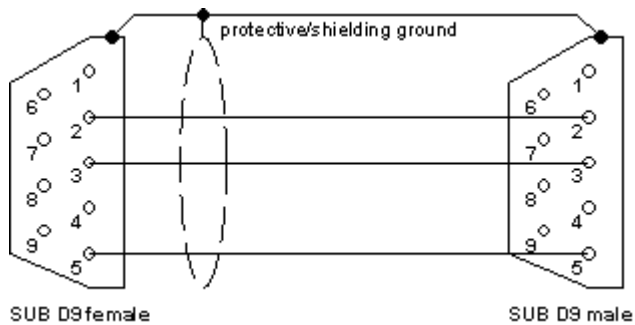
When connecting two serial devices, different cable types must be used, depending on the characteristics of the serial device and the type of communication used.

Overview of RS232 SUB D9 interconnections

serial device 1	serial device 2	flow control (handshaking)	recommended cable
DTE (Data Terminal)	DTE	no handshake signals	simple null modem cable
DTE	DTE	DTE/DCE compatible hardware flow control	null modem cable with partial handshaking
DTE	DCE (Data Communications Equipment)	no handshake signals	simple straight cable
DTE	DCE	hardware flow control	full straight cable
DCE	DCE	no handshake signals	simple null modem cable, but with SUB D9 male connectors on both ends
DCE	DCE	hardware flow control	null modem cable with partial handshaking but with SUB D9 male connectors on both ends

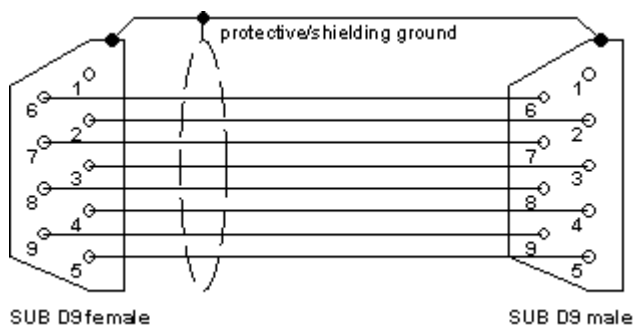
SUB D9 Simple Straight Cable

Area of Application: [DTE-DCE](#) Communication where no additional handshake signals are used.



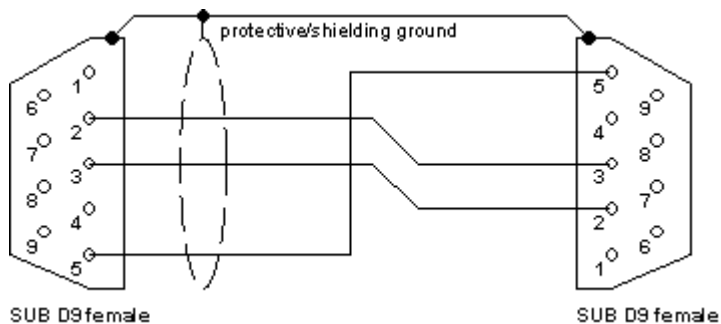
SUB D9 Full Straight Cable

Area of Application: [DTE-DCE](#) Communication with hardware flow control using additional handshake signals.



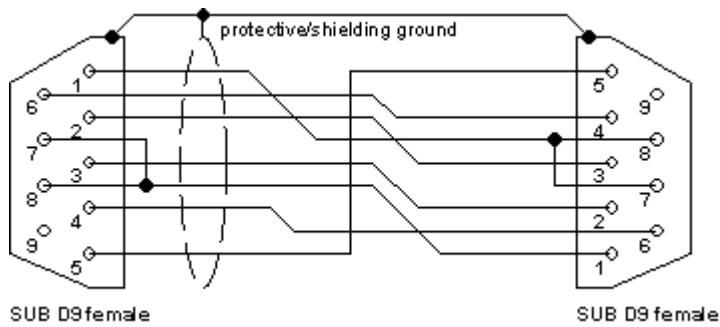
SUB D9 Simple Null Modem Cable without Handshaking

Area of Application: [DTE-DTE](#) Communication where no additional handshake signals are used.



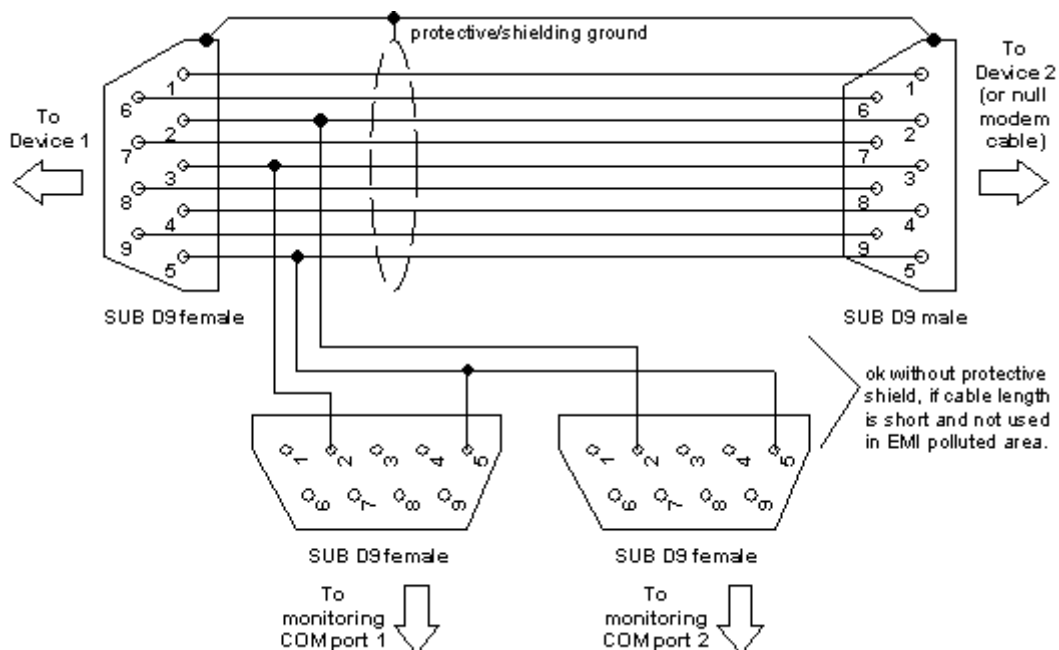
SUB D9 Null Modem Cable with Partial Handshaking

Area of Application: [DTE-DTE](#) Communication with DTE/DCE compatible hardware flow control. Works also when no handshake signals are used.



7.4 RS232 SUB D9 Monitoring Cable for Docklight

Area of Application: [Monitoring the serial communication between two devices](#)



7.5 Sample Project: ModemDiagnostics.ptp

The sample project is located in the folder **Samples** within the Docklight installation directory.

NOTE: If you are working on Windows 2000 or XP and you have a user account with restricted system access, you might not be able to write to the Docklight installation directory. In this case, saving a project file into the **Samples** folder will produce an error.

The Docklight project **ModemDiagnostics.ptp** can be used to perform a modem check. In the send sequences list, a set of modem diagnostics commands are defined.

To perform a test run, follow the hints from chapter [Testing a serial device or a protocol implementation](#). Use the following [communication options](#):

Communication Mode	Send/Receive
COM Port Settings	9600 Baud, No parity, 8 Data Bits, 1 Stop Bit

The send sequences list includes the following standard AT modem commands:

Send Sequence	Description / Modem Response
ATQ0V1E0	Initializes the query.
AT+GMM	Model identification (ITU V.250 recommendation is not supported by all modems).
AT+FCLASS=?	Fax classes supported by the modem, if any.
AT#CLS=?	Shows whether the modem supports the Rockwell voice command set.
ATI<n>	Displays manufacturer's information for <n> = 1 through 7. This provides information such as the port speed, the result of a checksum test, and the model information. Check the manufacturer's documentation for the expected results.

The **Samples** folder also contains a log file **SampleModemLog_asc.txt**. It shows a test run where the above send sequences were applied to a real modem.

8 Glossary / Terms Used

8.1 Character

A character is the basic unit of information processed by Docklight. Docklight always uses 8 bit characters. Nevertheless, the communication settings also allow data transmission with 7 bits or less. In this case, only a subset of the 256 possible 8 bit characters will be used, but the characters are still stored and processed in an 8 bit format.

8.2 Sequence

A sequence consists of one or several 8 bit [characters](#). A sequence may be any part of the serial communication you want to investigate. It may consist of printable ASCII characters, but can also include every non-printable character between 0 and 255 decimal value.

Example:

ATL2 (ASCII format)

41 54 4C 0D 0A (Hex format)

This sequence is a modem command to set the speaker volume on AT compatible modems. It includes a Carriage Return (0D) and a Line Feed (0A) character at the end of the line.

8.3 Send Sequence

A send sequence is a [sequence](#) that can be sent by Docklight. A send sequence is specified by

1. an unique name (e.g. "Set modem speaker volume"),
2. a sequence (e.g. "41 54 4C 0D 0A" in hex format),
3. additional attributes to specify how the sequence is sent.

There are two ways to make Docklight send a sequence:

- Sending a sequence can be triggered manually by pressing the send button in the send sequences window.
- Sending a sequence may be one possible reaction of Docklight when detecting a specific

receive sequence within the incoming data (see [Action](#)).

8.4 Receive Sequence

A receive sequence is a [sequence](#) that can be detected by Docklight within the incoming serial data. A receive sequence is specified by

1. an unique name (e.g. "Modem Answer OK"),
2. a sequence (e.g. "6F 6B 13 10" in hex format),
3. an [action](#) that is triggered when then Docklight receives the defined sequence.

8.5 Action

For a receive sequence, the user may define an action that is performed after receiving the specific sequence. Possible actions are

- sending out a [send sequence](#),
- inserting a comment into the protocol.

8.6 RS232

The RS232 standard is defined by the EIA/TIA (Electronic Industries Alliance / Telecommunications Industry Associations). The standard defines the asynchronous serial data transfer mechanism, as well as the physical and electrical characteristics of the interface.

RS232 uses serial bit streams transmitted at a predefined baud rate. The information is separated into characters of 5 to 8 bits lengths. Additional start and stop bits are added for synchronization, and a so-called parity bit may be included to provide a simple error detection mechanism.

The electrical interface includes unbalanced line drivers, i.e. all signals are represented by a voltage with reference to a common signal ground. RS232 defines two states for the data signals: mark state (or logical 1) and space state (or logical 0). The range of voltages for representing these states is specified as follows:

Signal State	Transmitter Voltage Range	Receiver Voltage Range
Mark (logical 1)	-15V to -5V	-25V to -3V
Space (logical 0)	+5V to +15V	+3V to +25V
Undefined	-5V to +5V	-3V to +3V

The physical characteristics of the RS232 standard is described in the section [RS232 Connectors / Pinout](#)

8.7 RS422

A RS422 communication link is a four-wire link with balanced line drivers. In a balanced differential system, one signal is transmitted using two wires (A and B). The signal state is represented by the voltage across the two wires. Although a common signal ground connection is necessary, it is not used to determine the signal state at the receiver. This results in a high immunity against EMI (electromagnetic interference) and allows cable lengths of over 1000m, depending on the cable type and baud rate.

The EIA Standard RS422-A "Electrical characteristics of balanced voltage digital interface circuits" defines the characteristics of a RS422 interface.

Transmitter and receiver characteristics according to RS422-A:

Signal State	Transmitter Differential Voltage V_{AB}	Receiver Differential Voltage V_{AB}
Mark (or logical 1)	-6V to -2V	-6V to -200mV
Space (or logical 0)	+2V to +6V	+200mV to 6V
Undefined	-2V to +2V	-200mV to +200mV
	Permitted Common Mode Voltage V_{cm} (mean voltage of A and B terminals with reference to signal ground)	
	-7V to +7V	

8.8 RS485

The RS485 standard defines a balanced two-wire transmission line, which may be shared as a bus line by up to 32 driver/receiver pairs. Many characteristics of the transmitters and receivers are the same as [RS422](#). The main differences to RS422 are

- Balanced line drivers with tristate capability: The RS485 line driver has an additional "enable" signal which is used to connect and disconnect the driver to its output terminal. The term "tristate" refers to the three different states possible at the output terminal: mark (logical 1), space (logical 0) or "disconnected"
- Extended Common Mode Voltage (V_{cm}) range from -7V to +12V.

The EIA Standard RS485 "Standard for electrical characteristics of generators and receivers for use in balanced digital multipoint systems" defines the characteristics of a RS485 system.

8.9 CAN

Controller Area Network. A serial bus communication standard originally developed for automotive applications, but now also widely used in any kind of distributed embedded system.

8.10 DTE

Data Terminal Equipment. E.g. a terminal or a PC. Most commonly equipped with a SUB D9 or SUB D9 **male** connector. All pinning specifications are written from a DTE perspective. See also [DCE](#).

8.11 DCE

Data Communications Equipment. E.g. a modem. Most commonly equipped with a SUB D9 or SUB D25 **female** connector. See also [DTE](#).