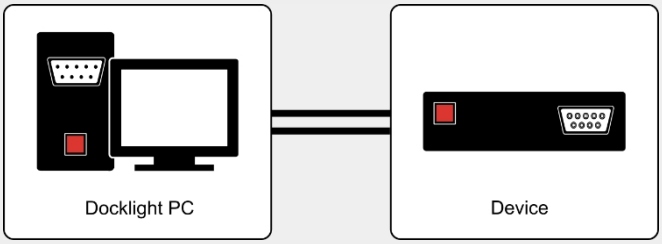
Docklight Scripting V2.4
Benutzerhandbuch 02/2023



Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

1. Urheberrecht 7

2. Einführung 9

2.1 Docklight - Übersicht 10

2.2 Docklight Scripting - Überblick 10

2.3 Typische Anwendungen 11

2.4 Systemanforderungen 12

3. Benutzeroberfläche 14

3.1 Hauptfenster (Scripting) 15

3.2 Zwischenablage - Ausschneiden, Kopieren und Einfügen 16

3.3 Bereich Dokumentation 16

4. Merkmale und Funktionen 18

4.1 Wie serielle Daten verarbeitet und angezeigt werden 19

4.2 Bearbeiten und Verwalten von Sequenzen 19

5. Arbeiten mit Docklight 21

5.1 Testen eines seriellen Geräts oder einer Protokollimplementierung 22

5.2 Ein serielles Gerät simulieren 23

5.3 Überwachung der seriellen Kommunikation zwischen zwei Geräten 25

5.4 Abfangen einer bestimmten Sequenz und Aufnehmen einer Momentaufnahme der Kommunikation 27

5.5 Protokollieren und Analysieren eines Tests 27

5.6 Prüfen auf Sequenzen mit zufälligen Zeichen (Empfangssequenz Wildcards) 28

5.7 Speichern und Laden von Projektdaten, Skript und Optionen 31

6. Arbeiten mit Docklight (Fortgeschrittene) 33

6.1 Senden von Befehlen mit Parametern (Sendesequenz-Wildcards) 34

6.2 Erhöhung der Verarbeitungsgeschwindigkeit und Vermeidung von "Input Buffer Overflow" Nachrichten 35

6.3 Wie man die beste Timing-Genauigkeit erreicht 36

6.4 Berechnen und Validieren von Prüfsummen 36

6.5 Steuerung und Überwachung von RS232-Handshake-Signalen 38

6.6 Erstellen und Erkennen von Verzögerungen zwischen Zeichen 42

6.7 Setzen und Erkennen eines "Break"-Status 44

6.8 Testen eines TCP-Server-Geräts (Skripting) 45

6.9 Überwachung einer Client/Server-TCP-Verbindung (Scripting) 46

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

7. Beispiele und Tutorials 49

7.1 Testen eines Modems - Beispielprojekt: ModemDiagnostics.ptp 50
7.2 Reagieren auf eine Empfangssequenz - Beispielprojekt: PingPong.ptp....51
7.3 Modbus RTU mit CRC-Prüfsumme - Beispielprojekt: ModbusRtuCrc.ptp....52

8. Beispiele und Tutorials (Scripting) 55

8.1 Automatisierte Modemtests - Beispielskript: ModemScript.pts....56
8.2 Starten von der Befehlszeile - Beispielskript: LogStartupScript.pts....59
8.3 Manipulation eines RS232-Datenstroms - Beispielskript: ZeichenManipulation.pts....60
8.4 TCP/IP-Kommunikation - Beispielprojekte PingPong_TCP_Server/Client.ptp....61

9. Referenz 62

9.1 Menü und Symbolleiste (Scripting)....63
9.2 Dialog: Sendesequenz bearbeiten 65
9.3 Dialog: Empfangssequenz bearbeiten....66
9.4 Dialog: Protokollierung starten / Protokolldatei(en) erstellen 67
9.5 Dialog: HTML-Ausgabe anpassen....68
9.6 Dialog: Sequenz suchen....69
9.7 Dialog: Parameter Sendesequenz 70
9.8 Dialog: Projekteinstellungen - Kommunikation....70
9.9 Dialog: Projekteinstellungen - Flusskontrolle....74
9.10 Dialog: Projekteinstellungen - Kommunikationsfilter....74
9.11 Dialog: Optionen....75
9.12 Dialog: Experten-Optionen 76
9.13 Tastatur-Konsole 78
9.14 Prüfsummen-Spezifikation....78

10. Referenz (Skripting) 82

10.1 VBScript-Grundlagen 83
10.1.1 Copyright-Hinweis 84
10.1.2 Strukturen der Kontrolle....84
10.1.2.1 Strukturen der Entscheidung 84
10.1.2.2 Schleifen-Strukturen 85
10.1.3 Variablen, Arrays, Konstanten und Datentypen 86
10.1.4 Betreiber....88
10.1.5 Datum/Uhrzeit-Funktionen....89
10.1.6 Sonstiges....91
10.2 Docklight Script-Befehle - Das DL-Objekt 93
10.2.1 Methoden....94
10.2.1.1 Kommentar hinzufügen 94

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.2.1.2 ClearCommWindows 95
10.2.1.3 GetReceiveCounter 95

10.2.1.4 GetDocklightTimeStamp 96
10.2.1.5 OpenProject 97
10.2.1.6 Pause 98
10.2.1.7 Beenden Sie 99
10.2.1.8 ResetEmpfangszähler 99
10.2.1.9 Sendesequenz 100
10.2.1.10 StartCommunication 102
10.2.1.11 StopCommunication 102
10.2.1.12 StartLogging 102
10.2.1.13 StopLogging 104
10.2.1.14 WaitForSequence 105
10.2.2 Methoden (Fortgeschrittene) 106
10.2.2.1 CalcChecksum 106
10.2.2.2 ConvertSequenceData 109
10.2.2.3 GetChannelSettings 111
10.2.2.4 GetChannelStatus 112
10.2.2.5 GetCommWindowData 114
10.2.2.6 GetEnvironment 114
10.2.2.7 GetHandshakeSignals 116
10.2.2.8 GetKeyState 117
10.2.2.9 GetReceiveComments 118
10.2.2.10 Eingabefeld2 119
10.2.2.11 MsgBox2 119
10.2.2.12 LoadProgramOptions 120
10.2.2.13 PlaybackLogFile 121
10.2.2.14 SaveProgramOptions 123
10.2.2.15 SetChannelSettings 124
10.2.2.16 SetContentsFilter 127
10.2.2.17 SetHandshakeSignals 128
10.2.2.18 SetUserOutput 129
10.2.2.19 SetWindowLayout 130
10.2.2.20 ShellRun 131
10.2.2.21 UploadDatei 133
10.2.3 Eigenschaften 134
10.2.3.1 NoOfSendSequences 134
10.2.3.2 NoOfReceiveSequences 135
10.3 OnSend / OnReceive Ereignisprozeduren 135
10.3.1 Sub DL_OnSend() - Manipulation der Sendesequenzdaten 136
10.3.2 Sub DL_OnReceive() - Auswertung der empfangenen Sequenzdaten 138
10.3.3 OnSend / OnReceive - Zeitablauf und Programmfluss 143
10.4 FileInput- / FileOutput-Objekte zum Lesen und Schreiben von Dateien 145
10.4.1 FileInput - Dateien lesen 145
10.4.2 FileOutput - Dateien schreiben 147
10.4.3 Mehrere Eingabedateien / Mehrere Ausgabedateien 148
10.5 Seitenkanäle - Verwendung mehrerer Datenverbindungen 148
10.5.1 OpenSideChannel / CloseSideChannel - Verwaltung mehrerer Kanäle 148
10.5.2 DirectSend 150

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.6 Debug-Objekt / Skript-Debugging 151

10.7 #include Richtlinie 153
10.8 Befehlszeilen-Syntax 153
10.9 Dialog: Anpassen / Externer Editor 154

11. Unterstützung 157
11.1 Web-Support und Fehlerbehebung 158
11.2 E-Mail-Unterstützung 158

12. Anhang 159
12.1 ASCII-Zeichensatztabellen 160
12.2 Tastenkombinationen 162
12.3 RS232 Steckverbinder / Pinout 164
12.4 Standard RS232-Kabel 166
12.5 Docklight Überwachungskabel RS232 SUB D9 169
12.6 Docklight Wasserhahn 170
12.7 Docklight Wasserhahn Pro / Wasserhahn 485 171

13. Glossar / Verwendete Begriffe 173
13.1 Aktion 174
13.2 Pause 174
13.3 Charakter 174
13.4 CRC 174
13.5 DCE 175
13.6 DTE 175
13.7 Durchflusskontrolle 175
13.8 HID 175
13.9 LIN 176
13.10 Modbus 176
13.11 Multidrop-Bus (MDB) 176
13.12 Benanntes Rohr 176
13.13 Empfangssequenz 176
13.14 RS232 177
13.15 RS422 177
13.16 RS485 177
13.17 Sendesequenz 178
13.18 Sequenz 178
13.19 Sequenz-Index 178
13.20 Server für serielle Geräte 179

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

13.21 Schnappschuss 179

13.22 TCP....179
13.23 Auslöser 179
13.24 UART....179
13.25 UDP....179
13.26 Virtuelles Nullmodem....180
13.27 Wildcard 180

Index 0

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Urheberrecht

1 Urheberrecht

Copyright 2002-2023 Flachmann und Heggelbacher GmbH & Co. KG und Kickdrive Software-Lösungen

Alle Rechte vorbehalten. Kein Teil dieses Werkes darf ohne schriftliche Genehmigung des Herausgebers in irgendeiner Form oder mit irgendwelchen Mitteln - grafisch, elektronisch oder mechanisch, einschließlich Fotokopien, Aufzeichnungen, Tonbandaufnahmen oder Informationsspeicher- und -abrufsystemen - vervielfältigt werden.

Markenzeichen

Produkte, auf die in diesem Dokument Bezug genommen wird, können entweder Marken und/oder eingetragene Marken der jeweiligen Eigentümer sein. Der Herausgeber und der Autor erheben keinen Anspruch auf diese Marken.

Microsoft und *Windows* sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den Vereinigten Staaten und/oder anderen Ländern.

Haftungsausschluss

Obwohl bei der Erstellung dieses Dokuments alle Vorsichtsmaßnahmen getroffen wurden, übernehmen der Herausgeber und der Autor keine Verantwortung für Fehler oder Auslassungen oder für Schäden, die sich aus der Verwendung der in diesem Dokument enthaltenen Informationen oder aus der Verwendung von Programmen und Quellcode ergeben, die diesem Dokument möglicherweise beigefügt sind. In keinem Fall haften der Herausgeber und der Autor für entgangenen Gewinn oder andere kommerzielle Schäden, die direkt oder indirekt durch dieses Dokument verursacht wurden oder angeblich verursacht wurden.

Kontakt

E-Mail-Unterstützung: support@docklight.de
www.docklight.de

Flachmann und Heggelbacher GmbH & Co. KG
Waldkirchbogen 27
D-82061 Neuried
Deutschland
www.fuh-edv.de

Kickdrive Software Solutions e.K.
Robert-Bosch-Str. 5
D-88677 Markdorf
Deutschland
www.kickdrive.de

Einführung

2 Einführung

2.1 Docklight - Übersicht

Docklight ist ein Test-, Analyse- und Simulationswerkzeug für serielle Kommunikationsprotokolle (RS232, RS485/422 und andere). Es ermöglicht Ihnen, die Kommunikation zwischen zwei seriellen Geräten zu überwachen oder die serielle Kommunikation eines einzelnen Geräts zu testen. Docklight ist einfach zu bedienen und funktioniert auf fast jedem Standard-PC mit *Windows 11*, *Windows 10, Windows 8* oder *Windows 7*.

Zu den wichtigsten Funktionen von Docklight gehören

- **Simulation serieller Protokolle** - Docklight kann je nach verwendetem Protokoll benutzerdefinierte Sequenzen aussenden und auf eingehende Sequenzen reagieren. Dies ermöglicht es, das Verhalten eines seriellen Kommunikationsgeräts zu simulieren, was besonders nützlich ist, um Testbedingungen zu erzeugen, die mit dem Originalgerät nur schwer reproduzierbar sind (z. B. Problemzustände).
- **Protokollierung von RS232-Daten** - Alle seriellen Kommunikationsdaten können in zwei verschiedenen Dateiformaten protokolliert werden. Verwenden Sie das reine Textformat für die schnelle Protokollierung und Speicherung großer Datenmengen. Ein HTML-Dateiformat mit gestyltem Text lässt Sie leicht zwischen eingehenden und ausgehenden Daten oder zusätzlichen Informationen
- unterscheiden. Docklight kann auch jeden binären Datenstrom protokollieren, einschließlich ASCII 0 <NUL>-Bytes und anderer Steuerzeichen. **Erkennung spezifischer Datensequenzen** - In vielen Testfällen müssen Sie nach einer bestimmten Sequenz innerhalb der RS232-Daten suchen, die auf einen Problemzustand hinweist.
  Docklight verwaltet eine Liste solcher Datensequenzen für Sie und kann nach dem Erkennen einer Sequenz benutzerdefinierte Aktionen durchführen, z.B. einen Schnappschuss aller Kommunikationsdaten vor und nach dem Erhalt der
- Fehlermeldung machen.
  Antworten auf eingehende Daten - Docklight ermöglicht es Ihnen, benutzerdefinierte Antworten auf die verschiedenen empfangenen Kommunikationssequenzen festzulegen. So können Sie innerhalb weniger Minuten einen einfachen Simulator für Ihr serielles Gerät erstellen. Es kann Ihnen auch helfen, einen bestimmten Fehler zu finden, indem Sie nach dem Empfang der Fehlermeldung einen Diagnosebefehl senden.

Docklight arbeitet mit den von Ihrem Betriebssystem bereitgestellten COM-Schnittstellen. Physikalisch handelt es sich bei diesen Schnittstellen in vielen Fällen um RS232 SUB D9 Schnittstellen. Es ist jedoch auch möglich, Docklight für andere Kommunikationsstandards wie RS485 und RS422 zu verwenden, die einen anderen elektrischen Aufbau als RS232 haben, aber dem RS232-Kommunikationsmechanismus folgen.

Docklight wurde auch erfolgreich mit vielen gängigen USB-zu-Seriell-Konvertern, seriellen Bluetooth-Ports, GPS-Empfängern, virtuellen Nullmodems, Arduino, MicroPython/pyboard oder anderen Embedded/UART-Boards getestet, die einen COM-Port in Windows hinzufügen.

Für RS232-Vollduplex-Überwachungsanwendungen empfehlen wir unser Docklight Tap USB-Zubehör oder unser Docklight Monitoring Cable.

In diesem Handbuch wird nur auf die seriellen RS232-Verbindungen eingegangen, da dies die Grundlage für die anderen oben genannten seriellen Verbindungen ist.

TIPP: Schauen Sie sich für den Einstieg die Docklight-Beispielprojekte an, die einige der grundlegenden Docklight-Funktionen demonstrieren.

2.2 Docklight Scripting - Überblick

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Docklight Scripting ist eine erweiterte Version von Docklight RS232 Terminal / RS232 Monitor. Sie bietet eine einfach zu bedienende Skriptsprache und einen eingebauten Editor zum Erstellen und Ausführen von automatisierten Testaufträgen. Mit einem Docklight-Skript können Sie alle grundlegenden Docklight

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Operationen (Senden von vordefinierten Datensequenzen, Erkennen bestimmter Sequenzen im eingehenden Datenstrom, ...) und betten sie in Ihren eigenen Testcode ein.

Docklight Scripting ist netzwerkfähig. Anstelle eines seriellen COM-Ports kann Docklight Scripting TCP-Verbindungen aufbauen (TCP-Client-Modus), eine TCP-Verbindung an einem lokalen Port annehmen (TCP-Server-Modus) oder als UDP-Peer agieren. Es unterstützt auch USB-HID-Verbindungen und Named Pipes.

Docklight Scripting bietet Ihnen sowohl Flexibilität als auch Einfachheit. Innerhalb von Minuten können Sie Ihre eigenen automatisierten Testwerkzeuge entwickeln und erstellen:

- **zeitgesteuerte Prüfaufträge** (z. B. Senden eines Diagnosebefehls alle 5 Minuten und Meldung eines Fehlers, wenn die Antwort des Geräts nicht OK ist)
- **wiederholte Testzyklen** (z. B. Dauertest für ein Bewegungssteuerungs-
- /Antriebssystem) **automatische Skripte zur Gerätekonfiguration** (z. B. Zurücksetzen eines RS232-Geräts auf die Werkseinstellungen vor der Auslieferung)
- **Fehleranalysewerkzeuge für Wartungs- und Instandhaltungsaufgaben** (z. B. Ausführung einer Reihe von Diagnosebefehlen und Durchführung einer
- automatischen Fehleranalyse)
  Protokolltester mit automatischer Prüfsummenberechnung (z. B. CRC - Cyclic Redundancy Codes)
- **Docklight-Startskripte** (z.B. automatischer Start einer COM-Port-Protokollierungsaufgabe beim PC-Start)

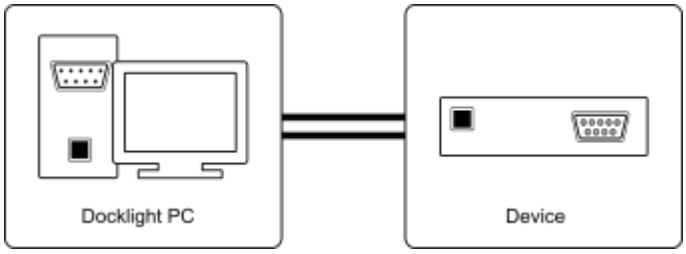
Docklight Scripting verwendet die VBScript-Engine, die es Ihnen ermöglicht, Ihre Tests in einer einfachen, bekannten Skriptsprache zu schreiben. Die grundlegenden Funktionen und Eigenschaften von Docklight werden durch einen kleinen und praktischen Satz von Docklight-Skriptbefehlen verfügbar gemacht.

TIPP: Für den Einstieg sehen Sie sich das Docklight-Modemtest-Skript an, das die Verwendung von Docklight-Skriptbefehlen für einen automatisierten Modemtest demonstriert. Eine einfache Demonstration der TCP/IP-Fähigkeiten finden Sie im TCP-Client/Server-Beispiel .

2.3 Typische Anwendungen

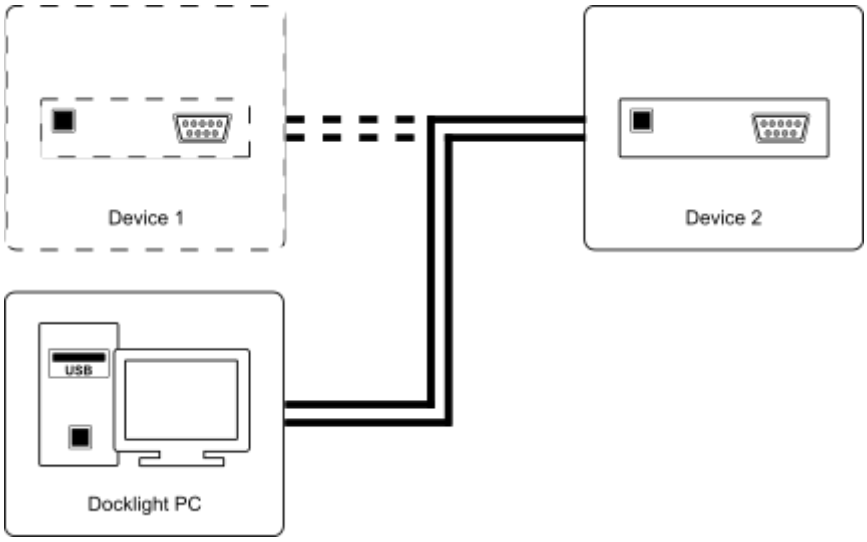
Docklight ist das ideale Werkzeug zur Unterstützung Ihres Entwicklungs- und Testprozesses für serielle Kommunikationsgeräte. Docklight kann verwendet werden für

- Testen Sie die Funktionalität oder die Protokollimplementierung eines seriellen Geräts.
  Sie können von Ihrem Gerät erkannte Steuersequenzen definieren, diese senden, die Antworten protokollieren und analysieren und die Reaktion des Geräts testen.

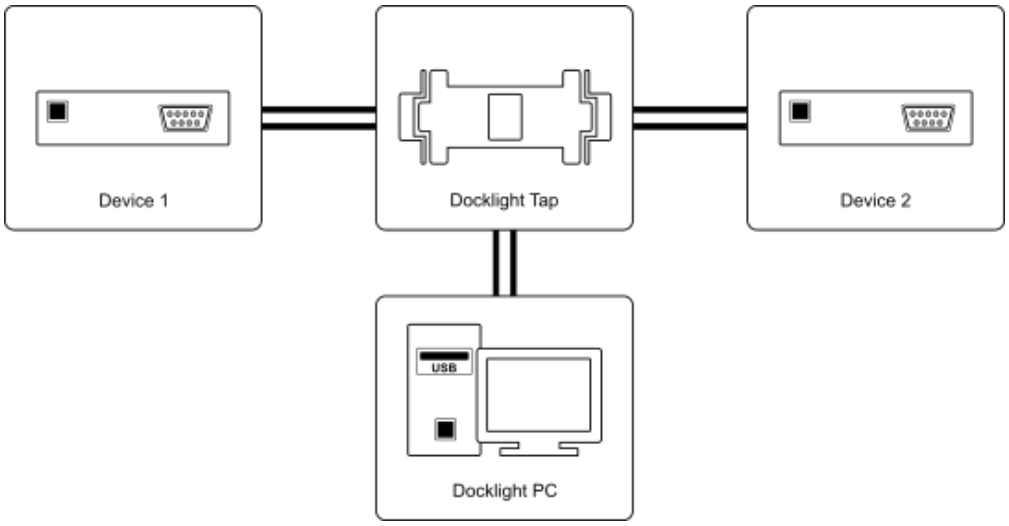


- Simulieren Sie ein serielles Gerät.
  Obwohl selten, muss die Möglichkeit eines Hardware-Fehlers in den meisten Systemen berücksichtigt werden. Stellen Sie sich vor, Sie haben ein Gerät, das im Falle eines Hardwarefehlers eine Fehlermeldung sendet. Ein zweites Gerät soll diese Fehlermeldung empfangen und irgendeine Reaktion ausführen. Mit Docklight können Sie die zu sendende Fehlermeldung leicht simulieren und die Reaktion des zweiten Geräts testen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de



- Überwachen Sie die Kommunikation zwischen zwei Geräten.
  Schalten Sie Docklight in die Kommunikationsverbindung zwischen zwei seriellen Geräten ein. Überwachen und protokollieren Sie die serielle Kommunikation in beide Richtungen. Erkennen Sie fehlerhafte Kommunikationssequenzen oder spezielle Fehlerbedingungen innerhalb der überwachten Kommunikation. Erstellen Sie einen Schnappschuss der Kommunikation, wenn eine solche Fehlerbedingung aufgetreten ist.



2.4 Systemanforderungen

Das Betriebssystem
- *Windows 11, Windows 10, Windows 10 x64, Windows 8, Windows 8 x64, Windows 7, Windows 7 x64.*

Zusätzliche Anforderungen
- Für RS232-Tests oder Simulationen: Mindestens ein COM-Anschluss verfügbar. Zwei COM-Anschlüsse zur Überwachung der Kommunikation zwischen zwei
- seriellen Geräten.
  Für die Überwachung mit niedriger Latenz mit Docklight Tap, Docklight Tap Pro oder Docklight Tap 485: Ein USB-Anschluss.
- Für Docklight Scripting TCP oder UDP Anwendungen: Netzwerk mit IPv4-Adressierung.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Für den Anschluss der zu prüfenden Geräte können zusätzliche Kabel oder Softwaretreiber erforderlich sein. Siehe die Abschnitte über Docklight Tap, Docklight Monitoring Cable RS232 SUB D9, Standard RS232 Kabel und virtuelle Nullmodemtreiber.

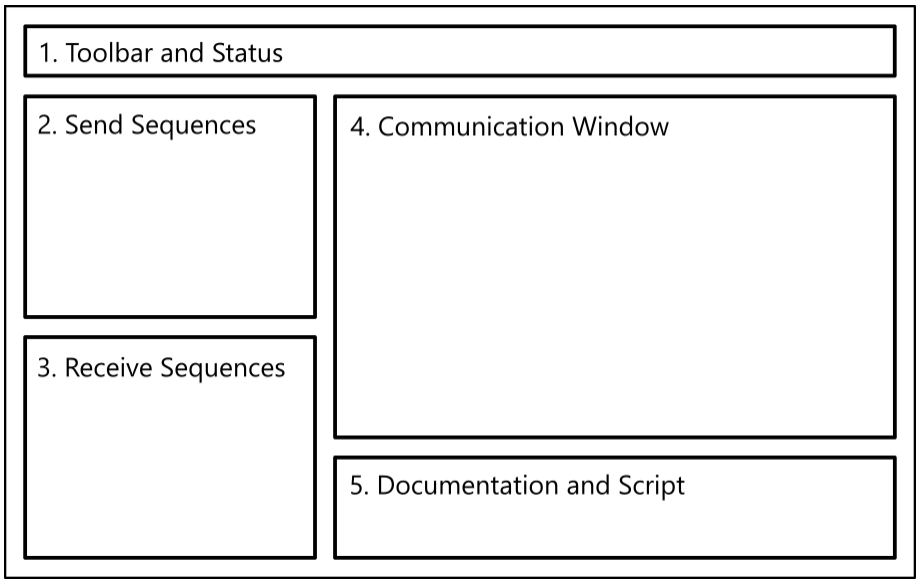
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Benutzeroberf
läche

3 Benutzeroberfläche

3.1 Hauptfenster (Scripting)

Das Hauptfenster von Docklight Scripting ist in fünf Bereiche unterteilt:



1. Symbolleiste und Status
Sie können alle wichtigen Docklight-Funktionen über die Symbolleiste auswählen. Die Statuszeile darunter zeigt zusätzliche Informationen über den Kommunikationsstatus und die aktuellen Einstellungen an.

2. Sequenzen senden
Definieren, bearbeiten und verwalten Sie hier Ihre Sendesequenzen. Verwenden Sie das Pfeilsymbol oder die Leertaste, um die ausgewählte Sequenz zu versenden. Doppelklicken Sie auf das leere Feld am Ende einer Liste, um eine neue Sequenz zu erstellen. Über ein Kontextmenü (rechte Maustaste) können Sie ganze Sendesequenzen ausschneiden, kopieren oder in die Zwischenablage einfügen. Siehe Bearbeiten und Verwalten von Sequenzen und Dialog: Sendesequenz bearbeiten für weitere Informationen.

3. Empfangene Sequenzen
Definieren, bearbeiten und verwalten Sie hier Ihre Empfangssequenzen. Doppelklicken Sie auf das leere Feld am Ende einer Liste, um eine neue Sequenz zu erstellen. Die Liste der Empfangssequenzen unterstützt die gleichen Operationen zur Neuordnung und Zwischenablage wie die Liste der Sendesequenzen. Sie können auch eine Sendesequenz in die Zwischenablage kopieren und in die Liste der Empfangssequenzen einfügen.
Siehe Bearbeiten und Verwalten von Sequenzen und Dialog: Empfangssequenz bearbeiten für weitere Informationen.

Sie können die Sequenzlisten per Drag&Drop neu anordnen: Erlauben Sie zunächst das Umsortieren der Liste, indem Sie auf das kleine Schloss-Symbol in der linken oberen Ecke klicken. Wenn nicht gesperrt ist, kann die Liste geändert werden, indem eine Sequenz mit gedrückter linker Maustaste an eine neue Position gezogen wird.

Durch Klicken auf die Markierung **|<** können Sie den Bereich Sende-/Empfangssequenzen minimieren.

4. Fenster Kommunikation
Zeigt die ausgehende und eingehende Kommunikation der seriellen Datenverbindung an. Für die Kommunikationsdaten stehen verschiedene

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Anzeigeoptionen zur Verfügung, einschließlich ASCII / HEX / Dezimal / Binär-Anzeige, Zeitstempel und Hervorhebung (siehe Optionen). Wenn die serielle Kommunikation gestoppt wird, können alle Daten aus dem Kommunikationsfenster in die Zwischenablage kopiert oder gedruckt werden. Sie können auch nach bestimmten Sequenzen suchen, indem Sie die Suchfunktion

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Sequenzfunktion. Weitere Informationen finden Sie unter Wie werden serielle Daten verarbeitet und angezeigt?

5.
5a. Dokumentation von Projekten und Abläufen
Geben Sie zusätzliche Kommentare zu Ihrem Projekt oder zu einer bestimmten Sendesequenz / Empfangssequenz ein. Docklight zeigt die sequenzspezifische Dokumentation an, wenn Sie eine Sendesequenz oder Empfangssequenz aus der Liste auswählen (2. und 3.). Docklight wechselt zur Hauptdokumentation des Projekts, wenn Sie auf die leere untere Zeile der Sequenzliste klicken oder wenn Sie in das Kommunikationsfenster klicken (4.).

Um ein versehentliches Bearbeiten zu vermeiden, ist der Dokumentationsbereich standardmäßig gesperrt, und Sie müssen die Bearbeitung aktivieren, indem Sie auf das kleine Schlosssymbol über dem Bereich klicken. Wenn entsperrt ist, können Sie seinen Inhalt frei bearbeiten/kopieren/einfügen/löschen.

5b. Schrift
Bearbeiten Sie hier Ihren Docklight-Skriptcode. Über ein Kontextmenü (rechte Maustaste) können Sie Code ausschneiden, kopieren, einfügen, löschen oder suchen/ersetzen. Für fortgeschrittene Editierfunktionen gibt es Unterstützung für externe Editoren. Weitere Informationen zur Erstellung von Docklight-Skripten finden Sie in der Docklight Scripting Reference.

Durch Klicken auf das **v** auf der rechten Seite können Sie den Bereich Dokumentation/Skript minimieren.

3.2 Zwischenablage - Ausschneiden, Kopieren und Einfügen

Docklight unterstützt die Operationen Ausschneiden/Kopieren/Einfügen. Zwischenablageoperationen sind in der

- Hauptfenster - Sendesequenzen
- Hauptfenster - Empfangen von Sequenzen
- Hauptfenster - Kommunikation
- Hauptfenster - Dokumentation
- Hauptfenster - Skript-Editor (nur Docklight Scripting)
- Dialog: Sendesequenz bearbeiten
- Dialog: Empfangssequenz bearbeiten
- Dialog: Sequenz suchen
- Dialog: Parameter Sendefolge
- Bereich Dokumentation
- Tastatur-Konsole

Sie können eine serielle Datensequenz aus dem Kommunikationsfenster ausschneiden und eine neue Sende- oder Empfangssequenz erstellen, indem Sie sie in die entsprechende Liste einfügen. Oder Sie bearbeiten eine Sendesequenz, kopieren einen Teil dieser Sequenz in die Zwischenablage und erstellen daraus eine neue Empfangssequenz, indem Sie sie in das Fenster "Empfangssequenz" einfügen.

TIPP: Verwenden Sie das Kontextmenü der **rechten Maustaste** für die Operationen Ausschneiden/Kopieren/Einfügen oder den entsprechenden Tastatur-Hotkey.

3.3 Bereich Dokumentation

Docklight bietet Dokumentationsbereiche im unteren rechten Teil des Hauptfensters und in den Dialogen Sendesequenz bearbeiten oder Empfangssequenz bearbeiten an.

Sie können diese Bereiche nutzen, um zusätzliche Hinweise zu Ihrer Docklight-Anwendung zu notieren. Zum Beispiel, wie Sie die Sende-/Empfangssequenzen und Sequenzparameter verwenden, oder Hinweise auf zusätzliche Testgeräte, usw.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Die Dokumentationsinhalte werden zusammen mit allen anderen Docklight-Projekteinstellungen gespeichert und geladen (siehe Speichern und Laden Ihrer Projektdaten, Skripte und Optionen).

TIPP: Die Dokumentationsbereiche sind einfache Textfelder ohne Formatierungsmenüs oder Werkzeuge. Für eine formatierte Dokumentation mit Bildern und Tabellen können Sie Ihre Dokumentation in *Windows WordPad* oder *Microsoft Word* vorbereiten und per Copy & Paste in den Docklight-Dokumentationsbereich einfügen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Merkmale und Funktionen

4 Merkmale und Funktionen

4.1 Wie serielle Daten verarbeitet und angezeigt werden

Docklight verarbeitet alle seriellen Daten auf eine 8-Bit-orientierte Weise. Jede Folge von seriellen Daten besteht aus einem oder mehreren 8-Bit-Zeichen. Mit Docklight können Sie

- die seriellen Daten entweder im ASCII-, HEX-, Dezimal- oder Binärformat anzeigen
- serielle Daten in die Zwischenablage kopieren und in eine Standardtextdatei oder eine formatierte Datei einfügen
  Microsoft® Word-Dokument, oder erstellen Sie eine Sende-/Empfangssequenz unter Verwendung der Daten.

- Ausdruck von seriellen Daten, Benutzerkommentaren und anderen Informationen

Das Kommunikationsfenster von Docklight zeigt die aktuelle Kommunikation über die ausgewählte(n) serielle(n) Schnittstelle(n) an. Docklight unterscheidet zwischen zwei Kommunikationskanälen (Kanal 1 und Kanal 2), die im Sende-/Empfangsmodus die ein- und ausgehenden Daten oder im Überwachungsmodus die beiden zu beobachtenden Kommunikationskanäle darstellen. Die Daten von Kanal 1 und Kanal 2 werden in unterschiedlichen Farben oder Schriftarten angezeigt. Die Kommunikationsdaten können gedruckt oder als Protokolldatei im Text- oder HTML-Format gespeichert werden.

Neben den seriellen Daten fügt Docklight auch Datums- und Zeitstempel in die Kommunikationsanzeige ein. Standardmäßig wird ein Datums-/Zeitstempel jedes Mal eingefügt, wenn die Datenflussrichtung zwischen Kanal 1 und Kanal 2 wechselt oder bevor eine neue Sendesequenz übertragen wird. Es stehen mehrere Optionen zur Verfügung, um zusätzliche Zeitstempel einzufügen. Dies ist besonders nützlich bei der Überwachung einer Halbduplex-Leitung mit nur einem Kommunikationskanal. Siehe Optionen --> Datum/Zeitstempel

Docklight ist in der Lage, serielle Datenströme zu verarbeiten, die einen beliebigen ASCII-Code von 0 - 255 dezimal enthalten. Da es nicht druckbare Steuerzeichen (ASCII-Code < 32) und verschiedene Kodierungen für ASCII-Code > 127 gibt, können nicht alle diese Zeichen im ASCII-Textfenster angezeigt werden. Dennoch werden alle Zeichen von Docklight korrekt verarbeitet und können im HEX-, Dezimal- oder Binärformat angezeigt werden. Docklight verarbeitet die seriellen Daten auf allen Sprachversionen des Windows-Betriebssystems auf die gleiche Weise, auch wenn die ASCII-Anzeige unterschiedlich sein kann. Für Steuerzeichen (ASCII-Code < 32) steht eine zusätzliche Anzeigeoption zur Verfügung, um deren Textäquivalent im Kommunikationsfenster anzuzeigen. Siehe Dialog Optionen und Anhang, ASCII-Zeichensatztabellen.

Docklight ermöglicht es Ihnen, alle seriellen Originaldaten zu unterdrücken, wenn Sie einen Test durchführen, bei dem Sie nicht die eigentlichen Daten sehen müssen, sondern nur die zusätzlichen Auswertungen, die mit Hilfe von Empfangssequenzen erzeugt werden. Siehe dazu die Projekteinstellungen für Kommunikationsfilter.

4.2 Bearbeiten und Verwalten von Sequenzen

Ein Docklight-Projekt besteht hauptsächlich aus benutzerdefinierten Sequenzen. Dabei kann es sich entweder um Sendesequenzen handeln, die von Docklight selbst gesendet werden, oder um Empfangssequenzen, die dazu dienen, eine spezielle Nachricht in den eingehenden seriellen Daten zu erkennen.

Sequenzen werden mit Hilfe des Dialogfensters Sendesequenz bearbeiten oder Empfangssequenz bearbeiten definiert. Dieses Dialogfenster wird geöffnet
1. indem Sie im Kontextmenü der **rechten Maustaste** die Option **Bearbeiten** wählen.
2. durch Doppelklick auf eine vorhandene Sequenz oder durch Drücken von **Strg + E** bei ausgewählter Liste "Sequenz senden" oder "Sequenz

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

empfangen".
3. beim Erstellen einer neuen Sequenz durch Doppelklick auf das leere Feld am Ende einer Liste (oder durch Drücken von **Strg + E**).
4. wenn eine neue Sequenz in die Sequenzliste eingefügt wird.

Docklight unterstützt die Verwendung von Wildcards (z.B. Wildcard "?" als Platzhalter für ein beliebiges Zeichen) in Empfangs- und Sendesequenzen. Siehe die Abschnitte

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

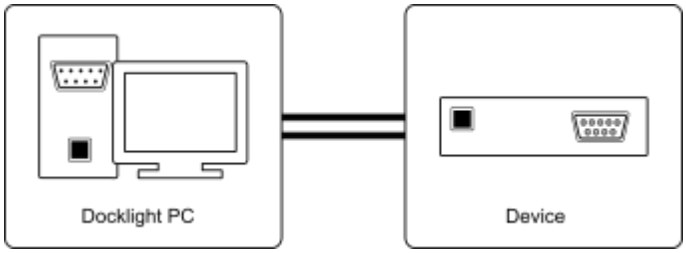
Senden von Befehlen mit Parametern und Prüfung auf Sequenzen mit zufälligen Zeichen für Details und Beispiele.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Arbeiten mit Docklight

5 Arbeiten mit Docklight

5.1 Testen eines seriellen Geräts oder einer Protokollimplementierung



Voraussetzungen

- Sie benötigen die Spezifikation des zu prüfenden Protokolls, z.B. in schriftlicher Form.
- Das zu testende serielle Gerät sollte an einen der COM-Anschlüsse des PCs angeschlossen werden. Siehe Abschnitt Standard-RS232-Kabel für Details zum Anschluss von zwei seriellen Geräten.
- Das serielle Gerät muss betriebsbereit sein.

Durchführung des Tests

A) Ein neues Projekt erstellen
Erstellen Sie ein neues Docklight-Projekt, indem Sie das Menü **Datei >** **Neues Projekt** wählen.

B) Einstellen der Kommunikationsoptionen

1. Wählen Sie das Menü **Extras >** **Projekteinstellungen...**
2. Wählen Sie den Kommunikationsmodus **Senden/Empfangen**
3. Stellen Sie unter **Senden/Empfangen auf dem Kommunikationskanal** den COM-Port ein, an dem Ihr serielles Gerät angeschlossen ist.
4. Stellen Sie die Baudrate und alle anderen erforderlichen **COM-Port-Einstellungen** ein.
5. Bestätigen Sie die Einstellungen und schließen Sie das Dialogfeld, indem Sie auf die Schaltfläche **OK** klicken.

C) Definieren der zu verwendenden Sendesequenzen
Sie werden Ihr serielles Gerät wahrscheinlich testen, indem Sie bestimmte Sequenzen entsprechend dem vom Gerät verwendeten Protokoll senden und die Reaktion des Geräts beobachten. Führen Sie die folgenden Schritte aus, um Ihre Liste von Sequenzen zu erstellen:

1. Doppelklicken Sie auf die letzte Zeile der Tabelle Sendesequenzen. Der Dialog Sendesequenz bearbeiten wird angezeigt (siehe auch Sequenzen bearbeiten und verwalten).
2. Geben Sie einen **Namen** für die Sequenz ein. Der Sequenzname sollte für jede definierte Sendesequenz eindeutig sein.
3. Geben Sie die **Sequenz** selbst ein. Sie können die Sequenz entweder im ASCII-, HEX-, Dezimal- oder Binärformat eingeben. Mit den Optionsfeldern für den **Bearbeitungsmodus** können Sie jederzeit zwischen den verschiedenen Formaten umschalten.
4. Nachdem Sie auf die Schaltfläche **OK** geklickt haben, wird die neue Sequenz zu den Listen der Sendesequenzen hinzugefügt.

Wiederholen Sie die Schritte 1 - 4, um die anderen Sendesequenzen zu definieren, die Sie für die Durchführung Ihres Tests benötigen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

D) Festlegen der verwendeten Empfangssequenzen
Wenn Sie möchten, dass Docklight beim Empfang bestimmter Sequenzen reagiert, müssen Sie eine Liste von Empfangssequenzen definieren.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

1. Doppelklicken Sie auf die letzte Zeile der Tabelle Empfangssequenzen. Der Dialog Empfangsreihenfolge bearbeiten wird angezeigt. Der Dialog besteht aus drei Teilen: Namensfeld, Sequenzfeld und Aktionsfeld.
2. Bearbeiten Sie die Felder **Name** und **Sequenz**.
3. Geben Sie eine Aktion an, die ausgeführt werden soll, nachdem die Sequenz von Docklight empfangen wurde. Es sind vier Arten von Aktionen möglich:
   Antwort - Senden Sie nach dem Empfang der Sequenz eine der Sendesequenzen.
   Kommentar - Fügen Sie nach dem Empfang der Sequenz einen benutzerdefinierten Kommentar in das Kommunikationsfenster (und die Protokolldatei, falls vorhanden) ein.
   Trigger - Dies ist eine fortgeschrittene Funktion, die unter Auffangen einer bestimmten Sequenz...
   Stop - Nach dem Empfang der Sequenz beendet Docklight die Kommunikation.
4. Klicken Sie auf die Schaltfläche **OK**, um die neue Sequenz in die Liste aufzunehmen.

Wiederholen Sie die Schritte 1 - 4, um die anderen Empfangssequenzen zu definieren, die Sie für die Durchführung Ihres Tests benötigen.

E) Speichern des Projekts
Vor der Durchführung des eigentlichen Tests empfiehlt es sich, die festgelegten Kommunikationseinstellungen und Sequenzen zu speichern. Dies geschieht über das Menü **Datei > Projekt speichern**.

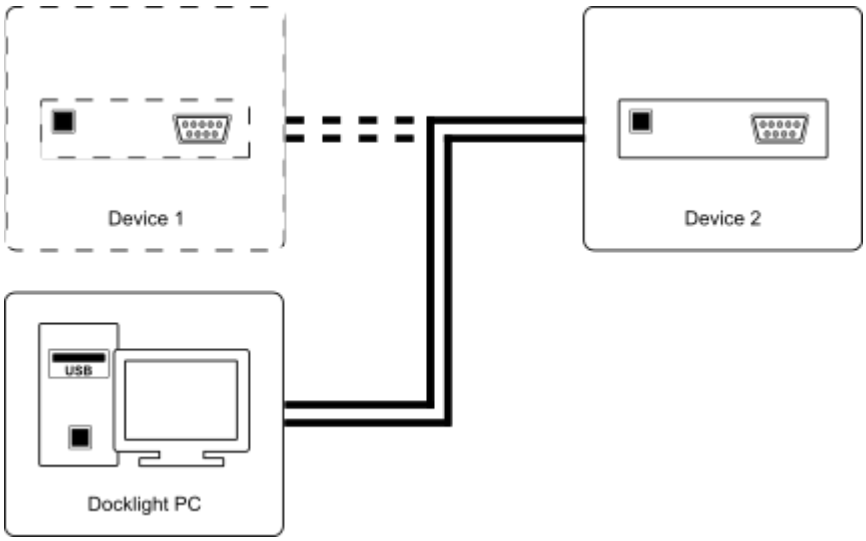
F) Durchführung des Tests
Starten Sie Docklight, indem Sie **Ausführen > Kommunikation starten** wählen.

Docklight öffnet eine serielle Verbindung gemäß den angegebenen Parametern. Es zeigt dann alle ein- und ausgehende Kommunikation im Kommunikationsfenster an.

Verwenden Sie die Schaltfläche ----> **Senden**, um eine der definierten Sequenzen an das serielle Gerät zu senden. Die Anzeige aller Datenübertragungen auf dem Bildschirm ermöglicht es Ihnen, das Verhalten des Geräts zu überprüfen. Alle Protokollinformationen können zur weiteren Analyse in einer Textdatei protokolliert werden. Siehe Abschnitt Protokollieren und Analysieren eines Tests.

TIPP: Über den Dokumentationsbereich können Sie ganz einfach zusätzliche Notizen machen oder Teile des Kommunikationsprotokolls kopieren und einfügen, um sie weiter zu dokumentieren.

5.2 Ein serielles Gerät simulieren



Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Voraussetzungen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

- Sie benötigen die Spezifikation des Verhaltens des seriellen Geräts, das Sie simulieren möchten, z. B. welche Art von Informationen nach Erhalt eines bestimmten Befehls zurückgesendet werden.
- Ein zweites Gerät wird an einen PC-COM-Port angeschlossen, der mit Ihrem Simulator kommuniziert.

Dieses zweite Gerät und sein Verhalten ist das eigentliche Objekt von Interesse. Ein Beispiel könnte ein Gerät sein, das regelmäßig den Status einer USV (Unterbrechungsfreie Stromversorgung) über ein serielles Kommunikationsprotokoll überprüft. Sie könnten Docklight verwenden, um das grundlegende Verhalten der USV und bestimmte USV-Problemfälle zu simulieren. Dies ist sehr nützlich, wenn man das andere Gerät testet, da es ziemlich schwierig sein kann, einen Alarmzustand (wie eine schlechte Batterie) bei der echten USV zu reproduzieren.

HINWEIS: Das zweite Gerät kann auch eine zweite Softwareanwendung sein. Es ist möglich, sowohl Docklight als auch die Softwareanwendung auf demselben PC laufen zu lassen. Verwenden Sie einfach für jede der beiden Anwendungen einen anderen COM-Port und verbinden Sie die beiden COM-Ports mit einem RS232-Nullmodemkabel. Sie können zu diesem Zweck auch ein virtuelles Nullmodem verwenden.

Durchführung des Tests

A) Ein neues Projekt erstellen

Erstellen Sie ein neues Docklight-Projekt, indem Sie das Menü **Datei >** **Neues Projekt** wählen.

B) Einstellen der Kommunikationsoptionen

1. Wählen Sie das Menü **Extras >** **Projekteinstellungen...**
2. Wählen Sie den Kommunikationsmodus **Senden/Empfangen**
3. Stellen Sie unter **Senden/Empfangen auf dem Kommunikationskanal** den COM-Port ein, an dem Ihr serielles Gerät angeschlossen ist.
4. Stellen Sie die Baudrate und alle anderen erforderlichen **COM-Port-Einstellungen** ein.
5. Bestätigen Sie die Einstellungen und schließen Sie das Dialogfeld, indem Sie auf die Schaltfläche **OK** klicken.

C) Festlegen der verwendeten Sendesequenzen

Definieren Sie alle Reaktionen Ihres Simulators. Denken Sie an Reaktionen, wenn sich das simulierte Gerät im Normalzustand befindet, und an Reaktionen im Fehlerzustand. In dem oben erwähnten USV-Beispiel wäre ein Batterieausfall ein solcher Problemfall, der mit dem Originalgerät nur schwer zu reproduzieren ist. Um zu testen, wie andere Geräte auf einen Batterieausfall reagieren, definieren Sie die entsprechende Antwortsequenz, die Ihre USV in diesem Fall senden würde.

HINWEIS: Siehe Testen eines seriellen Geräts..., um zu erfahren, wie Sie Sendesequenzen definieren können.

D) Festlegen der verwendeten Empfangssequenzen

In den meisten Fällen wird Ihr simuliertes Gerät keine unaufgeforderten Daten senden, sondern von dem anderen Gerät abgefragt werden. Das andere Gerät wird eine Reihe von vordefinierten Befehlssequenzen verwenden, um verschiedene Arten von Informationen anzufordern. Definieren Sie hier die Befehlssequenzen, die von Ihrem Simulator interpretiert werden müssen.

Geben Sie für jede definierte Befehlssequenz **Antwort** als Aktion an. Wählen Sie eine der unter C) definierten Sequenzen. Wenn Sie zwei oder mehr alternative Antwortsequenzen verwenden möchten, erstellen Sie mehrere Kopien derselben Empfangssequenz und geben Sie ihnen einen unterschiedlichen Namen (z. B. "status cmd - answer ok", "status cmd - answer battery failure", "status cmd - Antwort Netzausfall") und weisen Sie verschiedene Sendesequenzen als Aktion zu.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Im Beispiel hätten Sie drei Elemente in der Liste Empfangssequenzen, die auf denselben Befehl mit drei verschiedenen Antworten reagieren würden. Während des Tests können Sie entscheiden, welche Antwort gesendet werden soll, indem Sie die Listenelemente in der Spalte **Aktiv** aktivieren oder deaktivieren.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

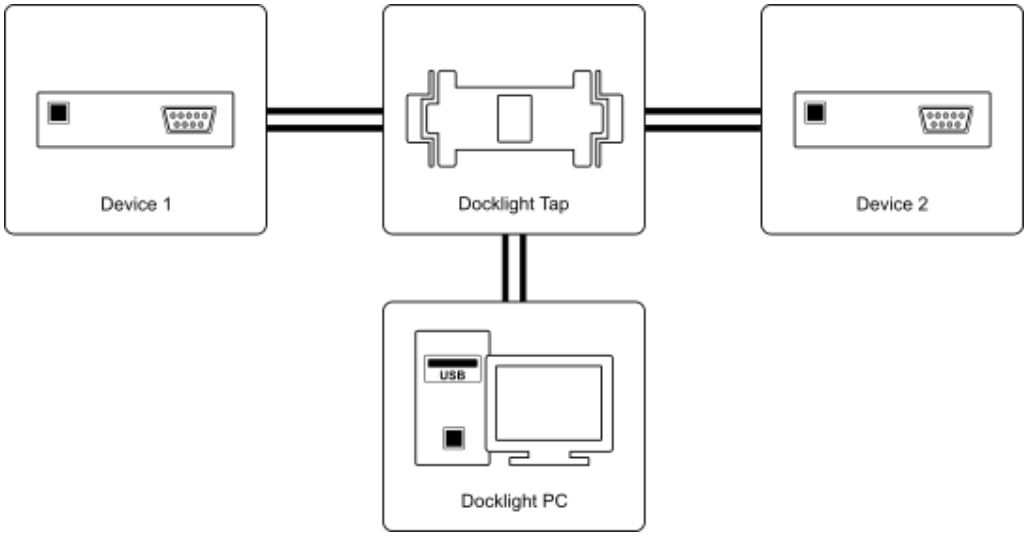
E) Speichern des Projekts
Vor der Durchführung des eigentlichen Tests empfiehlt es sich, die festgelegten Kommunikationseinstellungen und Sequenzen zu speichern. Dies geschieht über das Menü **Datei > Projekt speichern**.

F) Durchführung des Tests
Starten Sie Docklight, indem Sie **Ausführen > Kommunikation starten** wählen.

Docklight reagiert nun auf alle Befehle, die von dem angeschlossenen seriellen Gerät empfangen werden. Die Datenübertragungsanzeige auf dem Bildschirm ermöglicht es Ihnen, den Kommunikationsfluss zu überwachen. Alle Protokollinformationen können zur weiteren Analyse in einer Textdatei gespeichert werden. Siehe Abschnitt Protokollieren und Analysieren eines Tests.

TIPP: Im Dokumentationsbereich können Sie ganz einfach zusätzliche Notizen machen oder Teile des Kommunikationsprotokolls zur weiteren Dokumentation kopieren und einfügen.

5.3 Überwachung der seriellen Kommunikation zwischen zwei Geräten



Voraussetzungen

- Ein Docklight Monitoring Cable, Docklight Tap oder Docklight Tap Pro/485 wird benötigt, um die RS232-Sendesignale der beiden seriellen Geräte abzuhören und in Docklight einzuspeisen, ohne die Kommunikation zwischen den Geräten zu
- stören.
  Für die Einrichtung eines Docklight-Überwachungskabels müssen zwei COM-Anschlüsse an Ihrem PC für die Überwachung verfügbar sein. Jeder Anschluss empfängt die Daten von einem der zu überwachenden seriellen Geräte.
- Gerät 1 und Gerät 2 müssen betriebsbereit sein.

Durchführung des Tests

A) Ein neues Projekt erstellen
Erstellen Sie ein neues Docklight-Projekt, indem Sie das Menü **Datei > Neues Projekt** wählen.

B) Einstellen der Kommunikationsoptionen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

1. Wählen Sie das Menü **Extras >** **Projekteinstellungen...**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

2. Kommunikationsmodus wählen **Überwachung**

Alternative I - Verwendung des Docklight-Überwachungskabels:

3. Stellen Sie unter **Receive Channel 1** den COM-Port ein, an dem das Überwachungssignal von seriellem Gerät 1 empfangen wird. Stellen Sie unter **Receive Channel 2** den COM-Port für das zweite Gerät ein.

   HINWEIS: Im Docklight-Monitoring-Modus werden alle empfangenen Daten von einem COM-Port auf dem Sendekanal des gegenüberliegenden COM-Ports weitergesendet ("Data Forwarding"). Dies hat keine Auswirkung auf Docklight Monitoring Cable Setups, da das TX-Signal nicht angeschlossen ist. Es kann aber für spezielle Anwendungen nützlich sein, bei denen Sie den seriellen Datenverkehr über eine Standard-RS232-Verkabelung durch Docklight leiten müssen. Wenn Sie ein rein passives Überwachungsverhalten wünschen, bei dem keine TX-Daten erscheinen, können Sie die "Datenweiterleitung" über das Menü **Extras >** Expertenoptionen... deaktivieren.

Alternative II - Verwendung von Docklight Tap

Öffnen Sie bei **Empfangskanal 1** die Dropdown-Liste, scrollen Sie nach unten zum Abschnitt **-- USB Taps --** und wählen Sie den ersten Tap-Anschluss, z. B. **TAP0**. Bei **Empfangskanal 2** wird der zweite Tap-Anschluss (z. B. **TAP1**) automatisch ausgewählt.

Alternative III - Verwendung von Docklight Tap Pro / Docklight Tap 485:

3. Öffnen Sie bei **Empfangskanal 1** die Dropdown-Liste, scrollen Sie nach unten zum Abschnitt **-- USB-Taps --** und wählen Sie den ersten VTP-Tap-Port, z. B. **VTP0**. Bei **Empfangskanal 2** wird der zweite VTP-Tap-Port (z. B. **VTP1**) automatisch ausgewählt.

4. Stellen Sie die Baudrate und alle anderen Kommunikationsparameter für das verwendete Protokoll ein.

   HINWEIS: Vergewissern Sie sich, dass die serielle Schnittstelle Ihres PCs mit der Baudrate und den Kommunikationseinstellungen, die von Gerät 1 und Gerät 2 verwendet werden, richtig funktioniert. Wenn Gerät 1 und 2 ein Hochgeschwindigkeits-Datenübertragungsprotokoll verwenden, können die seriellen Schnittstellen des PCs und die Docklight-Software selbst zu langsam sein, um alle Daten korrekt zu empfangen.

5. Bestätigen Sie die Einstellungen und schließen Sie das Dialogfeld, indem Sie auf die Schaltfläche **OK** klicken.

C) Festlegen der verwendeten Empfangssequenzen
Definieren Sie Empfangssequenzen, die im Prüfprotokoll markiert werden sollen oder eine Aktion innerhalb von Docklight auslösen. Docklight prüft auf beiden Überwachungskanälen auf Receive Sequence, d.h. es spielt keine Rolle, ob die Sequenzen vom seriellen Gerät 1 oder vom seriellen Gerät 2 kommen.

HINWEIS: Da für diesen Test ein spezielles Überwachungskabel verwendet wird, bleibt die gesamte Kommunikation zwischen dem seriellen Gerät 1 und dem seriellen Gerät 2 unverfälscht und es werden keine zusätzlichen Verzögerungen durch Docklight selbst verursacht. Dies ist besonders wichtig, wenn Docklight zum Aufspüren von Zeitproblemen verwendet wird. Das bedeutet aber auch, dass es keine Möglichkeit gibt, die serielle Kommunikation zwischen den beiden Geräten zu beeinflussen. Solange der Kommunikationsmodus **Überwachung** gewählt ist, ist es nicht möglich, Sendesequenzen zu verwenden.

D) Speichern des Projekts
Vor der Durchführung des eigentlichen Tests empfiehlt es sich, die festgelegten

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kommunikationseinstellungen und Sequenzen zu speichern. Dies geschieht über das Menü **Datei > Projekt speichern**.

E) Durchführung des Tests

Starten Sie Docklight, indem Sie **Ausführen > Kommunikation starten** wählen, dann die seriellen Geräte 1 und 2 aktivieren und einen Testlauf durchführen. Docklight zeigt die gesamte Kommunikation zwischen dem seriellen Gerät 1 und dem seriellen Gerät 2 an. Docklight verwendet verschiedene Farben und Schriftarten

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Schriftarten, um die Unterscheidung zwischen den von Gerät 1 und Gerät 2 übertragenen Daten zu erleichtern. Die Farben und Schriftarten können auf der Registerkarte Anzeige des Dialogs **Extras > Optionen...** ausgewählt werden.

TIPP: Die Funktion Snapshot ermöglicht es Ihnen, eine seltene Sequenz oder einen Fehlerzustand in einem Kommunikationsprotokoll mit einer großen Datenmenge zu lokalisieren.

TIPP: In den Abschnitten Wie man die Verarbeitungsgeschwindigkeit erhöht... und Wie man die beste Timing-Genauigkeit erreicht, erfahren Sie, wie Sie Docklight für Anwendungen mit großen Datenmengen oder erhöhten Anforderungen an die Timing-Genauigkeit anpassen können.

5.4 Erfassen einer bestimmten Sequenz und Erstellen eines Schnappschusses der Kommunikation

Bei der Überwachung der seriellen Kommunikation zwischen zwei Geräten kann es vorkommen, dass Sie auf einen seltenen Fehler testen möchten, wobei nur die serielle Kommunikation vor und nach diesem Ereignis von Interesse ist. Sie könnten nach dieser Situation suchen, indem Sie den Test protokollieren und die Protokolldateien nach der charakteristischen Fehlersequenz durchsuchen. Dies könnte bedeuten, dass Sie mehrere MB an Daten speichern und analysieren müssen, obwohl Sie eigentlich nur nach ein paar Bytes suchen, falls diese überhaupt auftauchen. Als Alternative können Sie die Snapshot-Funktion verwenden, wie unten beschrieben.

Voraussetzungen

- Docklight ist bereit, einen Test durchzuführen, wie er in den vorherigen Anwendungsfällen beschrieben wurde, z. B. die Überwachung der seriellen Kommunikation zwischen zwei Geräten.

Einen Schnappschuss machen

A) Definieren eines Auslösers für den Schnappschuss

1. Definieren Sie die Sequenz, die in Ihrer Fehlersituation auftritt, als Empfangssequenz.
2. Aktivieren Sie die Registerkarte "**Auslöser"** im Bereich "Aktion" des Dialogs "Sequenz empfangen": Die Option "Auslöser" muss aktiviert sein, wenn es sich um die Sequenz handelt, die Sie aufspüren möchten.

HINWEIS: Vergessen Sie nicht, die Trigger-Option für alle anderen Empfangssequenzen zu deaktivieren, die in Ihrem Test ignoriert werden sollen, damit sie den Snapshot nicht auslösen.

B) Erstellen eines Schnappschusses

Klicken Sie auf die Schaltfläche **Snapshot** in der Symbolleiste. Docklight startet die Kommunikation, zeigt aber nichts im Kommunikationsfenster an. Wenn die Triggersequenz erkannt wird, zeigt Docklight die Kommunikationsdaten vor und nach dem Triggerereignis an. Weitere Daten werden verarbeitet, bis sich die Triggersequenz etwa in der Mitte des Kommunikationsfensters befindet. Dann stoppt Docklight die Kommunikation und positioniert den Cursor auf die Triggersequenz.

5.5 Protokollieren und Analysieren eines Tests

Voraussetzungen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

- Docklight ist bereit, einen Test durchzuführen, wie er in den vorherigen Anwendungsfällen beschrieben wurde, z.B. das Testen eines seriellen Geräts oder einer Protokollimplementierung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Protokollierung des Tests

Klicken Sie in der Hauptsymbolleiste auf die Schaltfläche

Logging starten. Es öffnet sich ein Dialogfenster zur

Auswahl der Einstellungen für die Protokolldatei.

Für jede Darstellung (ASCII, HEX, ...) kann eine eigene Protokolldatei erstellt werden. Wählen Sie mindestens eine Darstellung. Die Protokolldateien haben die Dateierweiterung ".txt", ".htm" oder ".rtf", je nachdem, welches Dateiformat Sie gewählt haben. Docklight fügt dem Dateinamen auch die Darstellungsart hinzu, um die verschiedenen Protokolldateien zu unterscheiden. Wenn der Benutzer z.B. "Test1" als Dateinamen angibt, wird die ASCII-Textdatei "Test1_asc.txt" genannt, während eine RTF-HEX-Protokolldatei "Test1_hex.rtf" genannt wird.

Bestätigen Sie Ihre Einstellungen für die Protokolldatei und starten Sie die Protokollierung, indem Sie auf die Schaltfläche **OK** klicken.

Um die Protokollierung zu beenden und die Protokolldatei(en) zu schließen, klicken Sie in der Hauptsymbolleiste auf die Schaltfläche **Stop Logging**. Solange die Protokolldatei(en) nicht geschlossen wurde(n), ist es nicht möglich, ihren gesamten Inhalt einzusehen.

TIPP: Wenn Sie zusätzliche Anforderungen an Ihr Logdateiformat haben, z.B. das Starten einer neuen Datei jede Stunde, können Sie zu diesem Zweck ein Docklight-Skript und die Methode StartLogging verwenden. Siehe auch das Beispielskript **LogFileNamesTimestamp.zip** (Ordner **Extra\LogFileNamesTimestamp** in Ihrem Verzeichnis Script Samples).

5.6 Prüfung auf Sequenzen mit zufälligen Zeichen (Empfang von Sequenz-Wildcards)

Viele serielle Geräte unterstützen eine Reihe von Befehlen zur Übermittlung von Messdaten und anderen zugehörigen Informationen. Bei gängigen textbasierten Protokollen besteht die Antwort des seriellen Geräts aus einem festen Teil (z. B. "temperature=") und einem variablen Teil, der den tatsächlichen Wert darstellt (z. B. "65F"). Um alle diese Antworten im seriellen Datenstrom korrekt zu erkennen, können Sie Empfangssequenzen mit Platzhaltern definieren.

Nehmen wir zum Beispiel die folgende Situation an: Ein serielles Gerät misst die Temperatur und sendet periodisch den aktuellen Messwert. Docklight zeigt die folgende Ausgabe an: `07/30/2012 10:20:08.022 [RX] - temperature=82F<CR>`

```
30.07.2012 10:22:10.558 [RX] - Temperatur=85F<CR>
30.07.2012 10:24:12.087 [RX] - Temperatur=93F<CR>
30.07.2012 10:26:14.891 [RX] - Temperatur=102F<CR>
```

...

Für jeden möglichen Temperaturwert eine eigene Empfangssequenz zu definieren, wäre keine praktische Option. Stattdessen würden Sie eine Empfangssequenz mit Platzhaltern definieren. Zum Beispiel:
t | e | m | p | e | r | a | t | u | r | e | = | ? | # | # | F | r ("r" ist das abschließende <CR> Carriage Return Zeichen)

Diese ReceiveSequence würde bei jedem der oben aufgeführten Temperaturstrings ausgelöst. Sie erlaubt einen 1-3 stelligen Wert für die Temperatur (d.h. von "0" bis "999"). Das folgende Beispiel beschreibt Schritt für Schritt, wie die obige Sequenz zu definieren ist. Siehe auch die zusätzlichen Anmerkungen am Ende dieses

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Abschnitts für zusätzliche Informationen über '#'-Platzhalter.

HINWEIS: Siehe Berechnung und Validierung von Prüfsummen über den Empfang und die Validierung von Prüfsummendaten, z. B. CRCs. Es sind keine Platzhalter für Prüfsummenbereiche erforderlich. Verwenden Sie stattdessen einige Standardzeichenwerte, z. B. "00 00" in HEX-Darstellung.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Voraussetzungen

- Docklight ist bereit, einen Test durchzuführen, wie er in den vorherigen Anwendungsfällen beschrieben wurde, z.B. den Test eines seriellen Geräts oder einer Protokollimplementierung.
- Das serielle Gerät (in unserem Beispiel das Temperaturmessgerät) ist in Betrieb.

Verwendung von Empfangssequenzen mit Platzhaltern

A) Vorbereitung des Projekts
Erstellen Sie ein neues Docklight-Projekt und richten Sie alle Kommunikationsparameter ein.

B) Festlegen der verwendeten Empfangssequenzen

1. Erstellen Sie eine neue Empfangssequenz. Geben Sie einen **Namen** für die Sequenz ein.
2. Geben Sie den festen Teil Ihrer erwarteten Antwort in den Abschnitt "**Sequenz"** ein. Für unser Beispiel würden Sie die folgende Sequenz im ASCII-Modus eingeben:
   t | e | m | p | e | r | a | t | u | r | e | =
3. Öffnen Sie mit der **rechten Maustaste** das Popup-/Kontextmenü und wählen Sie **Platzhalter '?' (entspricht einem Zeichen)**, um den ersten Platzhalter an der Cursorposition einzufügen. Fügen Sie zwei '#'-Platzhalter über das Popup-Menü **Platzhalter '#' (entspricht null oder einem Zeichen)** hinzu. Die Sequenz sieht nun wie folgt aus:
   t | e | m | p | e | r | a | t | u | r | e | = | ? | # | #
4. Geben Sie das feste Ende unserer Temperaturkette ein, d.h. den Buchstaben "F" und die abschließende <CR>-Zeichen. Sie können das Standard-Kurzbefehlszeichen **Strg+Enter** verwenden, um den <CR> / ASCII-Code 13 einzugeben. Die Sequenz lautet nun:
   t | e | m | p | e | r | a | t | u | r | e | = | ? | # | # | F | r _
5. Geben Sie eine **Aktion** an, die ausgeführt werden soll, nachdem ein Temperaturmesswert erfasst wurde.
6. Klicken Sie auf **OK**, um die neue Sequenz zur Liste der Empfangssequenzen hinzuzufügen.

HINWEIS: Um die Platzhalter '?' und '#' von den regulären Fragezeichen oder Zahlenzeichen (Dezimalcode 63 / 35) zu unterscheiden, werden die Platzhalter im Sequenzeditor mit einer anderen Hintergrundfarbe dargestellt.

C) Durchführung des Tests
Starten Sie Docklight, indem Sie **Ausführen >** **Kommunikation starten** wählen.

Docklight erkennt nun jeden Temperaturwert und führt die angegebene Aktion aus.

HINWEIS: Die Ereignisprozedur DL_OnReceive() ermöglicht die weitere Auswertung und Verarbeitung der tatsächlich empfangenen Messdaten.

Zusätzliche Hinweise zu '#'-Platzhaltern

1. Platzhalter '#' am Ende einer Receive Sequence haben keine Wirkung.
Die Receive Sequence "HelloWorld###" verhält sich wie eine Receive Sequence "HelloWorld".

2. Eine "Übereinstimmung innerhalb einer Übereinstimmung" wird nicht zurückgegeben: Wenn eine Empfangssequenz "Hello#######World" definiert ist und die eingehenden Daten "Hello1Hello2World" sind, ist die erkannte Empfangssequenz "Hello1Hello2World", nicht "Hello2World".

Makros für den Empfang von Sequenzkommentaren

Makro-Schlüsselwörter können in der Textbox Edit Receive Sequence **> 3 - Action > Comment** verwendet werden, um Docklight-Kommentartexte mit dynamischen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Daten, z.B. den tatsächlich empfangenen Daten, zu erstellen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Makro	Wird ersetzt durch
%_S	Glockensignal. Erzeugen Sie einen "Piepton", abhängig von Ihrem *Windows* solides System.
%_L	Zeilenumbruch
%_T	Zeitstempel für die empfangenen Daten
%_C	Docklight Kanal-Nr. / Datenrichtung (1 oder 2) für die empfangenen Daten
%_X	Der Kanalname oder Kanalalias, der der Datenrichtung %_C entspricht. Z.B. "RX", "TX" oder "COM5".
%_I	Index der Empfangssequenzliste, siehe den Dialog: Empfangsreihenfolge bearbeiten
%_N	Name der Empfangssequenz
%_A	Die tatsächlichen Daten, die diese Empfangssequenz ausgelöst haben. ASCII-Darstellung verwenden
%_H	Dasselbe wie %_A, aber in HEX-Darstellung
%_D	Wie %_A, aber in Dezimaldarstellung
%_B	Wie %_A, aber in Binärdarstellung
%_A(1,4)	Erweiterte Syntax: Fügen Sie nur die ersten 4 Zeichen dieser Empfangssequenz ein (beginnend mit Zeichen Nr. 1, Sequenzlänge = 4).
%_H(3,-1)	Erweiterte Syntax: Fügen Sie alles ab dem dritten Zeichen bis zum Ende der Sequenz ein (Länge = -1). Verwenden Sie die HEX-Darstellung.

Beispiel:
Für eine Empfangssequenz wie oben beschrieben (t | e | m | p | e | r | a | t | u | r | e | = | ? | # | # | F | r_), könnten Sie den folgenden Kommentartext definieren:

Neue Temperatur = %_L %_A(13, -3) °F

Die Docklight-Ausgabe könnte dann wie folgt aussehen:

```
30.10.2012 10:20:08.022 [RX] - Temperatur=82F<CR>
 Neue Temp =
 82 °F
30.10.2012 10:22:10.558 [RX] - Temperatur=85F<CR>
 Neue Temp =
 85 °F
30.10.2012 10:24:12.087 [RX] - Temperatur=93F<CR>
 Neue Temp =
 93 °F
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

5.7 Speichern und Laden Ihrer Projektdaten, Skripte und Optionen

Sie können das Verhalten von Docklight Scripting über drei verschiedene Arten von Benutzerkonfigurationsdaten festlegen:

- Projektdaten
- Skripte
- Programm-Optionen

Speichern und Laden von Projektdaten

Die Projektdaten umfassen:
- Senden
- Sequenzen
- Empfangen Sequenzen
  Zusätzliche Projekteinstellungen: Kommunikationsmodus, verwendete COM-Ports, COM-Port-Einstellungen (Baudrate, Parität, ...)
- Inhalt der Dokumentation

Das Projekt wird mit dem Menü **Datei > Projekt speichern** oder **Datei > Projekt speichern unter...** in einer Docklight-Projektdatei (.ptp-Datei) gespeichert.

Es wird allgemein empfohlen, Ihr Projekt zu speichern, bevor Sie einen Testlauf

starten. HINWEIS: Beim Speichern Ihres Projekts werden nur die Sequenzen,

Einstellungen und
Dokumentation Bereich Daten. Wenn Sie ein Protokoll der Kommunikation während eines Tests speichern möchten
ausführen, siehe Abschnitt Protokollierung und Analyse eines Tests.

Das Laden eines Projekts erfolgt über das Menü **Datei > Projekt öffnen....**

Speichern und Laden von Skripten

Docklight-Skriptcode für automatisierte Testaufgaben wird in einer separaten Datei (.pts-Datei) gespeichert. Verwenden Sie das Menü **Skripting > Skript speichern** oder **Skripting > Skript speichern unter...**

Speichern und Laden von Programmoptionen

Die Docklight-Optionen (Textformatierung, Verhalten von Steuerzeichen usw.) können mit Hilfe des Dialogs Docklight-Optionen (Menü **Extras > Optionen...**) geändert werden.

TIPP: Wenn Sie Ihr Skript ausführen, möchten Sie vielleicht einen bestimmten Satz von Optionen verwenden, um sicherzustellen, dass Docklight die Kommunikation und die Protokollausgabe in einem genau definierten Format erzeugt. Verwenden Sie die Methoden SaveProgramOptions und LoadProgramOptions, um eine Optionsdatei zu erstellen und die Optionen beim Start Ihres Skripts zu laden.

Projekt- und Skriptpaare verwenden: _auto.pts

In den meisten Docklight-Scripting-Anwendungen benötigt ein Docklight-Skript (.pts-Datei) ein zugehöriges Projekt (.ptp-Datei). Sie können das folgende Namensschema

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

verwenden, um das automatische Laden und Starten des Skripts zu ermöglichen:

meinproject**.ptp**
meinproject_auto**.pts**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Wenn sich die beiden Dateien im selben Ordner befinden, führt Docklight Scripting die folgenden zusätzlichen Operationen durch:

- Wenn **myproject.ptp** geöffnet wird (entweder per Doppelklick im Windows Explorer oder über das Menü **Datei >** **Projekt öffnen...**), öffnet Docklight Scripting nebenbei auch **myproject_auto.pts**, falls es nicht bereits geöffnet ist.
- Wenn **myproject_auto.pts** geöffnet wird, öffnet Docklight Scripting auch **myproject.ptp** nebenan, falls nicht bereits geöffnet.
- Wenn **Kommunikation starten** ausgeführt wird, wird der Kommunikationsanschluss geöffnet und das Skript gestartet.

HINWEIS: Wenn **myproject.ptp** in Docklight (ohne Skripting) geöffnet wird, erscheint eine Warnung, dass es sich um ein Projekt mit Skriptunterstützung zu handeln scheint und seine Verwendung in Docklight (ohne Skripting) eingeschränkt ist.

HINWEIS: Die Skriptmethoden OpenProject und StartCommunication sind von dem **_auto.**pts-Verhalten nicht betroffen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Arbeiten mit Docklight (Fortgeschrittene)

6 Arbeiten mit Docklight (Fortgeschrittene)

6.1 Senden von Befehlen mit Parametern (Sendesequenz-Wildcards)

Wenn Sie ein serielles Gerät testen, wird das Gerät höchstwahrscheinlich eine Reihe von Befehlen unterstützen, die einen Parameter enthalten.

Beispiel: Eine Digitalkamera unterstützt einen Befehl zur Einstellung der Belichtungszeit. Um die Belichtungszeit auf 25 Millisekunden einzustellen, müssen Sie die folgende Sequenz senden:
e | x | p | | 0 | 2 | 5 | r ("r" ist ein abschließendes <CR> Carriage Return Zeichen)

Um zu vermeiden, dass Sie für jede Aufnahme eine neue Sendefolge definieren, können Sie stattdessen eine Sendefolge mit Platzhaltern verwenden:
e | x | p | | ? | ? | ? | r

Das folgende Schritt-für-Schritt-Beispiel beschreibt, wie Sie einen Belichtungszeitbefehl mit einem Parameter definieren und jedes Mal, wenn die Sequenz gesendet wird, einen anderen Belichtungswert verwenden.

Voraussetzungen

- Docklight ist bereit, einen Test durchzuführen, wie er im Abschnitt Testen eines seriellen Geräts oder einer Protokollimplementierung beschrieben ist.

Durchführung des Tests über Befehle mit Parametern

A) Vorbereitung des Projekts
Erstellen Sie ein neues Docklight-Projekt und richten Sie alle Kommunikationsparameter ein.

B) Definieren der verwendeten Befehle

1. Erstellen Sie eine neue Sendesequenz. Geben Sie einen **Namen** für die Sequenz ein.
2. Geben Sie den festen Teil Ihres Befehls in den Abschnitt "**Sequenz"** ein. Für unser Beispiel würden Sie die folgende Sequenz im ASCII-Modus eingeben:
   e | x | p | |
3. Öffnen Sie nun mit der **rechten Maustaste** das Kontextmenü und wählen Sie **Platzhalter '?' (entspricht einem Zeichen) F7**, um einen Platzhalter an der Cursorposition einzufügen. In unserem Beispiel müssten wir das so lange wiederholen, bis es drei '?'-Platzhalter für unsere dreistellige Belichtungszeit gibt. Die Sequenz sieht nun wie folgt aus:
   e | x | p | | ? | ? | ?
4. Fügen Sie nun das abschließende <CR>-Zeichen hinzu, indem Sie den Standard-Kurzbefehl für Steuerzeichen **Strg+Enter verwenden.** Die Beispielsequenz lautet nun
   e | x | p | | ? | ? | ? | r
5. Klicken Sie auf **OK**, um die neue Sequenz in die Liste der Sendesequenzen aufzunehmen.

Wiederholen Sie die Schritte 1 - 5, um weitere Befehle zu definieren, die für die Durchführung Ihres Tests erforderlich sind.

HINWEIS: Um einen '?'-Platzhalter von einem Fragezeichen-ASCII-Zeichen (Dezimalcode 63) zu unterscheiden, wird der Platzhalter im Sequenzeditor mit einer anderen Hintergrundfarbe dargestellt.

C) Senden eines Befehls an das serielle Gerät

1. Verwenden Sie die Schaltfläche ----> **Senden**, um den seriellen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kommunikationsanschluss zu öffnen und einen Befehl an das serielle Gerät zu senden.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

2. Die Kommunikation wird angehalten, und das Dialogfeld "Sendesequenzparameter" wird angezeigt, in das Sie den Parameterwert eingeben können. In unserem Beispiel ist dies eine Belichtungszeit, z. B. "025".
3. Bestätigen Sie mit der **Eingabetaste**. Die Sequenz wird nun an das serielle Gerät gesendet.

Es ist möglich, Befehle mit mehreren Parametern zu definieren und dabei mehrere Platzhalterbereiche innerhalb einer Sequenz zu verwenden. Der Dialog Sequenzparameter senden erscheint dann mehrmals, bevor eine Sequenz gesendet wird.

HINWEIS: Wenn Sie **Wildcard '?'** verwenden, müssen Sie beim Senden der Sequenz genau ein Zeichen für jedes '?' angeben. Für Parameter mit variabler Länge verwenden Sie **Wildcard**
'#' (entspricht null oder einem Zeichen) F8.

HINWEIS: Sie können eine Sendesequenz mit Platzhaltern nicht als automatische Antwort auf eine Empfangssequenz verwenden (siehe Aktion).

HINWEIS: Wenn Ihre Sendesequenz eine Prüfsumme erfordert, können Sie diese wie unter Berechnung und Validierung von Prüfsummen beschrieben definieren. Die Prüfsumme wird berechnet, nachdem der Platzhalter-/Parameterbereich mit den eigentlichen Daten gefüllt wurde, dann werden die resultierenden Sequenzdaten an die Sendewarteschlange übergeben.

6.2 Erhöhung der Verarbeitungsgeschwindigkeit und Vermeidung von "Input Buffer Overflow"-Meldungen

Bei der Überwachung der seriellen Kommunikation zwischen zwei Geräten kann Docklight die Menge der eingehenden Daten nicht kontrollieren. Da Docklight eine Reihe von Formatierungs- und Konvertierungsregeln auf die seriellen Daten anwendet, kann nur eine begrenzte Anzahl von Bytes pro Sekunde verarbeitet werden. Es gibt zahlreiche Faktoren, die die Verarbeitungsgeschwindigkeit bestimmen, z.B. die verwendeten PC- und COM-Geräte, die Display-Einstellungen und die definierten Empfangssequenz-Aktionen. Es ist daher nicht möglich, eine typische Datenrate anzugeben.

Die zeitaufwendigste Aufgabe für Docklight ist die Formatierung der Farben und Schriften, die standardmäßig angewendet wird (siehe Docklight-Anzeigeoptionen). Wenn Docklight mit der Formatierung der eingehenden Daten nicht Schritt halten kann, schaltet es automatisch in den einfacheren Klartextmodus um.

Wenn dies immer noch nicht schnell genug ist, um die eingehenden Daten zu verarbeiten, fügt Docklight die folgende Meldung in der Ausgabe des Kommunikationsfensters und in den Protokolldateien hinzu.

```
DOCKLIGHT meldet: EingangspufferÜberlauf auf COM1
```

TIPP: Suchen Sie nach dieser Nachricht mit der Funktion **Sequenz im Kommunikationsfenster suchen...** (Strg + F).

Wenn Sie das oben beschriebene Verhalten beobachten, bietet Ihnen Docklight mehrere Möglichkeiten, den Datendurchsatz zu erhöhen.

1. Vereinfachen Sie die Bildschirmausgabe:
   - Deaktivieren Sie alle nicht benötigten Anzeigemodi im Dialog **Optionen...**
   - Verwenden Sie von Anfang an den Klartextmodus (siehe das oben beschriebene automatische Umschaltverhalten).
   - Wenn Sie den ASCII-Modus verwenden, deaktivieren Sie die Option Beschreibung der Steuerzeichen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

2. Protokollieren Sie die Kommunikationsdaten in einer reinen Textdatei, anstatt das/die Kommunikationsfenster zu verwenden:
   - Verwenden Sie das "Klartext"-Protokolldateiformat
   - Erstellen Sie nur eine Protokolldatei für die Darstellung (ASCII / HEX / Dezimal / Binär), die Sie tatsächlich benötigen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

- Deaktivieren Sie die Kommunikationsfenster während der Aufzeichnung, indem Sie die Option High Speed Logging verwenden

3. Verwenden Sie den Kommunikationsfilter aus dem Dialog **Projekteinstellungen...** und deaktivieren Sie die ursprünglichen seriellen Daten für eine oder beide Kommunikationsrichtungen. Dies ist besonders nützlich, wenn Sie genau wissen, wonach Sie suchen, und eine oder mehrere Empfangssequenzen für diese Daten definieren können. Diese Receive Sequences können jedes Mal, wenn die Sequenz im Datenstrom auftaucht, einen Kommentar ausgeben, so dass Sie immer noch wissen, was passiert ist, auch wenn die ursprünglichen seriellen Daten von Docklight nicht angezeigt werden.

6.3 Wie man die beste Timing-Genauigkeit erreicht

Viele RS232-Überwachungsprogramme - so auch Docklight - können die seriellen Daten nur bedingt zeitgenau erfassen. Dies kann dazu führen, dass Daten aus zwei verschiedenen Kommunikationsrichtungen in zeitlich falscher Reihenfolge angezeigt werden oder mehrere Telegramme aus einer Kommunikationsrichtung als ein Datenpaket erscheinen.

Dieses Verhalten wird nicht durch eine schlechte Programmierung verursacht, sondern ist vielmehr charakteristisch für ein PC/Windows-System und die verschiedenen beteiligten Hardware- und Software-Schichten.

Nicht spezifizierte Verzögerungen und zeitliche Ungenauigkeiten können durch:

- Der Chipsatz des COM-Geräts, z. B. der interne FIFO-Datenpuffer (First-In-First-Out).
- Die USB-Bus-Übertragung (für USB-Seriell-Konverter).
- Der serielle Gerätetreiber für Windows.
- Die Planung von Aufgaben/Prozessen in einem Multitasking-Betriebssystem wie Windows.
- Die Genauigkeit des Anbieters von Datum und Uhrzeit.

Docklight verfügt über einen sehr genauen Datums- und Zeitgeber mit einer Granularität im Millisekundenbereich, muss aber dennoch die Einschränkungen der umgebenden Hardware- und Softwareumgebung akzeptieren.

Sie können Folgendes tun, um zusätzliche Verzögerungen und Ungenauigkeiten zu minimieren und eine typische Genauigkeit der Zeiterfassung von 5 Millisekunden oder besser zu erreichen:

1. Holen Sie sich unser Docklight Tap für niedrigste USB-bezogene Latenzzeiten. Oder verwenden Sie die integrierten RS232-Anschlüsse, falls diese auf Ihrem PC noch verfügbar sind.

2. Wählen Sie unter **Extras > Expertenoptionen...** den Prozessmodus Extern / Hohe Priorität. Dialog.

3. Halten Sie sich bei der Überwachung großer Datenmengen an die Empfehlungen aus dem vorigen Abschnitt Wie erhöhe ich die Verarbeitungsgeschwindigkeit..., um zu vermeiden, dass der Eingabepuffer überläuft und der Computer aufgrund der hohen CPU-Auslastung nicht mehr reagiert.

HINWEIS: Die oben empfohlenen **Experten-Optionen...** verändern die Gesamtbalance des Systems und müssen mit Vorsicht verwendet werden. Die besten Ergebnisse können nur erzielt werden, wenn Docklight **als Administrator ausgeführt wird**. Bitte stellen Sie sicher, dass Sie die Hinweise und Warnungen in der Dokumentation verstanden haben.

4. Als Alternative zu den oben genannten 1.-3: Verwenden Sie unser Zubehör Docklight Tap Pro oder Docklight Tap 485, die ihren eigenen Embedded Time Provider verwenden und PC-basierte Ungenauigkeiten vollständig ausschließen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

6.4 Berechnen und Validieren von Prüfsummen

Viele Kommunikationsprotokolle enthalten zusätzliche Prüfsummenfelder, um die Datenintegrität zu gewährleisten und Übertragungsfehler zu erkennen. Ein gängiger Algorithmus ist der CRC (Cyclic

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Redundancy Code), der in verschiedenen Varianten für unterschiedliche Protokolle verwendet wird. Im folgenden Beispiel wird Schritt für Schritt beschrieben, wie die fliegende Prüfsummenberechnung für eine Sendesequenz eingerichtet wird und wie die automatische Validierung eines Prüfsummenbereichs in einer Empfangssequenz aktiviert wird.

TIPP: Ein Beispiel für die Adressierung eines Modbus-Slave-Gerätes finden Sie im Tutorial Modbus RTU mit CRC-Prüfsumme.

TIPP: Siehe auch die Methode DL.CalcChecksum zur Berechnung von Prüfsummen unter Verwendung von Skriptcode als Alternative.

Voraussetzungen

Sie kennen die Prüfsummenspezifikation für die
- Protokollnachrichten: Welcher Bereich der Sequenzdaten wird
- durch eine Prüfsumme geschützt?
  Wo befindet sich die Prüfsumme? (Normalerweise am Ende der Sequenz.)
- Welcher Prüfsummenalgorithmus sollte verwendet werden? (Höchstwahrscheinlich einer der Standard-CRC-Typen oder eine einfache MOD256-Summe).

Verwendung von Sendesequenzen mit automatischer Prüfsummenberechnung

A) Definieren einer Sendesequenz, die eine Prüfsumme enthält

1. Erstellen Sie eine neue Sendesequenz. Geben Sie einen **Namen** für die Sequenz ein.

2. Geben Sie den Teil "Sequenz" Ihrer Nachricht im Abschnitt "**Sequenz" ein**. Hier verwenden wir zum Beispiel eine sehr einfache HEX-Nachricht als Sequenz:
   01 | 02 | 03 | 04 | **??**

   Verwenden Sie das Kontextmenü über die **rechte Maustaste** oder **F7**, um den Platzhalter **??** zu erstellen.

   HINWEIS: Weitere Informationen zu Platzhaltern und Parametern finden Sie im Abschnitt Sendesequenzparameter.

3. Fügen Sie nun einen weiteren Wert 00 als Platzhalter für die Prüfsumme hinzu. 01 | 02 | 03 | 04 | ?? | **00**

   HINWEIS: In einer Sendesequenz können Sie einen beliebigen Zeichencode von 00-FF als Platzhalter an den Stellen verwenden, an denen später die berechnete Prüfsumme eingefügt werden soll. Dies unterscheidet sich von der Funktionsweise in einer Empfangssequenz, wo Sie ?? Platzhalter. Siehe das folgende Beispiel für eine Empfangssequenz.

4. Gehen Sie zur Registerkarte **Zusätzliche Einstellungen** | **Prüfsumme** und legen Sie die Prüfsumme fest. Hier haben wir zum Beispiel **MOD256** aus der Dropdown-Liste gewählt.

   HINWEIS: Das Textfeld für die Prüfsumme lässt Kommentare zu. Alles, was hinter einem #-Zeichen steht, ist nur ein Kommentar. Sie können Ihre eigenen Kommentare hinzufügen, um zu beschreiben, worum es bei dieser Prüfsumme geht.

5. Klicken Sie auf **OK**, um die neue Sequenz in die Liste der Sendesequenzen aufzunehmen.

B) Durchführung des Tests

6. Verwenden Sie die Schaltfläche ----> **Senden**, um einen der vordefinierten Befehle zu senden. Geben Sie einen Parameterwert ein, z. B. 05.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Bevor die Daten gesendet werden, berechnet Docklight die aktuelle MOD256-Prüfsumme. Das Ergebnis steht an der angegebenen Prüfsummenposition. Bei MOD256 ist dies standardmäßig die letzte Zeichenposition, was bedeutet, dass der Platzhalter 00 mit dem Ergebnis der Prüfsumme überschrieben wird.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Wenn wir 05 als Parameter beim Senden der Sequenz verwenden, sehen die von Docklight gesendeten Daten wie folgt aus:

```
18.06.2015 11:07:23.251 [TX] - 01 02 03 04 05 0F
```

Der Platzhalter wurde durch die Summe der Nachrichtenbytes ersetzt:
1 + 2 + 3 + 4 + 5 = 15 oder Hex 0F.

Verwendung von Empfangssequenzen mit automatischer Prüfsummenüberprüfung

A) Definieren einer Empfangssequenz mit Prüfsumme

1. Erstellen Sie eine neue Empfangssequenz. Geben Sie einen **Namen** für die Sequenz ein.

2. Geben Sie die Sequenzdaten ein, einschließlich eines Platzhalterbereichs für ein zufälliges Nutzdatenbyte und einen Platzhalter für die Prüfsumme. Wir verwenden das gleiche Telegramm wie im obigen Beispiel der Sendesequenz:
   01 | 02 | 03 | 04 | ?? | **00**.

3. Gehen Sie auf die Registerkarte **Aktion | Kommentar** und geben Sie den folgenden Text ein: **Korrekte Prüfsumme**

4. Gehen Sie auf die Registerkarte **Prüfsumme** und wählen Sie **MOD256** in der linken Dropdown-Liste.

5. Behalten Sie die Option **Prüfsumme OK erkennen bei**. Sie bedeutet, dass die Empfangssequenz nur dann ausgelöst wird, wenn das MOD256-Prüfsummenbyte in den empfangenen Daten korrekt ist.

5. Klicken Sie auf **OK**, um die Änderungen zu bestätigen.

B) Durchführung des Tests

6. Starten Sie die Kommunikation und senden Sie einige Datentelegramme an Ihre Docklight-Anwendung / COM-Port.

Die Ausgabe im Kommunikationsfenster könnte wie folgt aussehen:

```
15.02.2016 17:43:28.072 [RX] - 01 02 03 04 05 0F Korrekt
Prüfsumme

15.02.2016 17:43:31.870 [RX] - 01 02 03 04 0F 19 Korrekt
Prüfsumme

15.02.2016 17:43:35.833 [RX] - 01 02 03 04 10 1A Korrekt
Prüfsumme
```

HINWEIS: In diesem Beispiel wurde gezeigt, wie man eine Empfangssequenz definiert, die nur bei Datentelegrammen mit korrekter Prüfsumme ausgelöst wird. Es ist auch möglich, das Gegenteil zu tun: einen Prüfsummenfehler zu erkennen. Wechseln Sie dazu auf die Registerkarte **Prüfsumme** und ändern Sie die Option **Prüfsumme OK erkennen** in **Prüfsumme Falsch**.

6.5 Steuerung und Überwachung von RS232-Handshake-Signalen

Die Docklight-Projekteinstellungen zur Unterstützung der Flusskontrolle bieten einen manuellen Modus, der es Ihnen ermöglicht, die RTS- und DTR-Signale manuell zu setzen oder zurückzusetzen, indem Sie auf die entsprechende Anzeige klicken. Der folgende Abschnitt beschreibt, wie Sie das **Funktionszeichen '!' (F11**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Taste), um die RTS- und DTR-Signale innerhalb einer Sendesequenz vorübergehend zu ändern oder Änderungen für die CTS-, DSR-, DCD- oder RI-Leitungen mithilfe einer Empfangssequenz zu erkennen.

Voraussetzungen

- Docklight ist bereit, einen Test auszuführen, wie er unter Testen eines seriellen Geräts oder einer Protokollimplementierung beschrieben ist.
- Die Flusssteuerungsunterstützung ist in den Projekteinstellungen auf "Manuell" eingestellt.
- Das Docklight-Projekt enthält bereits eine oder mehrere Sendesequenzen, aber es gibt eine zusätzliche Anforderung für den Wechsel von RTS/DTR-Signalen beim Senden.

Implementierung von RTS/DTR-Signaländerungen

Für unser Beispiel nehmen wir an, dass wir einen RS485-Konverter verwenden, der die RS485-Transceiver-Steuerung benötigt, aber das DTR-Signal anstelle von RTS zum Umschalten zwischen "Senden" und "Empfangen" verwendet. Wir nehmen ferner an, dass es bereits eine "Test"-Sendesequenz gibt, die im ASCII-Modus wie folgt aussieht:
T | e | s | t

A) Ändern der bestehenden Sendesequenz

1. Öffnen Sie den Dialog Sendesequenz bearbeiten.
2. Schalten Sie den **Bearbeitungsmodus** auf **Dezimal** um. Unser "Test"-Beispiel sieht im Dezimalmodus wie folgt aus:
   084 | 101 | 115 | 116
3. Fügen Sie am Anfang ein RTS/DTR-Funktionszeichen ein: Drücken Sie **F11**, oder öffnen Sie mit der **rechten Maustaste** das Kontextmenü und wählen Sie **Funktionszeichen '!' (RTS- und DTR-Signale)**. Die Beispielsequenz lautet nun:
   ! | 084 | 101 | 115 | 116
4. Fügen Sie den neuen RTS/DTR-Status als dezimalen Parameterwert hinzu (siehe unten). In unserem Beispiel muss das DTR-Signal auf High gesetzt werden. Wir wählen "002" als Parameterwert, so dass die Sequenz nun lautet:
   ! | 002 | 084 | 101 | 115 | 116
5. Fügen Sie am Ende der Sequenz ein RTS/DTR-Funktionszeichen ein und verwenden Sie "000" als Parameterwert, um das DTR-Signal auf Null zu setzen. Die Sequenzdaten lauten nun:
   ! | 002 | 084 | 101 | 115 | 116 | ! | 000
6. Klicken Sie auf **OK**, um die Änderungen zu bestätigen.

HINWEIS: Um ein '!' RTS/DTR-Funktionszeichen von einem ASCII-Ausrufezeichen (Dezimalcode 33) zu unterscheiden, wird das RTS/DTR-Funktionszeichen im Sequenzeditor mit einer anderen Hintergrundfarbe dargestellt.

HINWEIS: Das Zeichen nach einem RTS/DTR-Funktionszeichen wird zum Setzen der RTS/DTR-Signale verwendet und nicht an das serielle Gerät gesendet (siehe Parameterwerte unten).

B) Senden der Daten mit zusätzlicher DTR-Steuerung

1. Senden Sie die Testsequenz über die Schaltfläche ----> **Senden**.

Docklight setzt nun das DTR-Signal auf High, sendet die ASCII-Sequenz "Test" und setzt dann DTR zurück.

HINWEIS: Die RTS/DTR-Anzeigen zeigen jede Änderung des RTS- oder DTR-Status an. Im obigen Beispiel wird DTR jedoch sehr schnell gesetzt und zurückgesetzt, so

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

dass die DTR-Anzeige wahrscheinlich keine visuelle Rückmeldung geben wird.
Wenn Sie das DTR-Verhalten tatsächlich "sehen" wollen, versuchen Sie, eine kleine Verzögerung zwischen den Zeichen einzuführen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Funktionszeichen '!' (F11) - Einstellung von RTS und DTR

Zeichenwert (Dezimaler Modus)	RTS	DTR
000	Niedrig	Niedrig
001	Hoch	Niedrig
002	Niedrig	Hoch
003	Hoch	Hoch

Vorübergehende Paritätsänderungen / 9-Bit-Anwendungen

Einige Protokolle und Anwendungen erfordern ein 9. Datenbit, z. B. für die Geräteadressierung auf einem Bus. Die einzige Möglichkeit, solche Geräte mit einem Standard-UART mit maximal 8 Datenbits anzusprechen, besteht darin, serielle Einstellungen zu verwenden, die ein Paritätsbit enthalten, und dieses Paritätsbit innerhalb einer Sendesequenz vorübergehend zu ändern. Das Funktionszeichen '!' unterstützt zu diesem Zweck zusätzliche Parameterwerte:

Zeichenwert (Dezimaler Modus)	Parität
016	Keine Parität
032	Ungerade Parität
048	Gerade Parität
064	Markieren. Paritätsbit auf logisch '1' setzen
080	Leerzeichen. Paritätsbit auf logisch '0' setzen

Die neuen Paritätseinstellungen werden ab dem nächsten regulären Zeichen angewendet, sowohl auf der Sende- als auch auf der Empfangsseite. Die Parität wird wieder auf die ursprünglichen Kommunikationseinstellungen zurückgesetzt, nachdem die Sendesequenz vollständig übertragen wurde.

HINWEIS: Die nützlichsten Parameter für dieses Funktionszeichen sind die Einstellungen "Mark" und "Space", da sie es Ihnen ermöglichen, das Paritätsbit auf einen definierten Wert zu setzen, der effektiv als neuntes Datenbit dient.

HINWEIS: Es wird empfohlen, das Paritätsfehlerzeichen auf "(ignorieren)" zu setzen, damit Sie die eingehenden Daten in beiden Fällen, 9.

TIPP: Siehe auch das Beispielprojekt **SwitchParityDemo.ptp** (Ordner **Extras\ParitySwitch_9BitProtocols** in Ihrem Verzeichnis \Samples).

Funktionszeichen '!' (F11) - Erkennung von Handshake-Signaländerungen (CTS, DSR, DCD oder RI)

Docklight Scripting erkennt zwar Änderungen der Handshake-Signale CTS, DSR, DCD oder RI, aber im normalen Betrieb sind diese Änderungen im Docklight-Kommunikationsfenster nicht sichtbar (ähnlich wie ein Break State).

Mit dem Funktionszeichen '!' können Sie diese Änderungen sichtbar machen und/oder eine Aktion nach dem Erkennen solcher Änderungen definieren. Das Funktionszeichen '!' unterstützt zu diesem Zweck die folgenden Parameterwerte:

Zeichenwert (Dezimaler Modus)	Handshake-Signal

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

001	CTS = Hoch
002	DSR = Hoch
004	DCD = Hoch

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

008	RI (Ring Indicator) = Hoch

HINWEIS: Siehe auch DL.GetHandshakeSignals() für den erweiterten Satz von Signalzuständen, die in Tap Pro / Tap RS485-Anwendungen unterstützt werden.

Beispiel für Empfangssequenzdefinitionen im Dezimal-Editiermodus:

Empfangsreihenfolge (Dezimalmodus)	Beschreibung
! | 001	wird ausgelöst, wenn CTS=high, alle anderen Signale low
! | 006	wird ausgelöst, wenn CTS=low, DSR=high, DCD=high, RI=low
! | ???	wird bei jeder Änderung der Statuszeilen ausgelöst

Für das folgende Beispiel nehmen wir an, dass Docklight bereit ist, einen Test auszuführen, wie unter Testen eines seriellen Geräts oder einer Protokollimplementierung beschrieben, und dass die Flusssteuerungsunterstützung in den Projekteinstellungen auf "Manuell" eingestellt ist.

A) Erstellen Sie eine neue Empfangssequenz zur Erkennung von Handshake-Signaländerungen.

1. Öffnen Sie den Dialog Empfangssequenz bearbeiten.
2. Schalten Sie den **Bearbeitungsmodus** auf **Dezimal** um.
3. Fügen Sie am Anfang ein Funktionszeichen "Signale" ein: Drücken Sie **F11**, oder öffnen Sie mit der **rechten Maustaste** das Kontextmenü und wählen Sie **Funktionszeichen '!' (CTS/DSR/DCD/RI-Änderungen)**.
4. Fügen Sie den Handshake-Status als dezimalen Parameterwert hinzu (siehe oben). In unserem Beispiel wollen wir erkennen, wenn CTS high ist, während alle anderen Signale low sind. Das bedeutet, dass wir "001" als Parameterwert eingeben müssen, so dass die Sequenz nun lautet:
   ! | 001
5. Geben Sie einen **Kommentar** für diese Sequenz an, z. B. "[CTS = high, DSR/DCD/RI = low]".
6. Klicken Sie auf **OK**, um die neue Reihenfolge zu bestätigen.

B) Starten Sie den Test und bestätigen Sie, dass Docklight nun erkennt, wenn die CTS-Leitung von Low auf High wechselt.

HINWEIS: Dieses Beispiel funktioniert nur, wenn CTS die einzige Handshake-Leitung mit "hohem" Pegel ist. Für einen flexibleren Ansatz können Sie das Zeichen nach dem '!'-Funktionszeichen als Platzhalter definieren und die Ereignisprozedur DL_OnReceive() verwenden, um den Zustand der Handshake-Leitungen auszuwerten.

Funktionszeichen '^' (F12) - bitweise Vergleiche

Das Funktionszeichen '^' kann durch Drücken von **F12** im Dialogfeld "Edit Receive Sequence" hinzugefügt werden. Nach dem '^'-Zeichen geben zwei zusätzliche Zeichenwerte an, welche Bits geprüft werden sollen (*"mask"*) und welche Werte für diese Bits zu erwarten sind ("*value"*).

Empfangsreihenfolge (HEX-Modus)	Beschreibung
^ | *Maske* | *Wert*	Ist eine Übereinstimmung für das nächste empfangene Zeichen, wenn Folgendes zutrifft: ((*nextCharacterReceived* XOR *value*) AND *mask*) = 0 Mit anderen Worten - das Zeichen '^' wählt nur die in der *Maske* markierten Bits aus und vergleicht sie mit den entsprechenden Bits im *Wert*. Siehe unten für Beispiele.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

^ | 0F | 05	Ist eine Übereinstimmung, wenn für das nächste Zeichen folgendes zutrifft:

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	Bit 0 = 1 Bit 1 = 0 Bit 2 = 1 Bit 3 = 0 Bit 4-7 = (ist egal)
! | ^ | 04 | 04	Diese Empfangssequenz wird ausgelöst, wenn der neue Handshake-Signalstatus lautet DCD = Hoch. Alle anderen Handshake-Signale können einen beliebigen Zustand annehmen. HINWEIS: Diese Empfangssequenz wird bei jeder Änderung eines Handshake-Signals ausgelöst, falls DCD immer noch High bleibt.

TIPP: Diese Erweiterung wird auch im Docklight Scripting Beispielprojekt **Docklight_TapPro_Demo.ptp** demonstriert (siehe den Ordner **Extras\TapPro** in Ihrem \Samples Verzeichnis)

6.6 Erstellen und Erkennen von Verzögerungen zwischen Zeichen

Einige Anwendungen, insbesondere Mikrocontroller-Anwendungen ohne eigenen seriellen Datenpuffer, erfordern eine zusätzliche Verzögerung zwischen einzelnen Zeichen, um Pufferüberläufe zu vermeiden und dem Mikrocontroller die Ausführung von anderem Code zu ermöglichen.

In Docklight können Sie Verzögerungen zwischen den Zeichen realisieren, indem Sie ein oder mehrere **Funktionszeichen '&' (Taste F9)** in Ihre Sendesequenzdaten einfügen, gefolgt von einem Zeichen, das die gewünschte Verzögerungszeit von 0,01 Sekunden bis 2,55 Sekunden angibt.

Sie können auch das Verzögerungszeichen '&' innerhalb einer Empfangssequenz verwenden, um eine Mindestpausenzeit anzugeben, in der keine weiteren Zeichen empfangen werden sollen. Dies ist nützlich, um Pausen im Datenstrom zu erkennen, die den Beginn/Ende eines Telegramms anzeigen, insbesondere bei Protokollen, bei denen es kein eigenes Start- oder Endzeichen gibt.

Voraussetzungen

- Docklight ist bereit, einen Test auszuführen, wie er unter Testen eines seriellen Geräts oder einer Protokollimplementierung beschrieben ist.
- Das Docklight-Projekt enthält bereits eine oder mehrere Sendesequenzen, aber es wird eine zusätzliche Verzögerung an bestimmten Zeichenpositionen benötigt.

Senden von Daten mit Verzögerungen zwischen den Zeichen

Als Beispiel verwenden wir eine Mikrocontroller-Anwendung, die einen "get"-Befehl versteht. Im ASCII-Modus würde die Sendesequenz lauten:
g | e | t | r_ ("r" ist ein abschließendes <CR> Carriage Return Zeichen)

Die folgenden Schritte beschreiben, wie man eine zusätzliche Verzögerung von 20 Millisekunden zwischen den einzelnen Zeichen hinzufügt, um Pufferüberläufe auf der Mikrocontrollerseite zu vermeiden.

A) Ändern der bestehenden Sendesequenz

1. Öffnen Sie den Dialog Sendesequenz bearbeiten.
2. Schalten Sie den **Bearbeitungsmodus** auf **Dezimal** um. Unser "get"-Beispiel sieht im Dezimalmodus wie folgt aus: 103 | 101 | 116 | 013
3. Fügen Sie ein Verzögerungszeichen zwischen dem ersten und dem zweiten Zeichen ein: Drücken Sie **F9**, oder öffnen Sie mit der **rechten Maustaste** das

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kontextmenü und wählen Sie **Funktionszeichen '&' (Verzögerung** Die Beispielsequenz lautet nun:
103 | & | 101 | 116 | 013

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

4. Fügen Sie die Verzögerungszeit hinzu: In diesem Beispiel wird ein Dezimalwert von 002 (20 Millisekunden) nach dem Funktionszeichen "&" hinzugefügt. Die Sequenz ist nun:
   103 | & | 002 | 101 | 116 | 013
5. Fügen Sie zwischen allen anderen Zwischenzeichenpositionen eine Verzögerung ein: Das Verzögerungszeichen und die Verzögerungszeit können Sie mit **Strg+C** kopieren und mit **Strg+V** an den gewünschten Stellen einfügen. Unsere Beispielsequenz lautet schließlich:
   103 | & | 002 | 101 | & | 002 | 116 | & | 002 | 013
   Oder zurück in den ASCII-Modus:
   g | & U|| e | & U|| t | & U|| r
6. Klicken Sie auf **OK**, um die Änderungen zu bestätigen.

HINWEIS: Um ein '&'-Verzögerungszeichen von einem normalen kaufmännischen ASCII-Zeichen (Dezimalcode 38) zu unterscheiden, wird das Verzögerungsfunktionszeichen im Sequenzeditor mit einer anderen Hintergrundfarbe dargestellt.

HINWEIS: Das Zeichen nach einem Zeichen der Verzögerungsfunktion wird als Verzögerungszeit interpretiert und nicht an das serielle Gerät gesendet.

B) Senden des Befehls an die Mikrocontroller-Anwendung

1. Senden Sie die geänderten Sendesequenzen über die Schaltfläche ----> **Senden**.

Docklight sendet die gleichen Daten wie zuvor, lässt aber zusätzliche Zeitlücken, die durch die Verzögerungszeichen festgelegt sind. Das Kommunikationsdisplay zeigt die gleichen Kommunikationsdaten wie ohne die Verzögerungszeichen an.

HINWEIS: Die Genauigkeit von Docklight für die Verzögerungszeit ist begrenzt, da es keine Kontrolle über den internen TX-FIFO-Puffer des UART hat. Die angegebenen Verzögerungszeiten für das '&'-Verzögerungsfunktionszeichen sind Mindestwerte. Gemessene Verzögerungswerte sind wesentlich höher, da Docklight immer eine Mindestzeit wartet, um sicherzustellen, dass der TX-FIFO-Puffer des UART leer ist. Auch das Anzeigeformat und die zusätzlichen Leistungseinstellungen beeinflussen das Timing. Wenn Sie spezifischere Anforderungen an das Timing der Sendesequenz haben und die Docklight-Wartezeit sowie Ihre UART-FIFO-Einstellungen kontrollieren müssen, wenden Sie sich bitte an unseren E-Mail-Support.

TIPP: Wenn Sie die gleiche Verzögerung zwischen den einzelnen Zeichen der gesendeten Daten benötigen, sehen Sie sich das Beispielskript **SendByteTiming.pts an** (siehe Ordner **Extras\SendByteTiming** in Ihrem Skript-Samples-Verzeichnis). Dieses Skript zerlegt Ihre Sendesequenzen automatisch in einzelne Zeichen und sendet die Daten "Byte für Byte" mit einer vordefinierten Verzögerung zwischen den Zeichen.

Pausenerkennung mit einer Empfangssequenz

Docklight bietet bereits die Anzeigeoption Pausenerkennung..., um nach Kommunikationspausen zusätzliche Zeitstempel oder Zeilenumbrüche einzufügen.

Wenn Sie nicht nur eine visuelle Formatierung benötigen, sondern auch Aktionen nach einer Mindestpause definieren oder einfach sicherstellen möchten, dass der Erkennungsalgorithmus für die Empfangssequenz nach einer Pause neu startet, können Sie das Zeichen für die Verzögerungsfunktion zu Ihrer Definition der Empfangssequenz hinzufügen.

In den meisten Anwendungen ist der beste Platz für das Verzögerungszeichen am Anfang der Empfangssequenz, vor den eigentlichen Empfangsdatenzeichen. Sie können auch eine Empfangssequenz erstellen, die nur eine Verzögerungs-/Pausendefinition und keine eigentlichen seriellen Daten enthält. Dies kann sehr nützlich

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

sein, um zeitliche Beschränkungen zu implementieren, z. B. um die Telegrammerkennung nach einer Pause zurückzusetzen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

TIPP: Siehe das **LineParser.ptp / LineParser.pts** Projekt und die Skriptdatei (Ordner **Extras\LineParser** in Ihrem Verzeichnis Script Samples) für eine Beispielanwendung.

6.7 Setzen und Erkennen eines "Break"-Status

Einige serielle Anwendungsprotokolle (z.B. LIN) nutzen den sogenannten Break-Zustand für Synchronisationszwecke. Docklight Scripting unterstützt das Senden eines "Break" innerhalb einer Sendesequenz und das Erkennen eines "Break"-Zustandes mit Hilfe einer Empfangssequenzdefinition. "Break"-Signale werden durch Einfügen eines **Funktionszeichens '%' (F10-Taste)** in Ihre Sequenzdefinition eingefügt. Ein Docklight-"Break"-Signal hat eine Mindestlänge von 15 *. <Nominal-Bitlänge>.

Voraussetzungen

- Docklight ist bereit, einen Test durchzuführen, wie er im Abschnitt Testen eines seriellen Geräts oder einer Protokollimplementierung beschrieben ist.
- Das Docklight-Projekt enthält bereits eine oder mehrere Sendesequenzen, aber auch die Signalisierung oder Erkennung eines "Break"-Zustandes ist erforderlich.

Senden eines "Break"-Status

Wir gehen davon aus, dass es bereits eine "Test"-Sendesequenz gibt, die im ASCII-Modus wie folgt aussieht:
T | e | s | t

1. Öffnen Sie den Dialog Sendesequenz bearbeiten.
2. Fügen Sie am Anfang ein "Break"-Funktionszeichen ein: Drücken Sie **F10**, oder öffnen Sie mit der **rechten Maustaste** das Kontextmenü und wählen Sie **Funktionszeichen '%' (Break-Signal)** . Die Beispielsequenz lautet nun:
   % | T | e | s | t
3. Klicken Sie auf **OK**, um die Änderungen zu bestätigen.
4. Senden Sie die Testsequenz über die Schaltfläche ----> **Senden**.

Die Sendeleitung geht mindestens 15 Bit lang auf Leerzeichen (logisch 0), dann wird die ASCII-Sequenz "Test" übertragen. Das "Break"-Zeichen erscheint nicht in der Anzeige des Kommunikationsfensters.

Erkennung eines "Break"-Status

Empfangene "Break"-Signale werden im Kommunikationsfenster nicht angezeigt, da sie nicht Teil der eigentlichen Datensequenz sind. Dennoch ist es möglich, eine Empfangssequenz zu definieren, die ein "Break"-Funktionszeichen enthält.

1. Erstellen Sie eine neue Empfangssequenz. Geben Sie einen **Namen** für die Sequenz ein.
2. Hinzufügen eines **Funktionszeichens '%' (Unterbrechungssignal)** für die Sequenzdaten.
3. Geben Sie eine **Aktion für die** Empfangssequenz ein, zum Beispiel den Kommentar "BREAK detected".
4. Klicken Sie auf **OK**, um die Änderungen zu bestätigen.
5. Kommunikation starten.

Docklight fügt nun jedes Mal, wenn ein Unterbrechungssignal `erkannt wird`, `BREAK detected` zur Anzeige im Kommunikationsfenster hinzu.

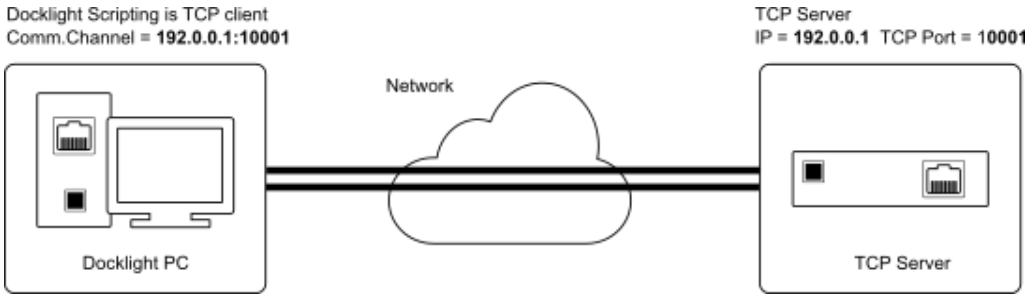
HINWEIS: Nach der Erkennung eines Unterbrechungssignals kann ein zusätzliches <NUL>-Zeichen (Dezimalcode 0) im empfangenen Datenstrom erscheinen. Dieses Verhalten kann nicht gesteuert werden durch

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Docklight, es hängt davon ab, wie der serielle UART des COM-Ports Ihres PCs den Break-Status interpretiert.

HINWEIS: Wenn Sie eine Empfangssequenz implementieren müssen, die auf ein Unterbrechungssignal gefolgt von zusätzlichen Daten prüft, beachten Sie, dass Docklight die genaue Position des Unterbrechungssignals im Datenstrom nicht bestimmen kann. Das Unterbrechungssignal wird manchmal früher im Datenstrom auftauchen, aber nie später als an der tatsächlichen Position. Um eine Empfangssequenz zu definieren, die bei Break + bestimmten Daten sicher auslöst, können Sie folgenden Workaround verwenden: Fügen Sie einige '#' (null oder ein Zeichen) Platzhalter zwischen dem Break-Zeichen und den zusätzlichen Daten ein. Die resultierende Empfangssequenz könnte wie folgt aussehen:
% | # | # | # | # | # | # | # | # | # | T | e | s | t

6.8 Testen eines TCP-Server-Geräts (Skripting)



Voraussetzungen

- Die IP-Adresse des Geräts ist bekannt, und das Gerät ist über das Netzwerk von dem Computer aus erreichbar, auf dem Docklight Scripting läuft - d.h. ein "Ping" zur IP-Adresse des Geräts funktioniert.
- Sie wissen, über welchen TCP-Port Sie eine Verbindung zu Ihrem Gerät herstellen können.
- Sie kennen die Protokollspezifikation für das Gerät, z. B. Modbus TCP, und den Satz von Befehlen, die das Gerät versteht.

Testen der TCP-Server-Protokollfunktionen

A) Einstellen der Kommunikationsoptionen

1. Wählen Sie das Menü **Extras > Projekteinstellungen...**
2. Wählen Sie den Kommunikationsmodus **Senden/Empfangen**
3. Geben Sie unter **Senden/Empfangen auf dem Kommunikationskanal** die IP-Adresse und die TCP-Portnummer für die Verbindung mit dem Gerät ein, z. B. 192.0.0.1:10001.
4. Bestätigen Sie die Einstellungen und schließen Sie das Dialogfeld, indem Sie auf die Schaltfläche **OK** klicken.

TIPP: Wenn Sie eine Verbindung zu einem Server herstellen wollen, der auf demselben Computer wie Docklight läuft, können Sie das Schlüsselwort **LOCALHOST** anstelle der eigentlichen IP-Adresse Ihres Computers verwenden (z.B. LOCALHOST:10001 für eine Verbindung zu einem Server an Port 10001 auf demselben Computer). Die Verwendung der Loopback-Adresse 127.0.0.1 hat den gleichen Effekt.

B) Festlegung der verwendeten Sende- und Empfangssequenzen:
Definieren Sie alle für Ihren Test erforderlichen Befehle und Antworten, wie unter Testen eines seriellen Geräts oder einer Protokollimplementierung beschrieben.

C) Durchführung des Tests

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Stellen Sie eine Verbindung her, indem Sie Run > Start Communication wählen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

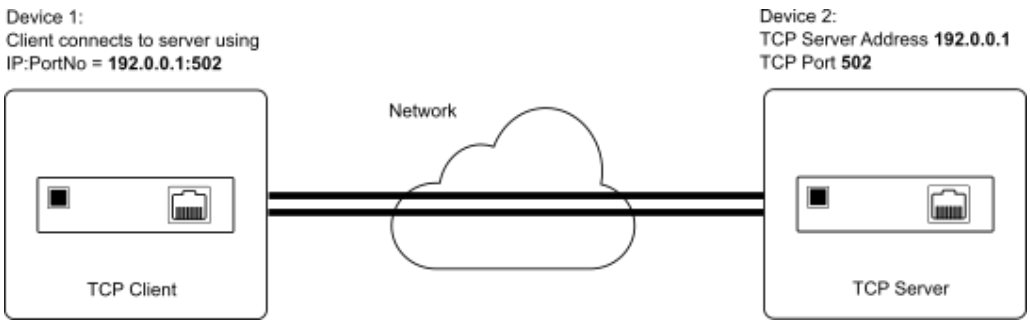
Docklight Scripting versucht nun, eine Verbindung mit dem TCP-Server-Gerät herzustellen. Nachdem die Verbindung hergestellt ist, können Sie eine Ihrer vordefinierten Sendesequenzen über die Schaltfläche ----> Senden senden. Solange der TCP-Server die Verbindungsanfrage nicht annimmt, werden im Kommunikationsfenster keine Sendedaten angezeigt.

HINWEIS: Wenn der Server die TCP-Sitzung beendet, bevor Sie in Docklight ■ Run > Stop Communication wählen, erhalten Sie die Fehlermeldung "TCP/IP connection closed by the remote computer", und die Kommunikation wird beendet.

HINWEIS: Wenn Sie beim Start der Kommunikation die Fehlermeldung "IP Address / TCP port in use" erhalten, prüfen Sie, ob ein anderer Server oder sogar eine andere Docklight Scripting-Instanz den Port blockiert. Versuchen Sie auch, Docklight Scripting zu schließen und neu zu starten - manchmal gibt die von Docklight Scripting verwendete TCP-Treiberschicht einen TCP-Port erst frei, wenn die ihn benutzende Anwendung geschlossen wird.

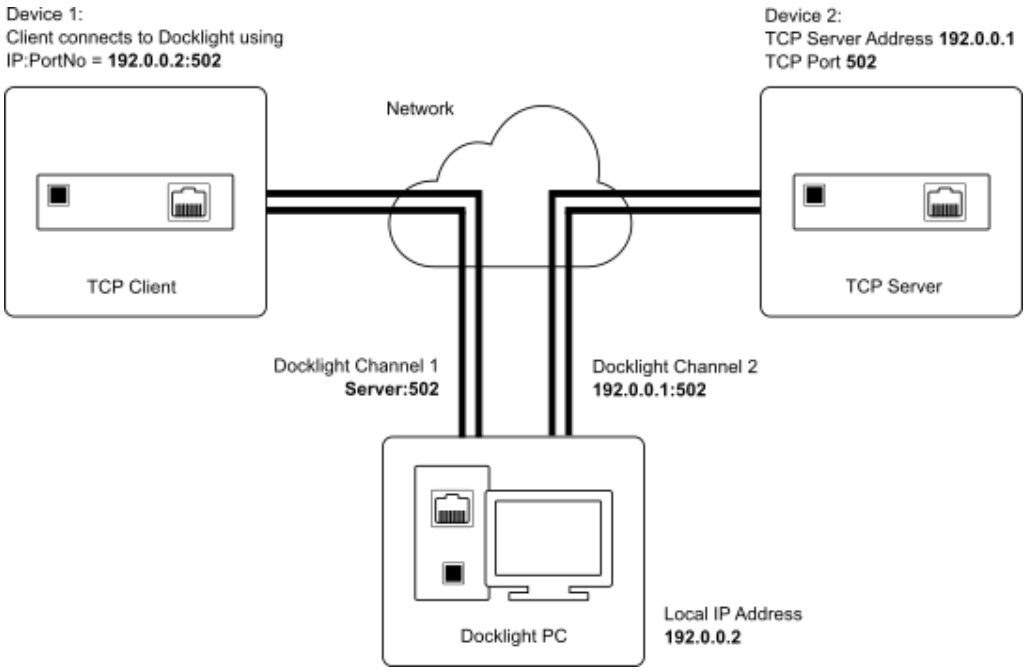
HINWEIS: Auch wenn kein anderer Server oder Client einen Port blockiert, kann es bis zu 4 Minuten dauern, bis ein Port tatsächlich freigegeben und wieder verfügbar ist. Dies ist eine Einschränkung in der TCP-Treiberschicht, die in Docklight Scripting verwendet wird, und leider kann Docklight Scripting dies nicht kontrollieren.

6.9 Überwachung einer Client/Server-TCP-Verbindung (Scripting)



Mit Docklight Scripting können Sie eine TCP-basierte Anwendung genauso einfach überwachen und debuggen wie bei der Verwendung von RS232-Ports und Kabeln. Anstatt ein Docklight-Überwachungskabel zwischen den beiden zu testenden Geräten zu verwenden, können Sie Docklight Scripting innerhalb des Netzwerks ausführen und einfach den Client (Gerät 1) anstelle des netzwerkfähigen Produkts (Gerät 2) mit Docklight Scripting verbinden lassen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de



Voraussetzungen

- Gerät 1, Gerät 2 und der PC mit Docklight sind in einem gemeinsamen Netzwerk (LAN) verbunden.
- Alle IP-Adressen und die TCP-Portnummer sind bekannt.
- Gerät 1 ist derzeit für die Verbindung mit Gerät 2 konfiguriert, und die Kommunikation zwischen den beiden Geräten funktioniert.

TCP-Verkehr routen und debuggen

A) Leiten Sie den Verkehr durch Docklight Scripting
Ändern Sie in Gerät 1 die Kommunikationsparameter: Gerät 1 muss sich mit dem Docklight-PC verbinden (in unserem Beispiel IP 192.0.0.2).

B) Docklight Scripting als Brücke für die Kommunikation zwischen Device 1 und Device 2 einrichten

1. Wählen Sie das Menü **Extras > Projekteinstellungen...**
2. Wählen Sie den Kommunikationsmodus **Überwachung (nur Empfangen)**
3. Geben Sie für den **Empfangskanal 1** das Schlüsselwort **SERVER** und den TCP-Port ein, der abgehört werden soll (z. B. SERVER:502).
4. Geben Sie für **Empfangskanal 2** die **IP-Adresse** und die **TCP-Portnummer** für die Verbindung mit Gerät 2 ein (z. B. 192.0.0.1:502).
5. Bestätigen Sie die Einstellungen und schließen Sie das Dialogfeld, indem Sie auf die Schaltfläche **OK** klicken.

C) Durchführung des Tests

Starten Sie Docklight Scripting mit **Run > Start Communication**. Lassen Sie Gerät 1 sich verbinden und führen Sie einen Testlauf durch. Docklight Scripting fungiert als Brücke zwischen den Geräten und zeigt Ihnen alle TCP-Daten, die zwischen den Geräten übertragen werden, im Kommunikationsfenster an.

HINWEIS: Docklight Scripting erlaubt keine Mehrfachverbindungen auf einem SERVER-Port. Es kann jeweils nur eine Verbindung verwendet werden. Dies entspricht dem Standardbetrieb vieler Serial Device Server.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

TIPP: Ein Beispiel, das auf jedem Computer mit einem Webbrowser und Internetzugang ausprobiert werden kann, ist das Projekt **TCP_Monitoring_HTTP.ptp**, das sich im Ordner **\Network** des Verzeichnisses \ScriptSamples befindet.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiele und Tutorials

7 Beispiele und Tutorials

In diesem Kapitel werden zwei Beispielprojekte beschrieben, die einige der grundlegenden Funktionen von Docklight demonstrieren. Die entsprechenden Docklight-Projektdateien (**.**ptp-Dateien) finden Sie im Verzeichnis **\Samples** Ordner im Docklight-Installationsverzeichnis (z.B. **C:\Programme\FuH\Docklight V2.3\Samples**).

TIPP: Der Ordner **\Samples** kann auch über den Docklight-Willkommensbildschirm erreicht werden (Menü **Hilfe > Willkommensbildschirm und Beispiele...**).

HINWEIS: Wenn Sie mit einem eingeschränkten Benutzerkonto angemeldet sind, haben Sie keine Berechtigung, Änderungen im Programmdateiverzeichnis vorzunehmen. In diesem Fall führt das Speichern einer Projektdatei oder anderer Daten im Ordner **\Samples** zu einem Fehler.

HINWEIS: Weitere Beispielprojekte und Anwendungshinweise finden Sie auch in unseren Online-Ressourcen unter https://docklight.de/examples/.

7.1 Testen eines Modems - Beispielprojekt: ModemDiagnostics.ptp

Mit dem Docklight-Projekt **ModemDiagnostics.ptp** können Sie eine Modemprüfung durchführen. Eine Reihe von Modemdiagnosebefehlen sind in der Liste Sendesequenzen definiert.

Dies ist ein einfaches Beispiel für den Test eines seriellen Geräts oder einer Protokollimplementierung. Das Beispielprojekt verwendet die unten aufgeführten Kommunikationseinstellungen. Dies sollte für die meisten Standardmodems funktionieren.

Kommunikationsmodus	Senden/Empfangen
COM-Port-Einstellungen	9600 Baud, keine Parität, 8 Datenbits, 1 Stoppbit

Erste Schritte

- Verwenden Sie den **Windows-Geräte-Manager**, um herauszufinden, welcher **COM-Port** ein Modemgerät ist. Dieses Demoprojekt kann mit jedem AT-kompatiblen Modem verwendet werden, das auf Ihrem PC vorhanden ist, z.B. ein eingebautes Notebook-Modem oder ein GSM- oder Bluetooth-Modemtreiber, auf den über einen virtuellen COM-Port zugegriffen werden kann.

TIPP: Für einen einfachen Test ohne spezielle Hardware fügen Sie Ihr Mobiltelefon als **Bluetooth-Gerät** auf Ihrem Windows-PC hinzu. Suchen Sie dann Ihr Telefon in der Windows-Liste "**Geräte und Drucker"**. Klicken Sie mit der rechten Maustaste darauf, wählen Sie **Eigenschaften** und gehen Sie auf die Registerkarte **Hardware**. In der Liste **Gerätefunktionen** sollten die modembezogenen COM-Anschlüsse aufgeführt sein.

- Rufen Sie das Dialogfeld **Projekteinstellungen... auf** und stellen Sie sicher, dass Sie denselben COM-Port für **Senden/Empfangen auf dem Kommunikationskanal** ausgewählt haben.
- Drücken Sie die Schaltfläche **Kommunikation starten** in der Symbolleiste.
- Versuchen Sie, einen der vordefinierten Modembefehle zu senden, indem Sie die Taste **Senden** Taste

Sie sollten nun eine Antwort von Ihrem Modem erhalten, z. B. "OK", wenn Ihr Befehl akzeptiert wurde, eine Modellidentifikationsnummer usw. Die Antwort ist je nach Modemmodell unterschiedlich.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Nach dem Senden mehrerer Sequenzen könnte das Docklight-Kommunikationsfenster wie folgt aussehen:

```
07.02.2013 18:17:54.083 [TX] - ATQ0V1E0<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
07.02.2013 18:17:54.107 [RX] - ATQ0V1E0<CR><LF>
<CR><LF>
OK<CR><LF>

07.02.2013 18:18:00.511 [TX] - ATI2<CR><LF>

07.02.2013 18:18:00.747 [RX] - <CR><LF>
V 11.10<CR><LF>
13-05-11<CR><LF>
RM-721<CR><LF>
(c) Nokia            <CR><LF>
<CR><LF>
OK<CR><LF>

07.02.2013 18:18:01.393 [TX] - ATI3<CR><LF>

07.02.2013 18:18:01.421 [RX] - <CR><LF>
Nokia C2-01<CR><LF>
<CR><LF>
OK<CR><LF>
...
```

Weitere Informationen

Die Liste der Sendesequenzen enthält die folgenden Standard-AT-Modembefehle:

Sequenz senden	Beschreibung / Modemantwort
ATQ0V1E0	Initialisiert die Abfrage.
AT+GMM	Modellkennung (die ITU V.250-Empfehlung wird nicht von allen Modems unterstützt).
AT+FCLASS=?	Vom Modem unterstützte Faxklassen, falls vorhanden.
AT#CLS=?	Zeigt an, ob das Modem den Rockwell-Sprachbefehlssatz unterstützt.
ATI<n>	Zeigt die Herstellerinformationen für <n> = 1 bis 7 an. Hier finden Sie Informationen wie die Anschlussgeschwindigkeit, das Ergebnis eines Prüfsummentests und die Modellinformationen. Prüfen Sie die die Dokumentation des Herstellers für die zu erwartenden Ergebnisse.

Der Ordner \Samples enthält auch eine Protokolldatei **ModemDiagnostics_Logfile_asc.txt**. Sie zeigt einen Testlauf, bei dem die oben genannten Sendesequenzen an ein echtes Modem gesendet wurden.

7.2 Reagieren auf eine Empfangssequenz - Beispielprojekt: PingPong.ptp

Das Docklight-Projekt **PingPong.ptp** ist ein einfaches Beispiel dafür, wie man Empfangssequenzen definiert und verwendet.

Erste Schritte

- Rufen Sie das Dialogfeld **Projekteinstellungen... auf** und wählen Sie einen COM-Anschluss.
- Legen Sie eine einfache Rückkopplungsschleife auf diesen COM-Anschluss: Verbinden Sie Pin 2 (RX) mit Pin 3 (TX). Siehe RS232 SUB D9 Pinout.
- Drücken Sie nun die Taste ----> **Senden** für eine der beiden Sendesequenzen. Die

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kommunikation wird gestartet und die Sendesequenz wird übertragen. Sie wird natürlich sofort auf der RX-Leitung des COM-Ports empfangen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Docklight erkennt, dass es sich bei den eingehenden Daten um eine der definierten Empfangssequenzen handelt. Es führt dann die für dieses Ereignis vordefinierte Aktion aus, nämlich das Senden einer weiteren Sequenz. Als Ergebnis sendet Docklight abwechselnd die Sendesequenzen "Ping" und "Pong" aus.

- Benutzen Sie die Schaltfläche ■ **Stop communication**, um die Demo zu beenden. Die Docklight-Kommunikationsanzeige sollte in etwa so aussehen:

```
3/8/2009 16:25:44.201 [TX] ------o Ping
3/8/2009 16:25:44.216 [RX] ------o Ping "Ping" erhalten
3/8/2009 16:25:44.218 [TX] - o---- Pong
3/8/2009 16:25:44.233 [RX] - o---- Pong "Pong" erhalten
3/8/2009 16:25:44.236 [TX] ------o Ping
3/8/2009 16:25:44.251 [RX] ------o Ping "Ping" erhalten
3/8/2009 16:25:44.254 [TX] - o---- Pong
3/8/2009 16:25:44.268 [RX] - o---- Pong "Pong" erhalten
3/8/2009 16:25:44.271 [TX] ------o Ping
3/8/2009 16:25:44.286 [RX] ------o Ping "Ping" erhalten
3/8/2009 16:25:44.289 [TX] - o---- Pong
3/8/2009 16:25:44.303 [RX] - o---- Pong "Pong" erhalten
3/8/2009 16:25:44.307 [TX] ------o Ping
3/8/2009 16:25:44.322 [RX] ------o Ping "Ping" erhalten
3/8/2009 16:25:44.324 [TX] - o---- Pong
...
```

Siehe auch die entsprechenden Protokolldateien im Ordner \Samples (**PingPong_Logfile_asc.htm** und **PingPong_Logfile_hex.htm**).

Weitere Informationen

Dieses Demoprojekt kann auch in drei alternativen Konfigurationen ausgeführt werden:

1. Führen Sie zwei Docklight-Anwendungen auf demselben PC über verschiedene COM-Ports aus. Die beiden COM-Ports werden mit einem einfachen Nullmodemkabel verbunden.
2. Anstelle von zwei RS232-COM-Anschlüssen und einem Nullmodemkabel können Sie ein virtuelles Nullmodem verwenden.
3. Verwenden Sie zwei PCs und lassen Sie Docklight auf jedem PC laufen. Verbinden Sie die beiden PCs mit einem einfachen Nullmodemkabel.

TIPP: Für Docklight Scripting gibt es auch ein verwandtes Beispielprojekt, das eine UDP-Loopback-Verbindung verwendet und keine seriellen Datenports benötigt. Siehe das Projekt **PingPong_UDP_Loopback.ptp** im Ordner **\Network** im Verzeichnis \ScriptSamples.

7.3 Modbus RTU mit CRC-Prüfsumme - Beispielprojekt: ModbusRtuCrc.ptp

Die Docklight-Projektdatei **ModbusRtuCrc.ptp** demonstriert, wie der CRC-Wert, der zum Senden eines gültigen Modbus-RTU-Frames erforderlich ist, automatisch berechnet wird.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Die Projektdatei verwendet die unten aufgeführten Kommunikationseinstellungen, entsprechend der Modbus-Implementierungsklasse "Basic".

Kommunikationsmodus	Senden/Empfangen
Senden/Empfangen auf Kommunikationskanal	COM1
COM-Port-Einstellungen	19200 Baud, gerade Parität, 8 Datenbits, 1 Stoppbit

Erste Schritte

- Öffnen Sie die Projektdatei **ModbusRtuCrc.ptp** (Menü Projekt öffnen ...). Die Datei befindet sich im Ordner \Samples.
- Verbinden Sie den COM-Port des PCs mit Ihrem Modbus-Netzwerk. Öffnen Sie das Dialogfeld **Projekteinstellungen...** und vergewissern Sie sich, dass Sie den richtigen COM-Port für **Senden/Empfangen am Kommunikationskanal** ausgewählt haben.
- Klicken Sie auf die Schaltfläche ----> **Senden** in der Zeile **Read Input Register Slave=?..** der Sendesequenzliste
- Geben Sie im Dialog **Sendesequenzparameter** eine Slavenummer ein, z. B. "01" für die Adressierung von Slave Nr. 1.

Nach dem Senden von "Read Input Register"-Kommandos an die Slaves 1 - 4 könnte das Kommunikationsfenster wie folgt aussehen:

```
23.09.2019 07:04:56.170 [TX] - 01 04 00 03 00 01 C1 CA
23.09.2019 07:04:56.282 [RX] - 01 04 02 FF FF B8 80
Erkannter Modbus-Frame = 01 04 02 FF FF B8 80
SlaveID=01
FunktionsCode=04
Adr/Daten=02 FF FF
CRC=B8 80

 Eingangsregister Antwort: Slave=001 ValueHex=FFFF

23.09.2019 07:05:21.761 [TX] - 02 04 00 03 00 01 C1 F9
23.09.2019 07:05:21.873 [RX] - 02 04 02 7F 58 DC FA
Erkannter Modbus-Frame = 02 04 02 7F 58 DC FA
SlaveID=02
FunktionsCode=04
Adr/Daten=02 7F 58
CRC=DC FA

 Eingangsregister Antwort: Slave=002 ValueHex=7F58

23.09.2019 07:05:35.713 [TX] - 03 04 00 03 00 01 C0 28
23.09.2019 07:05:35.824 [RX] - 03 04 02 01 0A 41 67
Erkannter Modbus-Frame = 03 04 02 01 0A 41 67
SlaveID=03
FunktionsCode=04
Adr/Daten=02 01 0A
CRC=41 67

 Eingangsregister Antwort: Slave=003 ValueHex=010A

23.09.2019 07:05:51.677 [TX] - 04 04 00 03 00 01 C1 9F
23.09.2019 07:05:51.789 [RX] - 04 04 02 40 00 44 F0
Erkannter Modbus-Frame = 04 04 02 40 00 44 F0
SlaveID=04
FunktionsCode=04
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Addr/Data=02 40 00
CRC=44 F0

Eingangsregister Antwort: Slave=004 WertHex=4000

Der Kanal [RX] zeigt die Antworten der Modbus-Slaves an: Slave 1 antwortete mit dem Wert "-1",
Sklave 2 antwortete "32600",
Slave 3 antwortete "266" und
Sklave 4 antwortete "16384".

HINWEIS: Wenn Sie das Docklight Modbus-Beispiel an einem RS485-Bus verwenden und keine Geräteantwort sehen, überprüfen Sie, ob Ihre RS485-Hardware-Schnittstelle automatisch zwischen Sende- und Empfangsmodus umschaltet, oder ob Sie die Option RS485 Transceiver Control verwenden müssen.

Weitere Informationen

- Die CRC-Berechnung erfolgt nach den Spezifikationen für die serielle Modbus-Leitungsübertragung (RTU-Modus). Docklight's Prüfsummenfunktion unterstützt zu diesem Zweck ein "CRC-MODBUS"-Modell. Weitere allgemeine Informationen zur Implementierung von Prüfsummenberechnungen finden Sie unter Berechnen und
- Validieren von Prüfsummen.
  Wenn Sie keine Modbus-Slave-Geräte zur Verfügung haben, können Sie einen Software-Simulator verwenden. Siehe www.plcsimulator.org/ wie ursprünglich erwähnt auf www.modbus.org, "Modbus Technical Resources", "Modbus Serial RTU Simulator". Dieser Simulator wurde verwendet, um die oben gezeigten Beispieldaten zu erzeugen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiele und Tutorials (Skripting)

8 Beispiele und Tutorials (Skripting)

Dieses Kapitel beschreibt Beispielskripte, die einige der Möglichkeiten von Docklight Scripting demonstrieren. Die entsprechenden Docklight-Skriptdateien (.pts-Dateien) und andere zugehörige Dateien finden Sie im Ordner **\ScriptSamples** im Docklight-Scripting-Installationsverzeichnis (z.B. **C:\Programme\FuH\Docklight Scripting V2.3\ScriptSamples**).

TIPP: Der Ordner **\ScriptSamples** kann auch über den Docklight Scripting Willkommensbildschirm erreicht werden (Menü **Hilfe > Willkommensbildschirm und Beispiele...**).

HINWEIS: Wenn Sie mit einem Benutzerkonto mit eingeschränktem Systemzugriff arbeiten, haben Sie möglicherweise keine Berechtigung zum Speichern im Programmdateiverzeichnis. In diesem Fall führt das Speichern einer Projektdatei oder anderer Daten im Ordner **\ScriptSamples** zu einem Fehler.

HINWEIS: Weitere Beispielskripte, Projekte und Anwendungshinweise finden Sie in unseren Online-Ressourcen unter https://docklight.de/examples/.

8.1 Automatisierte Modemtests - Beispielskript: ModemScript.pts

Das Docklight-Skript **ModemScript.pts** und die zugehörige Projektdatei **ModemATCommands.ptp** zeigen, wie man ein Docklight-Skript für einen automatisierten Test oder eine Konfigurationsaufgabe mit Benutzerinteraktion verwendet.

Die Projektdatei verwendet die unten aufgeführten Kommunikationseinstellungen. Dies sollte für die meisten Standardmodems funktionieren.

Kommunikationsmodus	Senden/Empfangen
Senden/Empfangen auf Kommunikationskanal	COM3
COM-Port-Einstellungen	9600 Baud, keine Parität, 8 Datenbits, 1 Stoppbit

Erste Schritte

- Schließen Sie das Modem an einen freien COM-Port an, z.B. COM1, und schalten Sie es ein. Die Demo kann auch auf einem Notebook mit eingebautem Modem laufen. In vielen Fällen finden Sie das eingebaute Modem Ihres Notebooks auf COM3, so dass Sie versuchen können, das Beispielskript auszuführen, ohne die Projekteinstellungen zu ändern.
- Gehen Sie zum Dialog **Projekteinstellungen...** und vergewissern Sie sich, dass Sie denselben COM-Port für **Senden/Empfangen auf dem Kommunikationskanal** ausgewählt haben.
- Drücken Sie die Schaltfläche **Skript ausführen** in der Symbolleiste.
- Geben Sie den AT-Befehlsbereich ein, der getestet werden soll, oder übernehmen Sie einfach den Standardwert durch Drücken der **Eingabetaste**.

Das Skript stellt nun eine Verbindung mit dem Modem her und durchläuft den AT-Befehlssatz. Nach erfolgreichem Abschluss des Testlaufs könnte das Docklight-Kommunikationsfenster wie folgt aussehen:

```
 Warten auf Modem ...

3/8/2009 16:23:08.870 [TX] - ATQ0V1E0<CR><LF>

3/8/2009 16:23:08.873 [RX] - ATQ0V1E0<CR>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
<CR><LF>
OK<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
Überprüfen des AT-Befehlssatzes...

3/8/2009 16:23:08.888 [TX] - ATI0<CR><LF>

3/8/2009 16:23:08.891 [RX] - <CR><LF>
Agere SoftModem Version 2.1.46<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:09.091 [TX] - ATI1<CR><LF>

3/8/2009 16:23:09.101 [RX] - <CR><LF>
OK<CR><LF>

3/8/2009 16:23:09.293 [TX] - ATI2<CR><LF>

3/8/2009 16:23:09.294 [RX] - <CR><LF>
OK<CR><LF>

3/8/2009 16:23:09.496 [TX] - ATI3<CR><LF>

3/8/2009 16:23:09.498 [RX] - <CR><LF>
Agere SoftModem Version 2.1.46<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:09.700 [TX] - ATI4<CR><LF>

3/8/2009 16:23:09.702 [RX] - <CR><LF>
Gebaut am 07/22/2004 14:50:10<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:09.901 [TX] - ATI5<CR><LF>

3/8/2009 16:23:09.912 [RX] - <CR><LF>
2.1.46, AMR Intel MB, AC97 ID:SIL REV:0x27, 06<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:10.104 [TX] - ATI6<CR><LF>

3/8/2009 16:23:10.110 [RX] - <CR><LF>
OK<CR><LF>

3/8/2009 16:23:10.308 [TX] - ATI7<CR><LF>

3/8/2009 16:23:10.315 [RX] - <CR><LF>
AMR Intel MB<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:10.510 [TX] - ATI8<CR><LF>

3/8/2009 16:23:10.513 [RX] - <CR><LF>
AC97 ID:SIL REV:0x27<CR><LF>
<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
OK<CR><LF>

3/8/2009 16:23:10.713 [TX] - ATI9<CR><LF>

3/8/2009 16:23:10.723 [RX] - <CR><LF>
Deutschland<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:10.916 [TX] - ATI10<CR><LF>

3/8/2009 16:23:10.921 [RX] - <CR><LF>
OK<CR><LF>

3/8/2009 16:23:11.119 [TX] - ATI11<CR><LF>

3/8/2009 16:23:11.120 [RX] - Beschreibung
  Status<CR><LF>
--------------------------------------------------<CR><LF>
Letzte
        VerbindungUnbekannt<CR><LF> Anfängliche
        Sende-Trägerrate0<CR><LF> Anfängliche
Empfangs-Trägerrate0<CR><LF> EndgültigeSende-
Trägerrate9600<CR><LF> EndgültigeEmpfangs-
Trägerrate9600<CR><LF>
ProtokollverhandlungsergebnisKeine<CR><LF> Ergebnis
der Datenkomprimierung
        KEINE<CR><LF>Geschätztes
Signal/Rauschverhältnis        (dB) 00<CR><LF>
Leistungspegel des Empfangssignals (-dBm)
00<CR><LF> Leistungspegel des Sendesignals (-
dBm) 10<CR><LF>Umlaufverzögerung (msec)
1000<CR><LF>
Nah-Echo-Pegel                (-dBm) 00<CR><LF>
FarLevel                      (-dBm)00<CR><LF>
Transmit Frame
                              Count0<CR><LF>
Transmit Frame Error Count          0<CR><LF>
Anzahl der empfangenen Frames       0<CR><LF>
Anzahl der Empfangsrahmenfehler     0<CR><LF>
Umlernen über lokales Modem         0<CR><LF>
Neuverhandlung        durch        entferntes
                                    Modem0<CR
><LF>  Raten-Neuverhandlung  durch  lokales
Modem  0<CR><LF>  Raten-Neuverhandlung  durch
entferntes  Modem  0<CR><LF>  Ursache  für
Rufabbruch                          0<CR><LF>
Robbed-Bit-Signalisierung0<CR><LF>Digitaler
Verlust                        (dB)00<CR><LF>
Remote Server ID                    NA<CR><LF>
<CR><LF>
OK<CR><LF>

3/8/2009 16:23:11.441 [TX] - ATI12<CR><LF>

3/8/2009 16:23:11.443 [RX] - <CR><LF>
OK<CR><LF>

3/8/2009 16:23:11.643 [TX] - ATI13<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
3/8/2009 16:23:11.654 [RX] - <CR><LF>
FEHLER<CR><LF>

3/8/2009 16:23:11.846 [TX] - ATI14<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
3/8/2009 16:23:11.852 [RX] - <CR><LF>
FEHLER<CR><LF>

 Ergebnisse:
 13 gültige AT-Befehle gefunden.
 2 AT-Befehl(e) funktionierte(n) nicht.
```

8.2 Starten von der Befehlszeile - Beispielskript: LogStartupScript.pts

Das Docklight-Skript **LogStartupScript.pts**, die zugehörige Projektdatei **LogStartupSettings.ptp** und die Batch-Datei **LogStartup.bat** zeigen, wie man Docklight von der Kommandozeile aus startet, eine Protokolldatei nach vordefinierten Einstellungen erzeugt und die Kommunikation automatisch startet.

Die Projektdatei verwendet die unten aufgeführten Kommunikationseinstellungen.

Kommunikationsmodus	Überwachung (nur Empfang)
Empfangskanal 1	COM1
Empfangskanal 2	COM3
COM-Port-Einstellungen	9600 Baud, keine Parität, 8 Datenbits, 1 Stoppbit

Erste Schritte

- Starten Sie die Batch-Datei **LogStartup.bat** von einer Befehlszeile aus oder wechseln Sie in das Verzeichnis \ScriptSamples und führen Sie **LogStartup.bat** durch Doppelklick auf die Datei aus.

Docklight Scripting wird gestartet, eine ASCII-Protokolldatei **C:\DocklightScripting_Logfile_asc.txt** wird erstellt und die Kommunikation wird sofort aufgenommen.

Verwenden Sie **Umschalt+F6**, um die Ausführung des Skripts zu beenden und die Kommunikationsports und die Protokolldatei zu schließen.

HINWEIS: Dieses Beispiel erfordert eine Softwarelizenz für die Docklight-Standardversion, da es die Docklight-Logging-Funktion nutzt. Eine Docklight Scripting-Lizenz ist für die Ausführung des Beispiels nicht erforderlich.

Weitere Informationen

Die Batch-Datei **LogStartup.bat** enthält die folgende Zeile:

```
..\Docklight_Scripting.exe -r LogStartupScript.pts
```

Dies startet Docklight Scripting, öffnet die Skriptdatei **LogStartupScript.pts** und führt sie sofort aus (Option **-r**). Das Skript **LogStartupScript.pts** enthält die folgenden Befehle:

```
' LogStartupScript.pts
' Protokollierung und Kommunikation
starten DL.OpenProject
"LogStartupSettings"
' Erstellen (oder Anhängen) einer ASCII-Protokolldatei
DL.StartLogging "C:\DocklightScripting_Logfile", True, "A"
DL.StartCommunication
' Kommunikation und Protokollierung aufrechterhalten,
bis der Benutzer aufhört Do
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
    DL.Pause 1 ' (die Pause reduziert die CPU-Last im
Leerlauf) Schleife
```

Die in **LogStartupSettings.ptp** verwendeten Kommunikationseinstellungen sind nur ein Beispiel. Wenn Sie andere Einstellungen benötigen, müssen Sie die Projektdatei öffnen, die Projekteinstellungen ändern und die Änderungen speichern. Es wird empfohlen, alle zugehörigen Dateien (**LogStartupScript.pts**, **LogStartupSettings.ptp** und **LogStartup.bat**) an einen anderen Ort zu kopieren, bevor Sie irgendwelche Änderungen vornehmen. In diesem Fall müssen Sie den vollständigen aktuellen Pfad zur **Docklight_Scripting.exe** Anwendung innerhalb der **.bat** Datei angeben.

8.3 Manipulation eines RS232-Datenstroms - Beispielskript: ZeichenManipulation.pts

Das Docklight-Skript **CharacterManipulation.pts** demonstriert, wie ein RS232-Datenstrom mit Hilfe der Ereignisprozedur DL_OnReceive() manipuliert werden kann. Alle Daten, die auf der RX-Leitung empfangen werden, werden auf der TX-Leitung wieder gesendet, wobei jedoch einige der Zeichen ersetzt werden.

Erste Schritte

- Öffnen Sie die Projektdatei **CharacterManipulationPrj.ptp** (über das Menü **Open Project ...**) und die Skriptdatei **CharacterManipulation.pts** (über das Menü **Open Script ...**). Die Dateien befinden sich im Ordner \ScriptSamples.
- Rufen Sie das Dialogfeld **Projekteinstellungen... auf** und
- wählen Sie einen COM-Anschluss. Drücken Sie die
- SchaltflächeSkript ausführen in der Symbolleiste.
  Starten Sie eine zweite Instanz von Docklight und öffnen Sie die Projektdatei **CharacterManipulationTest.ptp**.
- Wählen Sie einen anderen COM-Port für diese zweite Docklight-Instanz (oder verwenden Sie einen anderen Computer).
- Verbinden Sie die beiden COM-Ports mit einem einfachen Nullmodemkabel. Oder verwenden Sie stattdessen ein virtuelles Nullmodem.
- Verwenden Sie die Schaltfläche ----> **Senden** auf der zweiten Instanz von Docklight, um den Testsatz zu senden.

Die Kommunikationsanzeige der zweiten Instanz von Docklight sollte in etwa so aussehen:

```
2/21/2009 11:56:57.343 [TX] - Dies ist ein Test für das
Beispielskript zur Zeichenmanipulation
2/21/2009 11:56:57.502 [RX] - Dies ist ein Desd für den Charakter
manibuladion samble scribd
```

Jedes "T" wird durch ein "D" ersetzt, und jedes "P" durch ein "B". (Besucher des Nürnberger Raums, wo unser Unternehmen ansässig ist, werden vielleicht feststellen, dass Dialektsprecher hier etwas Ähnliches tun...)

Weitere Informationen

- Das Beispiel verwendet die Ereignisprozedur **DL_OnReceive()**, um jedes Mal, wenn ein neues Zeichen empfangen wird, zusätzliche Operationen durchzuführen. Weitere Einzelheiten finden Sie unter Auswerten der empfangenen Sequenzdaten.
- Die Leistung eines Zeichen-für-Zeichen-Prozessors in Docklight Scripting ist recht begrenzt. Sie können ihn leicht überlasten, indem Sie einen konstanten Datenstrom senden. Docklight zeigt in diesem Fall einen Kommentar im

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kommunikationsfenster an, z.B.
DOCKLI GHT meldet sich: Eingangspuffer über den Anschluss an COM1

- Aus Leistungsgründen ist die Anzeige aller TX- und RX-Daten in **ZeichenManipulationPrj.ptp**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

- Wenn Sie daran denken, einen Manipulator für Ihr eigenes Protokoll zu schreiben, sollten Sie einen paketbasierten Ansatz in Betracht ziehen, bei dem eine Empfangssequenz ein ganzes Paket oder einen Befehl Ihres Protokolls erkennen kann. Dies ermöglicht höhere Datenraten als der hier vorgestellte, auf basierende Ansatz.

8.4 TCP/IP-Kommunikation - Beispielprojekte PingPong_TCP_Server/Client.ptp

Die Projektdateien **PingPong_TCP_Server.ptp** und **PingPong_TCP_Client.ptp** im Verzeichnis \ScriptSamples\Network-Ordner zeigen, wie man Docklight Scripting als TCP-Server oder TCP-Client einsetzt und Daten austauscht.

Die Beispiele zeigen, wie ein Server und ein Client auf demselben Computer betrieben werden können. Dabei wird der Netzwerkname **LOCALHOST verwendet, der** sich immer auf den Computer bezieht, auf dem Docklight läuft.

Erste Schritte

- Öffnen Sie das Projekt **PingPong_TCP_Server.ptp** in Docklight
- Scripting. Drücken Sie die Schaltfläche **Kommunikation starten**
- in der Symbolleiste.
  Wenn Sie eine Personal Firewall auf Ihrem PC verwenden, wird diese Sie wahrscheinlich darauf hinweisen, dass Docklight Scripting als Server agieren
- möchte. Bestätigen Sie dies und erlauben Sie es, falls erforderlich.
  Starten Sie eine zweite Instanz von Docklight Scripting und öffnen Sie die **PingPong_TCP_Client.ptp** Projekt
- In dieser "Client"-Instanz drücken Sie die Schaltfläche **Senden** für die "Ping"-Sequenz.
- Wenn Sie eine Personal Firewall auf Ihrem PC verwenden, erlauben Sie Docklight Scripting, sich mit dem Internet zu verbinden.

Das "Client"-Docklight verbindet sich nun mit dem "Server"-Docklight, und die Daten werden so ausgetauscht, als ob die beiden Docklight-Instanzen über ein serielles Nullmodemkabel verbunden wären.

Das Kommunikationsfenster auf der Client-Seite zeigt nun die folgenden Meldungen an:

```
3/9/2009 17:29:24.192 [TX] ------o Ping
3/9/2009 17:29:24.218 [RX] - o---- Pong "Pong" erhalten
3/9/2009 17:29:24.221 [TX] ------o Ping
3/9/2009 17:29:24.249 [RX] - o---- Pong "Pong" erhalten
3/9/2009 17:29:24.254 [TX] ------o Ping
3/9/2009 17:29:24.281 [RX] - o---- Pong "Pong" erhalten
3/9/2009 17:29:24.283 [TX] ------o Ping
3/9/2009 17:29:24.312 [RX] - o---- Pong "Pong" erhalten
...
```

Auf der Serverseite sehen Sie dann etwa so aus:

```
3/9/2009 17:29:24.203 [RX] ------o Ping "Ping" erhalten
3/9/2009 17:29:24.206 [TX] - o---- Pong
3/9/2009 17:29:24.235 [RX] ------o Ping "Ping" erhalten
3/9/2009 17:29:24.238 [TX] - o---- Pong
3/9/2009 17:29:24.266 [RX] ------o Ping "Ping" erhalten
3/9/2009 17:29:24.268 [TX] - o---- Pong
3/9/2009 17:29:24.298 [RX] ------o Ping "Ping" erhalten
3/9/2009 17:29:24.301 [TX] - o---- Pong
...
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Referenz

9 Referenz

9.1 Menü und Symbolleiste (Scripting)

HINWEIS: Für gängige Menü- und Symbolleistenfunktionen sind Hotkeys verfügbar.

Menü "Datei

Neues Projekt
Schließen Sie das aktuelle Docklight-Projekt und legen Sie ein neues an.

Offenes Projekt ...
Schließen Sie das aktuelle Docklight-Projekt und öffnen Sie ein anderes Projekt.

Sequenzliste importieren ...
Importieren Sie alle Sendesequenzen und Empfangssequenzen aus einem zweiten Docklight-Projekt.

Projekt speichern / Projekt speichern unter ...
Speichern Sie das aktuelle Docklight-Projekt.

Projekt drucken ...
Druckt die Projektdaten, d.h. die Liste der definierten Sende- und Empfangssequenzen. Die Sequenzen werden in der gleichen Darstellung (ASCII, HEX, Dezimal oder Binär) gedruckt, die auch im Docklight-Hauptfenster verwendet wird. Die Darstellung kann über das Dialogfenster Optionen gewählt werden.

Kommunikation drucken ...
Druckt den Inhalt des Kommunikationsfensters. Die Kommunikationsdaten werden in der gleichen Darstellung gedruckt, die gerade im Kommunikationsfenster sichtbar ist.

Ausfahrt
Verlassen Sie Docklight.

Menü Bearbeiten

Sendesequenzliste bearbeiten ...
Bearbeiten Sie die Liste der Sendesequenzen, d.h. fügen Sie neue Sequenzen hinzu oder löschen Sie bestehende.

Empfangssequenzliste bearbeiten ...
Bearbeiten Sie die Liste der Empfangssequenzen, d.h. fügen Sie neue Sequenzen hinzu oder löschen Sie bestehende.

Vertauschen von Sende- und Empfangsfolgelisten
Konvertieren Sie alle Sendesequenzen in Empfangssequenzen und umgekehrt.

Sequenz im Kommunikationsfenster suchen...
Suchen Sie eine bestimmte Sequenz innerhalb der im Kommunikationsfenster angezeigten Daten. Siehe die Funktion Sequenz suchen.

Klares Kommunikationsfenster
Löschen Sie den Inhalt des Kommunikationsfensters. Dies gilt für alle vier Darstellungen (ASCII, HEX, Dezimal, Binär) des Kommunikationsfensters.

Menü ausführen

Start der Kommunikation
Öffnen Sie die Kommunikationsanschlüsse und aktivieren Sie die serielle Datenübertragung.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kommunikation beenden
Stoppen Sie die serielle Datenübertragung und schließen Sie die Kommunikationsanschlüsse.

Menü "Werkzeuge

Kommunikationsprotokollierung starten ...
Erstellen Sie eine oder mehrere neue Protokolldateien und beginnen Sie mit der Aufzeichnung der eingehenden/ausgehenden seriellen Daten. Siehe Protokollieren und Analysieren eines Tests.

Kommunikationsprotokollierung beenden
Beendet die Protokollierung und schließt die aktuell geöffnete(n) Protokolldatei(en).

Schnappschuss-Modus starten
Warten Sie auf eine Auslösesequenz und machen Sie einen Schnappschuss. Siehe Abfangen einer bestimmten Sequenz...

Schnappschuss-Modus beenden
Abbruch eines Schnappschusses und erneute Aktivierung der Anzeige des Kommunikationsfensters.

Tastatur Konsole Ein
Aktivieren Sie die Tastaturkonsole, um Tastatureingaben direkt zu senden.

Tastatur Konsole Aus
Deaktivieren Sie die Tastaturkonsole.

Minimieren/Wiederherstellen Dokumentation und Skript
Verkleinern Sie den Dokumentations- und Skriptbereich oder bringen Sie ihn auf die normale Größe zurück.

Minimieren/Wiederherstellen von Sequenzlisten
Verkleinern Sie die Listen der Sende-/Empfangsreihenfolge, oder bringen Sie sie auf die normale Größe zurück.

Projekt-Einstellungen...
Wählen Sie die aktuellen Projekteinstellungen (COM-Ports, Baudrate, ...).

Optionen...
Wählen Sie allgemeine Programmoptionen (z.B. Anzeigemodus, Datum / Zeitstempel).

Experten-Optionen...
Wählen Sie die Optionen des Expertenprogramms für fortgeschrittene Benutzer und spezielle Anwendungen (z. B. hohe Überwachungsgenauigkeit).

Menü "Skripting

Skript ausführen / fortsetzen
Führen Sie den Code im Skript-Editor aus.

Skript anhalten
Ein laufendes Skript anhalten.

Pause Skript
Unterbricht ein laufendes Skript.

Neue Schrift
Schließen Sie das aktuelle Docklight-Skript und erstellen Sie ein neues.

Skript öffnen ...
Schließen Sie das aktuelle Docklight-Skript und öffnen Sie ein anderes Skript.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Skript speichern / Skript speichern unter ...
Speichern Sie das aktuelle Docklight-Skript.

Anpassen / Externer Editor...
Verwenden Sie einen externen Editor anstelle des eingebauten Editors von Docklight.

9.2 Dialog: Sendesequenz bearbeiten

In diesem Dialogfeld können Sie neue Sendesequenzen definieren und bestehende Sequenzen bearbeiten (siehe auch Bearbeiten und Verwalten von Sequenzen).

Index
Der Index der unten angezeigten Sequenz. Die erste Sequenz hat den Index 0 (Null).

1 - Name
Eindeutiger Name für diese Sequenz (z. B. "Modemlautstärke einstellen"). Dieser Name dient der Referenzierung der Sequenz. Es handelt sich nicht um die Daten, die über die serielle Schnittstelle gesendet werden. Siehe "2 - Sequenz" unten.

2 - Sequenz
Die Zeichenfolge, die über die serielle Schnittstelle übertragen wird.

TIPP: Für die Übertragung größerer Datenblöcke, die die maximale Sequenzgröße überschreiten, verwenden Sie den Skriptbefehl DL.UploadFile.

TIPP: Spezielle Funktionszeichen stehen zur Verfügung, um Verzögerungen zwischen den Zeichen zu erzeugen, Handshake-Signale und Paritätsbits zu setzen oder einen Unterbrechungsstatus zu setzen.

3 - Zusätzliche Einstellungen
- **Wiederholen** - Aktivieren Sie die Option "Periodisch senden...", um eine Sequenz zu definieren, die periodisch gesendet wird. Es kann ein Zeitintervall zwischen 0,01 Sekunden und 9999 Sekunden festgelegt werden.

HINWEIS: Die für diesen Zweck verwendete Windows-Referenzzeit hat nur eine begrenzte Genauigkeit. Zeitintervalle < 0,03 Sekunden sind in der Regel nicht genau.

- **Prüfsumme** - Automatische Berechnung jeder Art von Prüfsumme, einschließlich jeder Art von CRC-Standard wie Modbus, CCITT, CRC32.

TIPP: Einen allgemeinen Überblick finden Sie unter Berechnen und Validieren von Prüfsummen und unter Prüfsummenspezifikation für das Textformat, das zur Definition einer Prüfsumme verwendet wird.

Wildcards
Platzhalter können verwendet werden, um Parameter in eine Sendesequenz einzufügen, die Sie jedes Mal, wenn die Sequenz gesendet wird, manuell einfügen möchten. Einzelheiten und Beispiele finden Sie im Abschnitt Befehle mit Parametern senden.

Tastenkombinationen für Steuerzeichen
Die Verwendung von Tastaturkürzeln ist eine große Hilfe bei der Bearbeitung einer Sequenz, die sowohl druckbare Zeichen (Buchstaben A-z, Ziffern 0-9, ...) als auch nicht druckbare Steuerzeichen (ASCII-Code 0 bis 31) enthält. Vordefinierte Tastenkombinationen sind:
Strg+Eingabe für Wagenrücklauf / <CR> / Dezimalcode 13
Strg+Shift+Enter für Zeilenvorschub / <LF> / Dezimalcode 10

Verwenden Sie Optionen... --> Steuerzeichenkurzbefehle, um andere Kurzbefehle zu definieren, die Sie für nützlich halten.

Dokumentation der Sequenz

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Fügen Sie hier eine Dokumentation zu dieser Sequenz hinzu. Diese Dokumentation wird auch im Hauptfenster angezeigt, wenn Sie die Sequenz in der Liste Sendesequenzen auswählen.

9.3 Dialog: Empfangssequenz bearbeiten

In diesem Dialogfeld können Sie neue Empfangssequenzen definieren und bestehende bearbeiten (siehe auch Bearbeiten und Verwalten von Sequenzen).

Index
Der Index der unten angezeigten Sequenz. Die erste Sequenz hat den Index 0 (Null).

1 - Name
Eindeutiger Name für diese Sequenz (z. B. "Ping empfangen"). Dieser Name dient dazu, die Sequenz zu referenzieren. Es handelt sich nicht um die über die serielle Schnittstelle empfangene Sequenz. Siehe "2 - Sequenz" unten.

2 - Sequenz
Die Zeichenfolge, die von Docklight in den eingehenden seriellen Daten erkannt werden soll.

TIPP: Spezielle Funktionszeichen stehen zur Verfügung, um Verzögerungen zwischen den Zeichen zu erkennen, Handshake-Signaländerungen zu bewerten oder einen Unterbrechungszustand zu erkennen.

3 - Aktion
Die Aktion(en), die durchgeführt werden, wenn Docklight die oben definierte Sequenz erkennt.

Sie haben die Wahl zwischen folgenden Aktionen:

- **Antwort** - Senden Sie nach dem Empfang der Sequenz eine der Sendesequenzen. Nur Sendesequenzen, die keine Platzhalter enthalten, können als automatische
- Antwort verwendet werden. **Kommentar** - Fügen Sie nach dem Empfang der Sequenz einen benutzerdefinierten Kommentar in das Kommunikationsfenster (und die Protokolldatei, falls vorhanden) ein. Für die Erstellung dynamischer
- Kommentartexte stehen verschiedene Kommentar-Makros zur Verfügung.
  Trigger - Auslösen eines Schnappschusses, wenn die Sequenz erkannt wird. Dies ist eine erweiterte Funktion, die im Abschnitt Erfassen einer bestimmten Sequenz
- beschrieben wird...
  Stop - Stoppt die Kommunikation und beendet den Testlauf.
- **Prüfsumme** - Führen Sie eine automatische Validierung einer Prüfsumme durch, einschließlich aller Arten von CRC-Standards wie Modbus, CCITT, CRC32. Legen Sie die Prüfsummen-Spezifikation fest, sowie was mit dem Ergebnis geschehen soll: **Detect Checksum OK** - die empfangenen Daten müssen die gleiche Prüfsumme haben wie der von Docklight berechnete Wert.
  Prüfsumme Falsch - das Gegenteil. Eine nicht übereinstimmende Prüfsumme stellt eine "Sequenzübereinstimmung" dar.
  Beide OK/Falsch - die Sequenz wird immer erkannt. Der Prüfsummenbereich enthält alle ASCII-"1" (HEX 31) für eine übereinstimmende Prüfsumme, oder ASCII-"0" (HEX 30) für eine falsche Prüfsumme.

TIPP: Einen allgemeinen Überblick finden Sie unter Berechnen und Validieren von Prüfsummen und unter Prüfsummenspezifikation für das Textformat, das zur Definition einer Prüfsumme verwendet wird.

Wildcards
Wildcards können verwendet werden, um auf Sequenzen zu prüfen, die einen variablen Teil mit wechselnden Werten haben (z. B. Mess- oder Statuswerte). Siehe Abschnitt Prüfung auf Sequenzen mit zufälligen Zeichen für Details und Beispiele.

Abkürzungen für Steuerzeichen
Die Verwendung von Tastaturkürzeln ist eine große Hilfe bei der Bearbeitung einer

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Sequenz, die sowohl druckbare Zeichen (Buchstaben A-z, Ziffern 0-9, ...) als auch nicht druckbare Steuerzeichen (ASCII-Code 0 bis 31) enthält. Vordefinierte Tastenkombinationen sind:

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Strg+Eingabe für Wagenrücklauf / <CR> / Dezimalcode 13
Strg+Shift+Enter für Zeilenvorschub / <LF> / Dezimalcode 10

Verwenden Sie Optionen... --> Steuerzeichenkurzbefehle, um andere Kurzbefehle zu definieren, die Sie für nützlich halten.

Dokumentation der Sequenz
Fügen Sie hier eine Dokumentation zu dieser Sequenz hinzu. Diese Dokumentation wird auch im Hauptfenster angezeigt, wenn Sie die Sequenz in der Liste Sendesequenzen auswählen.

9.4 Dialog: Protokollierung starten / Protokolldatei(en) erstellen

Menü **Extras > Kommunikationsprotokollierung**

starten ... Format der Protokolldatei
Die verfügbaren Protokollformate sind einfacher Text (.txt), HTML für Webbrowser (.htm) oder RTF Rich Text Format für *Microsoft* Word oder Wordpad (.rtf).

- **Eine einfache Textdatei (.txt)** ist eine gute Wahl, wenn Sie erwarten, dass Ihre Protokolldateien sehr groß werden.

TIPP: Der in *Windows* integrierte Editor Notepad kann beim Öffnen und Bearbeiten größerer Dateien sehr langsam sein. Wir empfehlen den beliebten Open-Source-Editor Notepad++, der unter http://notepad-plus.sourceforge.net verfügbar ist - er ist eine viel schnellere und leistungsfähigere Alternative.

HINWEIS: Es gibt keine Größenbeschränkung für Docklight-Logdateien, abgesehen von den Beschränkungen auf Ihrem Windows-PC. Wir haben Docklight erfolgreich in Langzeitüberwachungen / hochvolumigen Anwendungen getestet und Logdateien mit mehreren GB Größe ohne Stabilitätsprobleme erstellt.

- **HTML-Dateien (.htm)** sind bequemer zu analysieren, da sie alle visuellen Formatierungen der Docklight-Kommunikationsfenster enthalten (Farben, fette und kursive Zeichen). Allerdings ist die Festplattengröße für eine solche Datei größer als für ein reines Textformat, und große HTML-Dateien verlangsamen die gängigen Webbrowser.

  TIPP: Wenn Sie spezielle Anforderungen an das Ausgabeformat haben, können Sie die HTML-Ausgabe anpassen.

- **RTF Rich Text Format (.rtf)** ist eine gute Wahl sowohl für kleine als auch für große Protokolldateien mit formatiertem Text - sowohl *Microsoft* Word als auch Wordpad können schnell durch größere Dateien navigieren, ohne dass sie unempfindlich werden.

HINWEIS: Aufgrund der Besonderheiten des RTF-Dokumentenformats kann Docklight neue Daten nicht effizient an eine bestehende Protokolldatei anhängen, sondern muss zunächst eine temporäre Kopie des bestehenden Protokolls erstellen, was zu zusätzlichen Verzögerungen führen kann. Es wird auch nicht unterstützt, neue Protokolldaten mit anderen Farben und Schrifteinstellungen als am Anfang der Datei anzuhängen.

Verzeichnis und Basisname der Protokolldatei
Wählen Sie hier das Verzeichnis und den Basisdateinamen für die Protokolldatei(en). Der tatsächliche Dateipfad, der für die einzelnen Protokolldateidarstellungen verwendet wird, wird in den Textfeldern im Rahmen "Protokolldateidarstellung" angezeigt.

Überschreibe-/Anfügemodus
Wählen Sie "neue Daten anhängen", wenn Sie nicht wollen, dass Docklight bestehende

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Protokolldateien überschreibt. Docklight fügt dann beim Öffnen/Schließen der Protokolldateien eine "Start Logging / Stop Logging"-Meldung ein. Auf diese Weise können Sie auch im "Anfügemodus" erkennen, wann eine einzelne Protokolldatei begonnen oder beendet wurde.

Vertretung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Für jede Datendarstellung (ASCII, HEX, ...) kann eine eigene Protokolldatei erstellt werden. Wählen Sie mindestens eine Darstellung. Die Protokolldateien haben eine ".txt" oder ".htm" Dateierweiterung. Docklight fügt dem Dateinamen zusätzlich die Darstellungsart hinzu, um die verschiedenen Protokolldateien zu unterscheiden. Wenn der Benutzer z.B. "Test1" als Dateinamen angibt, heißt die ASCII-Logdatei "Test1_asc.txt", während die HEX-Logdatei "Test1_hex.txt" heißt.

Kommunikationsfenster während der Protokollierung deaktivieren
Wenn Sie eine Hochgeschwindigkeits-Kommunikationsverbindung überwachen oder Docklight auf einem langsamen Computer laufen lassen, kann es sein, dass Docklight nicht alle übertragenen Daten verarbeiten kann oder sogar einfriert (keine Reaktion auf Benutzereingaben).
Mit dieser Option können Sie die Ausgabe des Kommunikationsfensters deaktivieren, während die Daten in eine Datei aufgezeichnet werden. Docklight läuft dann viel schneller, da die kontinuierliche Anzeigeformatierung und -aktualisierung erhebliche CPU-Zeit erfordert.

HINWEIS: Weitere Informationen zu Hochgeschwindigkeitsanwendungen finden Sie auch im Abschnitt Wie man die Verarbeitungsgeschwindigkeit erhöht...

9.5 Dialog: HTML-Ausgabe anpassen

(über das Menü **Extras > Kommunikationsprotokollierung starten ...** , wählen Sie dann **HTML-Datei für Webbrowser (.htm)** und klicken Sie auf **HTML-Ausgabe anpassen**)

In diesem Dialog können Sie das Aussehen der HTML-Protokolldateien ändern, indem Sie den HTML-Vorlagencode modifizieren, den Docklight bei der Generierung der HTML-Protokolldaten verwendet.

Sie benötigen ein Grundverständnis von HTML-Dokumenten und CSS-Stilattributen. Wir empfehlen www.htmldog.com (Englisch) oder www.selfhtml.org (Deutsch) für einen schnellen Überblick über diese Themen.

HTML Kopfzeile Vorlage
Die Kopfzeile des HTML-Dokuments. Hier können Sie die Schriftart für die Daten der Protokolldatei ändern, indem Sie die folgenden CSS-Stileigenschaften verwenden:

CSS Style-Attribut	Beschreibung und Beispiel

font-family	Definiert eine oder mehrere Schriftarten (oder: Schriftartenkategorien), die der HTML-Browser zum Drucken eines Textes verwenden soll. Wenn der Browser die erste Schriftart nicht unterstützt, versucht er es mit der zweiten, usw. Die letzte Schriftart definiert normalerweise eine generische Schriftartkategorie, die jeder Browser unterstützt. Beispiele:
	`font-family:'Courier New', Courier, monospace`
	`font-family:'Times New Roman', Times, serif`
	`font-family:arial, helvetica, sans-serif`

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Schriftgröße	Gibt die Schriftgröße an. Es sind sowohl absolute als auch relative Größen möglich. Beispiele für absolute Schriftgrößen:
	`font-size:12pt`
	`font-size:xx-small`
	`font-size:x-small`
	`font-size:small`
	`font-size:medium`
	`font-size:large`
	`font-size:x-large`
	`font-size:xx-large`
	Beispiele für relative Schriftgrößen (relativ zum übergeordneten HTML Element)
	`font-size:kleiner`
	`font-size:größer`

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	`Schriftgröße:90%`

HINWEIS: Verwenden Sie das Semikolon (";") als Trennzeichen zwischen zwei verschiedenen CSS-Stilattributen, z. B.
`font-family:sans-serif; font-size:small`

HINWEIS: Docklight fügt zusätzliche <u> (unterstrichen), <i> (kursiv) und <b> (fett) HTML-Tags ein, wenn diese Formatierungsoptionen in den Anzeigeeinstellungen aktiviert sind. Sie müssen nicht das Attribut **font-style** oder **font-weight** verwenden, um diese Effekte zu erzeugen.

HTML Fußzeile Vorlage
Fügt zusätzlichen Fußzeilentext hinzu und schließt das HTML-Dokument.

Datenelement-Vorlage
Für jede neue Information in der Protokolldatei (Daten von Kanal 1, Daten von Kanal 2 oder ein Kommentartext) wird der HTML-Protokolldatei ein neues -Tag mit einer anderen Textfarbe hinzugefügt.

Der Vorlagencode für den Kopf-, Fuß- und Datenteil enthält Docklight-spezifische Platzhalter, die nicht gelöscht werden dürfen:

Platzhalter	Beschreibung
%BACKCOLOR%	Die Hintergrundfarbe, wie sie in den Anzeigeeinstellungen ausgewählt wurde
%HEADERMSG%	Kopfzeilentext am Anfang der Protokolldatei
%FOOTERMSG%	Fußzeilentext am Ende der Protokolldatei
%DATA%	einen Teil der Daten der Protokolldatei: Kanal-1-Daten, Kanal-2-Daten oder einen Kommentartext
%TEXTCOLOR%	Die für %DATA% anzuwendende Textfarbe, wie in den Anzeigeeinstellungen ausgewählt

Beim Erzeugen einer Protokolldatei ersetzt Docklight die Platzhalter durch die aktuellen Anzeigeeinstellungen und die tatsächlichen Kommunikationsdaten.

9.6 Dialog: Sequenz suchen

Menü **Bearbeiten > Sequenz im Kommunikationsfenster suchen...**

Die Funktion **Sequenz suchen** durchsucht den Inhalt des Kommunikationsfensters. Die Suche wird in der aktuell gewählten Registerkarte des Kommunikationsfensters durchgeführt (ASCII, HEX, Dezimal oder Binär). Sie können Ihre Suchzeichenfolge aber auch in jeder anderen Darstellung definieren.

Die Suche in den Kommunikationsfenstern ist nur möglich, wenn die Kommunikation gestoppt ist.

Sie können nach allem suchen, was bereits als Sende- oder Empfangssequenz definiert ist, oder Sie können eine eigene Suchsequenz definieren.

HINWEIS: Wenn Sie nach einer Sequenz innerhalb des ASCII-Kommunikationsfensters suchen, beachten Sie bitte die folgenden Einschränkungen:
- Die Funktion **Sequenz suchen** ist nicht in der Lage, Sequenzen zu finden, die nicht druckbare Steuerzeichen (ASCII-Dezimalcode < 32) oder andere Sonderzeichen (Dezimalcode > 127) enthalten. Dies ist auf die Art der ASCII-Anzeige zurückzuführen. Verwenden Sie stattdessen die Registerkarte des HEX- oder
- Dezimalkommunikationsfensters für die Suche.
  Im ASCII-Modus behandelt die Funktion **"Find Sequence"** Datums-/Zeitstempel und alle anderen Kommentare wie reguläre Kommunikationsdaten. Im HEX / Dezimal /

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Binärmodus werden alle zusätzlichen Informationen ignoriert, solange sie nicht wie ein Zeichenbyte-Wert aussehen.

9.7 Dialog: Parameter Sendefolge

Geben Sie hier einen oder mehrere Werte für eine Sendesequenz mit Platzhaltern ein. Wie im Dialogfeld "Sende-/Empfangssequenz bearbeiten" können Sie Steuerzeichenkürzel oder Funktionen der Zwischenablage verwenden.
Parameter Nr.
Eine Sendesequenz kann eine beliebige Anzahl von Platzhaltern enthalten. Jeder Satz von aufeinanderfolgenden Platzhaltern wird als eigener Parameter betrachtet. Der Wert für jeden Parameter wird separat eingegeben.

Erforderliche Mindestzeichen
Für jeden '?'-Platzhalter ist genau ein Zeichen erforderlich. Daher ist die Mindestanzahl der erforderlichen Zeichen gleich der Anzahl der '?'-Platzhalter innerhalb eines Parameters.

HINWEIS: Während der Dialog Sendesequenzparameter angezeigt wird, ist die gesamte serielle Kommunikation unterbrochen. Docklight empfängt keine Daten und sendet keine (periodischen) Sendesequenzen.

9.8 Dialog: Projekteinstellungen - Kommunikation

Menü **Extras > Projekteinstellungen... | Kommunikation**

Kommunikationsmodus

Senden/Empfangen
Docklight fungiert sowohl als Sender als auch als Empfänger von seriellen Daten. Dieser Modus wird verwendet, um die Funktionalität oder die Protokollimplementierung eines seriellen Geräts zu testen oder ein serielles Gerät zu simulieren.
Benennungskonventionen: Die empfangenen Daten (RX) werden als "Kanal 1" angezeigt und verarbeitet, die gesendeten Daten (TX) werden als "Kanal 2" angezeigt.

Überwachung
Docklight empfängt serielle Daten auf zwei verschiedenen Kommunikationskanälen. Dieser Modus wird z.B. bei der Überwachung der Kommunikation zwischen zwei Geräten verwendet.
Benennungskonventionen: Die seriellen Daten von Gerät 1 sind "Kanal 1", die Daten von Gerät 2 sind "Kanal 2".

Kommunikationskanäle - Serielle COM-Ports, Docklight TAP/VTP, Netzwerk TCP/UDP, HID, Named Pipes
In Docklight Scripting kann ein Kommunikationskanal konfiguriert werden als

- Serieller COM-Anschluss (RS232, RS422 oder RS485),
- TAP-Anschluss für Docklight Tap-Überwachung
- VTP-Anschluss für Docklight Tap Pro oder Tap 485 Überwachung
- Netzwerkkommunikations-Socket für TCP oder UDP
- HID-Anschlüsse für USB- oder Bluetooth-Geräte
- Benannte Rohre Client

Die folgenden Einstellungen können verwendet werden:

Einstellung / Beispiele	Beschreibung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

COMxxx	Der Kanal ist mit einem seriellen COM-Anschluss verbunden.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

COM1 COM256	Verwenden Sie die Dropdown-Liste, um alle auf Ihrem PC verfügbaren COM-Ports des Windows-Betriebssystems anzuzeigen.
TAPx TAP0 TAP1	Der Kanal ist mit einer der Docklight Tap-Überwachungsdatenrichtungen verbunden. Die TAP-Verbindungen sind nur verfügbar, wenn der Kommunikationsmodus auf 'Überwachung' eingestellt ist, das Docklight Tap eingesteckt ist und die Docklight Tap USB-Gerätetreiber korrekt installiert sind.
VTPx VTP0 VTP1	Der Kanal ist mit einer der Tap Pro / Tap 485 Überwachungsdatenrichtungen verbunden, ähnlich wie bei der Docklight Tap Anwendung mit TAPx Einstellungen.
EntfernterHost:*Entfernter Port* 192.168.1.100:10001 NIC.COM:80 LOCALHOST:504	Der Kanal fungiert als TCP-Client. Beim Start der Kommunikation wird eine Verbindung mit dem angegebenen Host und TCP-Port hergestellt. Für *RemoteHost* können Sie eingeben - eine IP4-Adresse, z. B. 192.168.1.100 ▪ einen Hostnamen, z. B. NIC.COM (für den Zugriff auf einen Server auf das Internet) oder der *Windows* NetBIOS-Name für einen anderen Computer in Ihrem lokalen Netzwerk. ▪ das Schlüsselwort LOCALHOST, das immer auf den Computer, auf dem Docklight läuft. Dies ist gleichbedeutend mit unter Verwendung der Loopback-IP-Adresse 127.0.0.1.
SERVER:*LocalPort* SERVER:10001 SERVER:80 SERVER:504	Der Kanal fungiert als TCP-Server. Wenn die Kommunikation gestartet wird, akzeptiert Docklight eine Verbindung von einem TCP-Client. Wenn ein Client verbunden ist, werden weitere Verbindungsversuche abgewiesen.
PROXY:*Lokaler* *Anschluss* PROXY:10001	Wie SERVER, aber im Überwachungsmodus kontrolliert er den zweiten Kanal entsprechend der vom Server akzeptierten Verbindung. Wenn der zweite Kanal eine Verbindung zwangsweise schließt, trennt der PROXY-Server auch die akzeptierte Verbindung.
UDP:*EntfernterHost*:*Anschluss* UDP:10.0.0.1:8001 UDP:LOCALHOST:10001	Der Kanal verhält sich wie eine UDP-Gegenstelle. Sendedaten werden an das Ziel *RemoteHost*:*Port* gesendet, und Docklight hört auf UDP-Daten an der lokalen UDP-Portnummer *Port.* Wenn Sie eine Kanaleinstellung wie UDP:LOCALHOST:10001 verwenden, erzeugen Sie effektiv eine Rückkopplung, ähnlich wie bei einer seriellen Schnittstelle, bei der alle ausgehenden Daten sofort erhalten.
UDP:*RemoteHost*:*RemotePort*:*LocalPort* UDP:10.0.0.1:8001:8002	Der Kanal verhält sich wie ein UDP-Peer, verwendet aber unterschiedliche Portnummern für ausgehende und eingehende Daten. Die Daten werden an den *RemotePort* übertragen, und Docklight hört am *LocalPort* zu.
UDP:*LocalPort* UDP:10001	Der Kanal fungiert als UDP-Server. Docklight lauscht auf dem *LocalPort* auf UDP-Daten. Sendedaten werden an die Quell-IP und Portnummer des letzten UDP-Pakets übertragen erhalten.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

PIPE:*myNamedPipe* PIPEREAD:*myNamedPipe* PIPEWRITE:*myNamedPipe*	Client-Verbindung zu einer Named Pipe mit Lese-/Schreibzugriff Client-Verbindung mit nur Lesezugriff Nur Schreibzugriff
USBHID:*vendorId*:*productId* USBHID:4D8:F708	USB-HID-Eingangs-/Ausgangsbericht Docklight öffnet eine Verbindung zum angegebenen USB-HID-Gerät (oder einem Bluetooth-HID-Gerät) und ermöglicht das Senden und Empfangen von HID-Eingabe- und Ausgabeberichtsdaten.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

(oder USB-Gerätepfad verwenden) USBHID:\\?\hid#vid_04d8&pid_...	Die Docklight-Kommunikationsanzeige ist reportbasiert: Jeder Eingangsbericht erzeugt einen neuen Docklight-Zeitstempel und druckt die ursprünglichen HID-Berichtsdaten, einschließlich der ID des Eingangsberichts als erstes Byte, falls > 0.
USBHID:*vendorId*:*productId*:P USBHID:4D8:F708:P	USB-HID-Zugang, protokollbasiert. Es werden nur die eigentlichen Nutzdaten angezeigt, ohne das vorangestellte Input-Report-ID-Byte und/oder die nachgestellten Null-Bytes. Zeitstempel werden nach den üblichen Zeitstempelregeln erzeugt, nicht vor jedem Bericht.
USBHID:*vendorId*:*productId*:I USBHID:4D8:F708:I	USB-HID-Zugriff, aber mit variabler Output-Report-ID als Teil der Sendesequenzdaten. In diesem Modus benötigen alle Sendesequenzdefinitionen ein zusätzliches Byte am Anfang der Sequenz, das die tatsächlich zu verwendende Output Report ID festlegt.
USBHID:*vendorId*:*productId*:*[P],outputID,outputPayloadSize* USBHID:4D8:F708:,1,63 USBHID:4D8:F708:P,2	USB HID erweiterte Syntax: *outputID*: falls angegeben, wird diese Output-Report-ID anstelle des Standardwerts Null verwendet. *outputPayloadSize*: falls angegeben, wird die Berichtslänge außer Kraft gesetzt und der Wert, den Windows über HID_CAPS.OutputReportByteLength meldet, ignoriert
PIPE:*myNamedPipe* PIPEREAD:*myNamedPipe* PIPEWRITE:*myNamedPipe*	Client-Verbindung zu einer Named Pipe mit Lese-/Schreibzugriff Client-Verbindung mit nur Lesezugriff Nur Schreibzugriff

Überwachungsmodus - Kanalkombinationen und ihre Anwendungen
Im Monitoring Mode stehen zwei Kommunikationskanäle zur Verfügung, die individuell eingerichtet werden können. Dadurch kann Docklight Scripting in einer Vielzahl von unterschiedlichen Anwendungen und Testumgebungen eingesetzt werden. Nachfolgend finden Sie eine Liste typischer Kanalkombinationen:

Einstellungen des Kommunikationskanals	**Beispiel - Einstellungen**	**Anmeldung**
Kanal 1: COM-Anschluss Ch2: COM-Anschluss	COM1 COM2	Überwachung der seriellen Kommunikation zwischen zwei Geräten mit einem Docklight-Überwachungskabel
Ch1: Docklight Tap Ch2: Docklight Tap	TAP0 TAP1	Überwachung der seriellen Kommunikation zwischen zwei Geräten mit einem Docklight Tap
Ch1: Tap Pro oder 485 Ch2: Gewindebohrer Pro oder 485	VTP0 VTP1	Überwachung der seriellen Kommunikation zwischen zwei Geräten mit einem Docklight Tap Pro oder Docklight Tap 485
Kanal 1: COM-Anschluss Ch2: TCP-Server	COM1 SERVER:10001	Emulation eines seriellen Gerätes auf dem Server. Ein Client kann sich mit dem Docklight-Server an Port 10001 verbinden und mit dem an COM1 angeschlossenen seriellen Gerät sprechen.
Ch1: TCP-Client Ch2: TCP-Server	10.0.0.1:502 SERVER:502	Überwachung einer Client/Server-TCP-Verbindung, bei der Docklight als Gateway

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

		zwischen den beiden Seiten fungiert.
Ch1: UDP Peer Ch2: UDP-Peer	UDP:10.0.0.1:8001 UDP:10.0.0.2:8002	Überwachung und Weiterleitung einer UDP-Übertragung, ähnlich wie im obigen TCP-Beispiel. Beachten Sie, dass Sie für jeden Kanal einen anderen UDP-Port angeben müssen, da jeder Kanal auf seine eigene Portnummer hören muss.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

COM-Port-Einstellungen (nur COM-, TAP- und VTP-Kanäle)

Baudrate
Wählen Sie eine Standard-Baudrate aus der Dropdown-Liste aus, oder verwenden Sie eine Nicht-Standard-Baudrate, indem Sie eine beliebige ganze Zahl zwischen 110 und 9999999 eingeben.

HINWEIS: Nicht standardisierte Baudraten funktionieren möglicherweise nicht an allen COM-Ports, abhängig von den Fähigkeiten des Hardware-UART-Chips Ihres COM-Ports. Sie erhalten keine Warnung, wenn eine Nicht-Standard-Baudrate nicht angewendet werden kann.

HINWEIS: Obwohl Sie in den Projekteinstellungen von Docklight Baudraten von bis zu 9 MBaud festlegen können, bedeutet dies nicht, dass Docklight diesen Datendurchsatz dauerhaft bewältigen kann. Der durchschnittliche Datendurchsatz hängt sehr stark von der Leistung Ihres PCs und den Docklight-Anzeigeeinstellungen ab. Siehe auch Wie Sie die Verarbeitungsgeschwindigkeit erhöhen können.

HINWEIS: Es gibt viele COM-Port-Treiber und -Anwendungen, die keine RS232/422- oder 485-Übertragung verwenden und die keinen der RS232-Kommunikationsparameter benötigen. In einigen Fällen geben solche COM-Port-Treiber sogar einen Fehler zurück, wenn sie versuchen, die RS232-Parameter zu setzen, so dass Docklight den COM-Kanal nicht öffnen kann. Verwenden Sie für diese Anwendungen die Einstellung **Keine** Baudrate.

Datenbits und Stoppbits
Geben Sie hier die Anzahl der Datenbits und Stoppbits an. Wie bei der Baudrate kann es sein, dass einige der verfügbaren Einstellungen von den COM-Port-Geräten auf Ihrem PC nicht unterstützt werden.

Hahn 485 Zeichen. Ebene.
Das Docklight Tap Pro / Tap 485 unterstützt neben den Standard-RS232-Spannungen zusätzliche Spannungspegel:
- **RS485/422** - die differentiellen Spannungspegel für RS485- und RS422-
- Busanwendungen. **Invertiert** - Invertierter RS232/TTL-Modus, bei dem der Markierungszustand (oder logische 1) die positive Spannung und der Leerzeichenzustand (logische 0) die negative Spannung oder Null Volt ist.

Parität
Alle gängigen Paritätsprüfungsoptionen sind hier verfügbar. (Die Einstellungen 'Mark' und 'Space' werden in der Praxis wahrscheinlich nicht verwendet. 'Mark' legt fest, dass das Paritätsbit immer 1 ist, 'Space', dass das Paritätsbit immer 0 ist, unabhängig vom übertragenen Zeichen).

Paritätsfehlerzeichen
Dies ist das Zeichen, das ein ungültiges Zeichen im Datenstrom ersetzt, wenn ein Paritätsfehler auftritt. Sie sollten ein ASCII-Zeichen (druckend oder nicht druckend) angeben, das normalerweise nicht in Ihrem seriellen Datenstrom erscheint. Zeichen können durch Eingabe des Zeichens selbst oder durch Eingabe des dezimalen ASCII-Codes (bitte mindestens zwei Ziffern eingeben) definiert werden.

HINWEIS: Wählen Sie "(ignorieren)" für das Paritätsfehlerzeichen, wenn Sie das Paritätsbit senden/empfangen müssen, Docklight aber alle eingehenden Zeichen beibehalten soll, auch wenn das Paritätsbit falsch ist. Dies ist nützlich für Anwendungen, bei denen ein 9. Bit zur Adressierung und nicht zur Fehlerprüfung verwendet wird.

Baudraten-Scan verwenden - nur VTP-Kanäle

Docklight Tap Pro / Tap 485 Geräte bieten einen Baudraten-Scan / Autodetect-Modus. Um den Baudraten-Scan zu aktivieren, wählen Sie in der Einstellung **Baudrate** die

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Option **Keine** und schließen Sie den Dialog Projekteinstellungen. Starten Sie nun die Kommunikation über das Menü **Ausführen > Kommunikation starten (F5)**. Das Docklight Tap Pro / Tap 485 scannt nun die Kommunikation unabhängig in beide Richtungen. Wenn serielle Daten in beiden Datenrichtungen erkannt werden konnten,

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

die wahrscheinlichsten Einstellungen werden als Kommentare im Kommunikationsfenster angezeigt. Sie werden auch in der Kommunikationsstatusleiste unter der Hauptsymbolleiste angezeigt.

HINWEIS: Die Genauigkeit dieser automatischen Erkennungsfunktion hängt vom tatsächlichen Datenstrom ab, der während des Scans vorliegt. Ein kontinuierlicher Strom hochgradig zufälliger Daten führt zu einer hohen Erkennungsgenauigkeit, während kleine Übertragungen einzelner Bytes oder sich wiederholende Muster zu falschen Schätzungen von Baudraten, Datenbits oder Paritäten führen können.

9.9 Dialog: Projekteinstellungen - Flusskontrolle

Menü **Extras > Projekteinstellungen... | Flusssteuerung**

Dient zur Angabe zusätzlicher Hardware- oder Software-Flusssteuerungseinstellungen für die serielle Kommunikation im Docklight-Sende-/Empfangsmodus.

Unterstützung der Flusskontrolle

Aus
Es wird kein Hardware- oder Software-Flusskontrollmechanismus verwendet. RTS und DTR sind aktiviert, wenn der COM-Anschluss geöffnet wird.

Handbuch
In diesem Modus können Sie die Signale RTS und DTR manuell steuern und den aktuellen Zustand der Leitungen CTS, DSR, DCD und RI anzeigen. Wenn die Flusssteuerung auf "Manuell" eingestellt ist, wird im Docklight-Hauptfenster ein zusätzliches Statuselement angezeigt. Sie können die RTS- und DTR-Leitungen durch einen Doppelklick auf die entsprechende Anzeige umschalten.

HINWEIS: Flusssteuerungssignale werden nicht als Kommunikationsdaten behandelt und werden nicht im Kommunikationsfenster angezeigt oder in einer Datei protokolliert.

Hardware-Handshaking, Software-Handshaking
Unterstützung für RTS/CTS-Hardware-Flusskontrolle und XON/XOFF-Software-Flusskontrolle. Dies sind Experteneinstellungen, die für aktuelle Kommunikationsanwendungen selten benötigt werden.

RS485-Transceiver-Steuerung
Einige RS232-zu-RS485-Konverter erfordern eine manuelle RTS-Steuerung, d. h. das RS232-Gerät (PC) teilt dem Konverter mit, wann er seinen RS485-Treiber für die Übertragung aktivieren soll. Wenn Sie "RS485 Transceiver Control" wählen, setzt der COM-Port RTS auf High, bevor das erste Zeichen einer Sendesequenz übertragen wird, und setzt es auf Low zurück, nachdem das letzte Zeichen übertragen wurde.

HINWEIS: Viele USB-zu-Seriell-Konverter oder virtuelle COM-Port-Treiber implementieren den Windows-Modus RTS_CONTROL_TOGGLE nicht richtig. Wenn Sie Probleme mit der RS485-Transceiver-Steuerung haben, versuchen Sie es mit einem PC mit integrierter COM-Schnittstelle oder einer Standard-PCI-COM-Karte.

9.10 Dialog: Projekteinstellungen - Kommunikationsfilter

Menü **Extras > Projekteinstellungen... |**

Kommunikationsfilter Inhaltsfilter
Verwenden Sie diese Option, wenn Sie die ursprünglichen Kommunikationsdaten auf der seriellen Leitung nicht sehen müssen und nur die zusätzlichen Kommentare benötigen, die durch eine Empfangssequenz eingefügt werden. Dies ist nützlich für Anwendungen mit hohem Datendurchsatz, bei denen die meisten Daten für Tests irrelevant sind und Sie nur auf ganz bestimmte Ereignisse achten müssen. Diese

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Ereignisse (und die zugehörige Display-Ausgabe) können mit Receive Sequences definiert werden.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Kanal-Alias
Damit können Sie die beiden Docklight-Datenrichtungen je nach Anwendungsfall neu beschriften. Z.B. [Docklight] / [Gerät] statt [TX] / [RX]. Oder [Master] / [Slave] anstatt von [TAP0] / [TAP1].

9.11 Dialog: Optionen

Menü **Extras > Optionen...**

Anzeige

Formatierte Textausgabe (Rich Text Format)
dient zur Einstellung des Aussehens des Docklight-Kommunikationsfensters. Die beiden verschiedenen seriellen Datenströme, "Kanal 1" und "Kanal 2", können mit unterschiedlichen Farben und Schriftarten dargestellt werden. In der Standardeinstellung werden unterschiedliche Farben für die beiden Kanäle verwendet, aber auch die Verwendung unterschiedlicher Schriftstile (z.B. Kursivschrift für "Kanal 2") ist möglich. Sie können hier auch die Gesamtgröße der Schrift wählen.

HINWEIS: Wenn Sie die Schriftgröße ändern, wird der Inhalt des Kommunikationsfensters gelöscht. Bei allen anderen Änderungen versucht Docklight, den Inhalt der Anzeige zu erhalten.

Plain Text Output (schnellere Anzeige, aber keine Farben und Schriftarten)
Die formatierte Textausgabe ähnelt der eines Textverarbeitungsprogramms und verbraucht eine beträchtliche Menge an CPU-Zeit. Außerdem ist eine häufige Speicherzuweisung und -freigabe erforderlich, was die Leistung Ihres PCs beeinträchtigen kann. Wenn Sie also eine Hochgeschwindigkeits-Kommunikationsverbindung überwachen, aber dennoch ein Auge auf die übertragenen seriellen Daten haben wollen, sollten Sie das Format "Plain Text Output" verwenden.

Steuerzeichen (ASCII 0 - 31)
Bei Kommunikationsdaten, die sowohl druckenden ASCII-Text als auch nichtdruckende Steuerzeichen enthalten, ist es manchmal hilfreich, die Namen der auftretenden Steuerzeichen im Anzeigefenster des ASCII-Modus zu sehen. Docklight bietet eine optionale Anzeigeeinstellung, die dies ermöglicht. Sie können die Steuerzeichen (mit Ausnahme von CR und LF) auch unterdrücken, wenn dies die Anzeige unübersichtlich machen würde.

Anzeige-Modi

Modi des Kommunikationsfensters
Docklight zeigt standardmäßig vier Darstellungen der seriellen Datenströme an: ASCII, HEX, Dezimal und Binär. Sie können einige dieser Modi deaktivieren, um die Gesamtleistung von Docklight zu erhöhen. Die binäre Darstellung der Daten wird zum Beispiel nur selten benötigt. Wenn Sie den Binärmodus für das Kommunikationsfenster deaktivieren, erhöht sich die Verarbeitungsgeschwindigkeit erheblich. Auch wenn der Binärmodus für das Kommunikationsfenster ausgeschaltet ist, ist die Protokollierung in allen Formaten weiterhin möglich.

Siehe auch die obige Option **"Textausgabe"**.

Datum/Zeit-Stempel

Hinzufügen eines Datums-/Zeitstempels
Docklight versieht alle gesendeten oder empfangenen Daten mit einem Datums- und Zeitstempel. Sie können diesen Datums-/Zeitstempel in das Kommunikationsfenster und die Protokolldatei einfügen, wenn sich die Datenflussrichtung zwischen Kanal 1 und Kanal 2 ändert.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Für Anwendungen, bei denen sich die Richtung des Datenflusses nicht sehr oft ändert, möchten Sie vielleicht zusätzliche Datums-/Zeitstempel in regelmäßigen Zeitabständen haben. Aktivieren Sie dazu die Option **Uhr - zusätzlicher Datums-/Zeitstempel...** und wählen Sie ein Zeitintervall.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Auf einer Halbduplex-Leitung (z.B. 2-Draht-RS485) sind Änderungen der Datenrichtung schwer zu erkennen. In den meisten Anwendungen wird es jedoch eine Pause auf dem Kommunikationsbus geben, bevor ein neues Gerät mit dem Senden beginnt. Verwenden Sie die Option **Pausenerkennung...**, um zusätzliche Zeitstempel einzuführen und die Pausen in Ihrem Kommunikationsprotokoll sichtbar zu machen.

Datum/Uhrzeit-Format
Docklight bietet Zeitstempel mit einer Auflösung von bis zu 1/1000 Sekunden (1 Millisekunde). Für die Kompatibilität zu früheren Docklight-Versionen (V1.8 und kleiner) sind auch 1/100 Sekunden verfügbar.

HINWEIS: Die resultierende Genauigkeit der Zeitmarkierung kann erheblich abweichen, z.B. nur 10-20 Millisekunden. Die tatsächliche Genauigkeit hängt von Ihrer seriellen Kommunikationsausrüstung, Ihrer PC-Konfiguration, den Docklight-Anzeigeeinstellungen (siehe oben) und den Docklight-Expertenoptionen ab. Weitere Informationen finden Sie im Abschnitt Wie man die beste Zeitmessungsgenauigkeit erhält.

Kurzbefehle für Steuerzeichen

Hier können Sie Ihre eigenen Tastenkombinationen für ASCII-Steuerzeichen (ASCII-Code < 32) oder für beliebige Zeichencodes > 126 definieren. Tastaturkürzel können in den folgenden Docklight-Dialogen und -Funktionen verwendet werden

- Dialog: Sendereihenfolge bearbeiten
- Dialog: Empfangssequenz bearbeiten
- Dialog: Sequenz finden
- Dialog: Parameter Sendefolge
- Tastatur-Konsole

Für jedes Zeichen von Dezimalcode 0 bis 31 und von 127 bis 255 können Sie eine Tastenkombination definieren, um dieses Zeichen in eine Sequenz einzufügen (**Shortcut**). Sie können auch einen Buchstaben definieren, der verwendet wird, um dieses Steuerzeichen beim Bearbeiten einer Sequenz im ASCII-Modus anzuzeigen (**Editor**).

Doppelklicken Sie, um den Wert eines Verknüpfungs- oder

Editorfeldes zu ändern. Vordefinierte Shortcuts sind:
Strg+Eingabe für Wagenrücklauf / <CR> / Dezimalcode 13
Strg+Shift+Enter für Zeilenvorschub / <LF> / Dezimalcode 10

9.12 Dialog: Experten-Optionen

Menü **Extras > Expertenoptionen...**

Expertenoptionen sind zusätzliche Einstellungen für spezielle Anwendungen mit zusätzlichen Anforderungen (z. B. hohe Genauigkeit der Zeiterfassung).

Leistung

Kommunikationstreiber-Modus

Verwenden Sie den Modus **Externer Prozess / Prozess mit hoher Priorität**, um ein häufiges Problem für jede Windows-Anwendung im Benutzermodus zu umgehen: Unbestimmte Verzögerungen und zeitliche Ungenauigkeiten können durch die Windows-Task-/Prozessplanung verursacht werden, insbesondere wenn Sie neben Docklight noch andere Anwendungen ausführen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Der Prozessmodus **Extern / Hohe Priorität** wird für eine hochgenaue Überwachung mit geringer Latenzzeit unter Verwendung des Docklight Tap empfohlen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

HINWEIS: Für eine noch höhere und garantierte Genauigkeit der Zeiterfassung verwenden Sie das Docklight Tap Pro / Tap 485 Zubehör. Ihre Genauigkeit hängt nicht von Windows- und Treiberlatenzen ab, und der Prozessmodus "Hohe Priorität" ist für Tap Pro- und Tap 485-Anwendungen nicht erforderlich.

Im Modus **Externer / Hochprioritärer Prozess** wird die Datenerfassung in Docklight zu einem separaten Windows-Prozess mit der Prioritätsklasse "Realtime". Er wird mit höherer Priorität ausgeführt als jede andere Benutzeranwendung oder zusätzliche Anwendungssoftware wie Internet Security / Antivirus. Um optimale Ergebnisse zu erzielen, muss Docklight **als Administrator ausgeführt werden**. Andernfalls wird der Datenerfassungsprozess mit der vom Betriebssystem maximal erlaubten Priorität ausgeführt, aber nicht mit der "Echtzeitklasse".

Extern / Hohe Priorität Der Prozessmodus muss mit Vorsicht verwendet werden, insbesondere wenn Sie eine Hochgeschwindigkeits-Datenverbindung mit großen Datenmengen überwachen wollen. Der PC könnte nicht mehr auf Benutzereingaben reagieren. Um eine solche Situation zu beheben, ziehen Sie einfach "den Stecker": Trennen Sie zunächst die Datenverbindungen oder das Überwachungskabel, um die CPU-Last zu verringern und die Reaktionsfähigkeit wiederherzustellen. Wählen Sie dann in Docklight ■ **Kommunikation beenden**.

HINWEIS: Einige Hintergrundinformationen zur Timing-Genauigkeit finden Sie im Abschnitt Wie erreiche ich die beste Timing-Genauigkeit?

Docklight Überwachungsmodus

Bei der Überwachung der seriellen Kommunikation zwischen zwei Geräten werden alle empfangenen Daten von einem COM-Port standardmäßig auf dem Sendekanal des gegenüberliegenden COM-Ports weitergesendet ("Data Forwarding"). Dies ist für spezielle Anwendungen gedacht, die eine Weiterleitung des seriellen Datenverkehrs durch Docklight über eine Standard-RS232-Verkabelung erfordern.

Verwenden Sie die Expertenoption **Keine Datenweiterleitung** für Anwendungen mit zwei seriellen COM-Ports, bei denen Sie vermeiden müssen, dass Sendedaten gesendet werden. Dies kann verwendet werden, um die Leistung bei der Verwendung eines Docklight-Überwachungskabels zu verbessern oder um Probleme zu umgehen, die durch instabile serielle Gerätetreiber verursacht werden.

Für Docklight-Tap-Anwendungen (z.B. mit Kommunikationskanal TAP0 / TAP1) hat die Einstellung 'Datenweiterleitung' keine Auswirkung. Auf das Docklight Tap wird immer nur lesend zugegriffen, und es werden keine Daten weitergeleitet.

Geräte

Windows COM-Geräte

Verwenden Sie **Deaktivieren Sie die E/A-Fehlererkennung**, wenn Sie wiederholt Fehlermeldungen wie diese erhalten:

```
DOCKLIGHT meldet: Allgemeiner E/A-Fehler auf COM1
```

HINWEIS: Docklight verwendet die serielle Windows-Kommunikation im "Overlapped I/O"-Modus, um die beste Effizienz und Zeitgenauigkeit zu erreichen, und wertet kontinuierlich Fehler aus den entsprechenden Win32-API-Aufrufen aus. In seltenen Fällen, wie z.B. bei COM-Geräten mit fehlerhaften oder veralteten COM-Gerätetreibern, können solche Fehler auch im normalen Lese-/Schreibbetrieb auftreten. In diesem Fall können Sie diese Option verwenden, um zum Verhalten von Docklight V2.2 und früheren Versionen zurückzukehren: Solche Fehler werden einfach ignoriert.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Gewindebohrer Pro / Gewindebohrer 485

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Die Firmware-Update-Funktionen für unser Hardware-Zubehör Docklight Tap Pro / Docklight Tap 485 werden nur in seltenen Fällen benötigt. Z.B. wenn Sie ein älteres Gerät (< Jahr 2017) verwenden, das die Baudraten-Scan-Funktion noch nicht unterstützt.

9.13 Tastatur-Konsole

Mit dem Werkzeug Tastaturkonsole können Sie Tastatureingaben direkt an die serielle Schnittstelle senden. Sie kann über das Menü **Tools >** **Keyboard Console On** aktiviert werden. Die Tastaturkonsole ist nur für den Kommunikationsmodus Senden/Empfangen verfügbar.

Nachdem Sie die Tastaturkonsole aktiviert haben, klicken Sie in das Kommunikationsfenster und geben einige Zeichen ein.

Docklight überträgt die Zeichen direkt über die ausgewählte serielle Schnittstelle. Das Kommunikationsfenster zeigt die Zeichen auf die gleiche Weise an wie eine Sendesequenz.

HINWEIS: Die Tastaturkonsole unterstützt das Einfügen und Übertragen einer Zeichensequenz aus der Zwischenablage mit der Tastenkombination **Strg + V**. Dies ist vergleichbar mit dem Einfügen von Daten aus der Zwischenablage im Dialogfeld "Sendesequenz bearbeiten". Inhalte der Zwischenablage, die die maximale Sequenzgröße von 1024 Zeichen überschreiten, werden abgeschnitten.

HINWEIS: Die Tastaturkonsole ist kein vollwertiges Terminal und unterstützt keine speziellen Terminalstandards wie VT 100. Die Enter-Taste wird als **<CR>** (ASCII 13) plus **<LF>** (ASCII 10) übertragen. Die ESC-Taste sendet **<ESC> (ASCII** 27). Verwenden Sie Steuerzeichenkürzel, um andere ASCII-Steuerzeichen zu senden.

HINWEIS: Die Tastaturkonsole unterstützt nicht das Einfügen von ASCII-Zeichen durch Drücken/Halten von ALT und die Verwendung des numerischen Tastenfelds. Bitte verwenden Sie den Dialog Sendesequenzen bearbeiten in HEX- oder Dezimaldarstellung, um jeden ASCII-Zeichencode > 127 zu erstellen.

9.14 Prüfsummen-Spezifikation

Prüfsummenangaben werden in den Dialogen Edit Send Sequence und Edit Receive Sequence sowie in der Docklight-Scripting-Methode CalcChecksum verwendet. Einen allgemeinen Überblick finden Sie unter Checksummen berechnen und validieren.

Unterstützte Prüfsummen-Spezifikationen / *checksumSpec* Argument

checksumSpec	Angewandter Prüfsummenalgorithmus
MOD256	Einfache 8-Bit-Prüfsumme: Summe über alle Bytes, modulo 256.
XOR	8-Bit-Prüfsumme: XOR über alle Bytes.
CRC-7	7 Bit breite CRC. Wird zum Beispiel in MMC/SD-Kartenanwendungen verwendet. Ein alternativer checksumSpec-Text für denselben Prüfsummentyp wäre: CRC:7,09,00,00,Nein,Nein Siehe die in der letzten Zeile beschriebene Syntax "CRC:*width*, *polynomial...*".
CRC-8	8 Bit breite CRC, z. B. für ATM Head Error Correction. Gleichbedeutend mit: CRC:8,07,00,00,Nein,Nein
CRC-DOW	8 Bit breite CRC, bekannt als DOW CRC oder CCITT-8 CRC. Kann in Dallas iButton(TM) Anwendungen gefunden werden. Gleichbedeutend mit: CRC:8,31,00,00,Yes,Yes

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

MOD65536	Einfache 16-Bit-Prüfsumme: Summe über alle Bytes, modulo 65536.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

CRC-CCITT	CRC mit 16 Bit Breite, wie von CCITT festgelegt. Gleichbedeutend mit: CRC:16,1021,FFFF,0000,No,No
CRC-XMODEM	CRC mit 16 Bit Breite, ähnlich wie CRC-CCITT, aber der Anfangswert ist Null. Gleichbedeutend mit: CRC:16,1021,0000,0000,No,No
CRC-16	CRC mit 16 Bit Breite, wie in IBM Bisynch, ARC verwendet. Gleichbedeutend mit: CRC:16,8005,0000,0000,Yes,Yes
CRC-MODBUS	CRC mit 16 Bit Breite, wie in Modbus verwendet. Ähnlich wie CRC-16, aber mit einem anderen Init-Wert. Dasselbe wie: CRC:16,8005,FFFF,0000,Yes,Yes
CRC-32	32-Bit-CRC, wie in PKZip, AUTODIN II, Ethernet, FDDI verwendet. Gleichbedeutend mit: CRC:32,04C11DB7,FFFFFFFF,FFFFFFFF,Ja,Ja
-MOD256 *oder* LRC	Ähnlich wie MOD256, gibt aber das negative 8-Bit-Ergebnis zurück, so dass die Summe aller Bytes einschließlich der Prüfsumme Null ist. Dies entspricht dem so genannten LRC (Longitudinal Redundancy Check), der z. B. bei POS-Anwendungen verwendet wird.
LRC-ASCII	Wie -MOD256 / LRC, aber es erwartet, dass die Quelldaten HEX-Zahlen als lesbaren ASCII-Text. Siehe das MODBUS-ASCII-Beispiel unten.
CRC:*width*, *polynomial*, *init*, *finalXOR*, *reflectedInput*, *reflectedOutput*	Generischer CRC-Rechner, bei dem alle CRC-Parameter individuell eingestellt werden können: *width* : Die CRC-Breite von 1..32. *polynomial* : HEX-Wert. Das abgeschnittene CRC-Polynom. *init* : HEX-Wert. Der anfängliche Rest, mit dem die Berechnung beginnt. *finalXor* : HEX-Wert. Wendet eine XOR-Operation auf den resultierenden Rest an, bevor er an den Benutzer zurückgegeben wird. *reflectedInput* : Ja = Reflektiert die Datenbytes (MSB wird LSB), bevor sie in den Algorithmus eingespeist werden. *reflectedOutput* : Ja = Reflektiert das Ergebnis nach Abschluss des Algorithmus. Dies geschieht vor der abschließenden XOR-Verknüpfung. Betrieb.

Bemerkungen

Jeder der vordefinierten CRC-Algorithmen (CRC-8, CRC-CCITT, ...) kann durch einen Spezifikationsstring für die generische CRC-Berechnung (CRC:8,07,00...) wie oben beschrieben ersetzt werden. Wir haben unsere Implementierungen sorgfältig getestet und mit der gängigen Literatur und den im CRC-Glossar aufgeführten Ressourcen abgeglichen.

Leider gibt es eine Vielzahl von CRC-Varianten und -Algorithmen, und die Wahl (um nicht zu sagen: das Verständnis) der richtigen CRC-Variante kann eine ziemlich schwierige Aufgabe sein. Eine gute Möglichkeit, um sicherzustellen, dass Ihre CRC-Berechnung sinnvoll ist, besteht darin, sie mit einer ASCII-Testzeichenfolge von "123456789" durchzuführen. Dies ist die am häufigsten verwendete Testzeichenfolge, und viele Spezifikationen beziehen sich auf diese Zeichenfolge und liefern Ihnen die korrekte Prüfsumme, die der CRC bei Anwendung auf diese Zeichenfolge ergeben sollte.

Prüfsummen in Sendesequenz bearbeiten / Empfangssequenz bearbeiten

Wählen Sie auf der Registerkarte **Prüfsumme** eine der vordefinierten Definitionszeichenfolgen aus der Dropdown-Liste oder geben Sie Ihre eigene

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Definition im folgenden Format ein:

[(*startPos*, *len*)] ***checksumSpec*** [A *oder* L] [@ *targetPos*] [# *optionaler Benutzerkommentar*], wobei alles innerhalb von [ ] ein optionaler Teil ist.

Teil	Beschreibung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

checksumSpec	Erforderlich. Zeichenfolge, die den Prüfsummenalgorithmus und seine Parameter entsprechend der obigen Tabelle "checksumSpec Format" angibt.
(*startPos*, *len*) z.B. (1, 4)	Optional. Beginn und Länge des Zeichenbereichs, der zur Berechnung der Prüfsumme verwendet wird. Standardmäßig wird alles vor der Prüfsumme Ergebnis verwendet wird.
A	Optional. Falls verwendet, wird der resultierende Prüfsummenwert in eine HEX-Zahl als lesbaren ASCII-Text umgewandelt. Siehe das MODBUS-ASCII-Beispiel unten.
L	Optional. Little Endian - der resultierende Prüfsummenwert wird mit dem niedrigstwertiges Byte (LSB) zuerst. Standard ist Big Endian / MSB first.
@ *ZielPos* z.B. @ -4	Optional. Gibt die erste Zeichenposition für die Speicherung des resultierenden Prüfsummenwerts an. Standardmäßig schreibt Docklight das Ergebnis der Prüfsumme an die letzten Datenpositionen der Sequenz, es sei denn, Sie haben "A" für ASCII-Ergebnis angegeben. In diesem Fall wird das Ergebnis ein Zeichen vor dem Ende gespeichert, so dass noch Platz für ein "Zeilenende"-Zeichen ist, typischerweise ein CR wie bei Modbus-ASCII.
# *Kommentar*	Sie können einen Kommentar zu dieser Prüfsummenangabe eingeben

Bemerkungen

startPos, *len* und *targetPos* unterstützen auch negative Werte, um Positionen relativ zum Ende der Sequenz und nicht relativ zum Anfang der Sequenz anzugeben.
Beispiele:
startPos ist -4 : Beginn der Berechnung am 4. Zeichen vom Ende.
len ist -1 : alles bis zum Ende der Sequenz verwenden.
targetPos ist -1: Das erste (und einzige) Byte des Ergebnisses wird an der letzten Zeichenposition der Sequenz gespeichert.
targetPos ist -2 : Das Ergebnis wird ab dem 2. Zeichen vom Ende gespeichert.
targetPos ist -3 : Das Ergebnis wird ab dem 3. Zeichen vom Ende her gespeichert.

Beispiele

Prüfsummen - Spezifikation	**Beispiel für eine Sendesequenz**	**Tatsächliche Sendedaten**	**Bemerkungen**
# (aus, keine Prüfsumme)	01 | 02 | 03 | 04 | 05 | 00	01 02 03 04 05 00	nach einem **#** können Sie einen beliebigen Kommentar eingeben, um Ihre Prüfsumme beschreiben
MOD256 # einfache Ein-Byte-Summe über alle bis auf das letzte Zeichen	01 | 02 | 03 | 04 | 05 | **00**	01 02 03 04 05 **0F**	Als Platzhalter für die Prüfsumme wurde eine zusätzliche **00** hinzugefügt, aber Sie können jeden Wert von **00 bis FF** verwenden.
CRC-MODBUS L # Modbus RTU-Prüfsumme. Unteres Byte zuerst ('Little Endian')	01 | 06 | 01 | 02 | 00 | 07 | **FF** | **FF**	01 06 01 02 00 07 **68 34**	**CRC-MODBUS** ist ein 16 Bit-Prüfsumme, die standardmäßig an den letzten beiden Zeichenpositionen in

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

			den Sequenzdaten platziert wird.
(2, -5) LRC-ASCII A @ -4 # MODBUS ASCII Prüfsumme ist LRC über lesbare HEX-Daten, ausgenommen Start	: | 1 | 1 | 0 | 3 | 0 | 0 | 6 | b | 0 | 0 | 0 | 3 | **X** | **X** | r | n	:1103006b00037E<CR><LF>	**LRC-ASCII** behandelt die Sequenzdaten als lesbare HEX-Zeichenfolge, wobei jedes Datenbyte durch zwei Zeichen dargestellt wird. Die Verwendung der Option **A** erzeugt eine lesbare 2

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

':' und Ende 'CR/LF'			Buchstaben-Prüfsummentext anstelle eines einstelligen Ergebnisses. Das **@ -4** setzt das Ergebnis an die 4. Zeichenposition von rechts (wobei die CR/LF am Ende intakt).
CRC:8,07,00,00,No,Ja # CRC mit benutzerdefinierten, nicht standardisierten Spezifikationen	01 | 02 | 03 | 04 | 05 | **00**	01 02 03 04 05 **3D**	Seltene oder benutzerdefinierte CRCs Flavors können von Docklight berechnet werden, aber Sie müssen die erforderlichen CRC-Berechnungsparameter kennen. Weitere Details finden Sie in den Ressourcen, die im CRC-Glossar aufgeführt sind.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Referenz (Skripting)

10 Referenz (Skripting)

10.1 VBScript-Grundlagen

Wenn Sie *Visual Basic®* oder *Visual Basic®* for Applications (VBA) bereits kennen, wird Ihnen VBScript sehr vertraut sein. Schauen Sie sich die unten aufgeführten Definitionen und Beispiele an. Für den Anfang können Sie einige der folgenden Beispiele ausprobieren, indem Sie den Code in den Skript-Editor kopieren und einfügen und das Skript ausführen. Docklight Scripting enthält auch eine Reihe von Beispielskripten, die Sie ausprobieren können.

In diesem Kapitel werden einige grundlegende Funktionen und Merkmale von VBScript vorgestellt. Eine vollständige Referenz finden Sie in der Originaldokumentation von *Microsoft®* an den folgenden Stellen:

- Visual Basic Scripting Edition (oder gehen Sie auf www.microsoft.com und suchen Sie nach "VBScript")
- VBScript Benutzerhandbuch
- VBScript-Sprachreferenz.

TIPP: Verwenden Sie die Funktion ScriptEngine, um herauszufinden, welche Version von VBScript auf Ihrem Computer installiert ist.

HINWEIS: Docklight Scripting führt den VBScript-Code im "sicheren Modus" (safe subset) aus und verbietet potentiell schädliche Aktionen. Das Erzeugen eines "FileSystemObjects" (Datei-E/A) ist zum Beispiel eine der Aktionen, die im sicheren Modus von VBScript nicht erlaubt sind.
Teilmenge. Das Docklight-Skript wird mit einer Fehlermeldung abgebrochen. Bitte kontaktieren Sie unseren E-Mail-Support, wenn Sie spezielle Anforderungen haben und "unsichere" VBScript-Anweisungen verwenden müssen. Auf vielfachen Wunsch ist nun auch die Datei-Eingabe mit Docklight's FileInput möglich / FileOutput-Objekte.

Docklight-spezifische Merkmale

- Docklight Script-Befehle - Das DL-Objekt
- Docklight OnSend / OnReceive Ereignisprozeduren
- Docklight FileInput / FileOutput Objekt zum Lesen und Schreiben von Dateien

VBScript-Grundfunktionen nach Kategorien

- Kontrollstrukturen (Entscheidungsstrukturen, Schleifenstrukturen)
- Variablen, Arrays, Konstanten und Datentypen
- Betreiber
- Datum/Uhrzeit-Funktionen
- Sonstiges

VBScript-Grundfunktionen in alphabetischer Reihenfolge

- Datum Funktion
- Tag Funktion
- Tun bis ...Schleife
- Do...Schleife While
- Für...Weiter
- Stundenfunktion
- Wenn...dann
- Wenn...dann...sonst

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

- InputBox Funktion

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

- LBound-Funktion
- Minutenfunktion
- Monat Funktion
- Jetzt Funktion
- ScriptEngine Funktion
- Zweite Funktion
- Fall auswählen
- Zeit Funktion
- Timer-Funktion
- UBound-Funktion
- Während...Wend
- Jahr Funktion

10.1.1 Copyright-Hinweis

Die folgenden Abschnitte des Kapitels "VBScript-Grundlagen" basieren auf der Hilfedatei Script56.CHM *der Microsoft® Windows* Script V5.6 Dokumentation. Für diese Hilfedatei gilt der folgende Copyright-Hinweis: "© 2001 *Microsoft®* Corporation. All rights reserved."

Die Verwendung von urheberrechtlich geschütztem Material *von Microsoft®* richtet sich nach der "Zehn-Prozent-Regel" von *Microsoft®* (siehe http://www.microsoft.com/permission).

10.1.2 Strukturen kontrollieren

Mit VBScript-Kontrollstrukturen können Sie den Ablauf der Skriptausführung steuern. Mehr über bestimmte Kontrollstrukturen erfahren Sie in den folgenden Themen:

- Entscheidungsstrukturen Eine Einführung in die für die Verzweigung verwendeten Entscheidungsstrukturen.
- Schleifenstrukturen Eine Einführung in Schleifenstrukturen, die zur Wiederholung von Prozessen verwendet werden.

10.1.2.1 Entscheidungs-Strukturen

- **Wenn...dann**

  Verwenden Sie eine If...Then-Struktur, um eine oder mehrere Anweisungen unter Vorbehalt auszuführen. Sie können entweder eine einzeilige Syntax oder eine mehrzeilige Blocksyntax verwenden:

```
Wenn-Bedingung-Dann-Anweisung

Wenn Bedingung Dann
Aussagen
Ende wenn
```

  Die Bedingung ist normalerweise ein Vergleich. Wenn die Bedingung True ist, führt VBScript alle Anweisungen aus, die auf das Then-Schlüsselwort folgen. Sie können entweder eine einzeilige oder eine mehrzeilige Syntax verwenden, um nur eine Anweisung bedingt auszuführen (diese beiden Beispiele sind gleichwertig):

```
If anyDate < Now Then anyDate = Now

If anyDate < Now Then
   anyDate = Now
End If
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beachten Sie, dass bei der einzeiligen Form von If...Then keine End If-Anweisung verwendet wird. Wenn Sie mehr als eine Codezeile ausführen möchten, wenn die Bedingung wahr ist, müssen Sie die mehrzeilige Blocksyntax If...Then...End If verwenden.

-If...Then...Else

Verwenden Sie einen If...Then...Else-Block, um mehrere Anweisungsblöcke zu definieren, von denen jeweils einer ausgeführt wird:

```
If condition1 Then
[statementblock-1]
[ElseIf condition2 Then
[statementblock-2]] ...
[Ausschluss
[Anweisungsblock-n]]
Ende wenn
```

-Koffer auswählen

VBScript bietet die Select Case-Struktur als Alternative zu If...Then...Else für die selektive Ausführung eines Anweisungsblocks aus mehreren Anweisungsblöcken. Eine Select Case-Anweisung bietet ähnliche Möglichkeiten wie die If...Then...Else-Anweisung, macht den Code aber lesbarer, wenn mehrere Möglichkeiten zur Auswahl stehen.

```
' Beispiel
Select Case
   Wochentag(jetzt) Case 2
      DL.AddComment "Montag"
   Fall 3
      DL.AddComment "Dienstag"
      Fall 4
      DL.AddComment "Mittwoch"
   Fall 5
      DL.AddComment
   "Donnerstag" Fall 6
      DL.AddComment "Freitag"
   Case Else
      DL.AddComment
"Wochenende!" End Select
```

10.1.2.2 Schleifen-Strukturen

- **Tun bis ...Schleife**

```
Beispiel
Do Until DefResp = vbNo
   MyNum = Int (6 * Rnd + 1)    ' Erzeugt eine zufällige
  ganze Zahl zwischen 1 und 6.
   DefResp = MsgBox (MeineZahl & "Möchten Sie eine andere
  Zahl?", vbYesNo)
Schleife
```

- **Do...Schleife While**

```
Beispiel
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
  Do
MeineZahl = Int (6 * Rnd + 1) ' Erzeugen einer zufälligen Ganzzahl
    zwischen 1 und 6.
      DefResp = MsgBox (MeineZahl & "Möchten Sie eine andere
    Zahl?", vbYesNo)
  Schleife While DefResp = vbYes
```

- **Während...Wend**

```
  'Beispiel
  Dim Counter
  Zähler = 0     ' Variable initialisieren.
  While Zähler < 20     ' Wert des Zählers testen.
     Zähler = Zähler + 1      ' Zähler
     inkrementieren. DL.AddComment Zähler
  Wend    ' While-Schleife beenden, wenn Zähler > 19
```

- **Für...Weiter**

```
  Beispiel
  Für I = 1 bis 5
        Für J = 1 bis 4
              Für K = 1 bis 3
                 DL.AddComment I & " " & J & " " & K
              Next
        Weiter
  Weiter
```

10.1.3 Variablen, Arrays, Konstanten und Datentypen

Bei der Durchführung von Berechnungen mit VBScript müssen Sie häufig Werte vorübergehend speichern. So können Sie beispielsweise mehrere Werte berechnen, sie vergleichen und je nach Ergebnis des Vergleichs verschiedene Operationen mit ihnen durchführen.

- **Variablen**

Variablennamen folgen den Standardregeln für die Benennung von Dingen in VBScript. Ein Variablenname:

- Muss mit einem alphabetischen Zeichen beginnen.
- Darf keinen eingebetteten Punkt enthalten.
- Darf 255 Zeichen nicht überschreiten.
- Muss in dem Bereich, in dem er deklariert wird, eindeutig sein.

```
  Beispiele
  ApplesSold = 10 ' Der Wert 10 wird an die Variable übergeben.
  ApplesSold = ApplesSold + 1 ' Die Variable wird inkrementiert.
```

- **Arrays**

Mit Arrays können Sie auf eine Reihe von Variablen mit demselben Namen verweisen und eine Zahl (einen Index) verwenden, um sie voneinander zu unterscheiden. Dies hilft Ihnen in vielen Situationen bei der Erstellung von kleinerem und einfacherem Code, da Sie Schleifen einrichten können, die eine beliebige Anzahl von Fällen effizient behandeln, indem Sie die Indexnummer verwenden.

```
  '
  Beispiel
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
Dim A(10)
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
A(0) = 256
A(1) = 324
A(2) = 100
' ...
A(10) = 55
```

LBound-Funktion
Gibt das kleinste verfügbare Subscript für die angegebene Dimension eines Arrays zurück.

Syntax

LBound (*arrayname* **[,***dimension* **])**

Teil	Beschreibung
arrayname	Name der Array-Variable; folgt den Standard-Variablennamen-Konventionen.
Dimension	Optional. Ganze Zahl, die angibt, welche untere Grenze der Dimension zurückgegeben wird. Verwenden Sie 1 für die erste Dimension, 2 für die zweite und so weiter. Wird die *Dimension* weggelassen, wird 1 angenommen.

UBound-Funktion
Gibt das größte verfügbare Subscript für die angegebene Dimension eines Arrays zurück.

Syntax

UBound (*arrayname* **[,***dimension* **])**

Teil	Beschreibung
arrayname	Name der Array-Variable; folgt den Standard-Variablennamen-Konventionen.
Dimension	Optional. Ganze Zahl, die angibt, welche untere Grenze der Dimension zurückgegeben wird. Verwenden Sie 1 für die erste Dimension, 2 für die zweite und so weiter. Wird die *Dimension* weggelassen, wird 1 angenommen.

```
' Beispiel
Dim A(100,3,4)
UBound(A,1) ' ergibt 100
UBound(A,2) ' ergibt 3
UBound(A,3) ' ergibt 4
```

- **Konstanten**

Eine Const-Anweisung kann eine mathematische oder datums-/zeitbezogene Größe darstellen:

```
' Beispiel
Const conPi = 3.14159265358979
```

- **Datenarten**

In VBScript gibt es nur einen einzigen Datentyp, die Variante. Eine Variante ist eine besondere Art von Datentyp, der je nach Verwendung unterschiedliche Informationen enthalten kann.
Da Variant der einzige Datentyp in VBScript ist, ist es auch der Datentyp, der von allen Funktionen in VBScript zurückgegeben wird.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Varianten-Subtypen
Neben den einfachen Klassifizierungen als Zahl oder Zeichenkette kann eine Variante weitere Unterscheidungen über die spezifische Art der numerischen Informationen treffen. So kann eine numerische Information beispielsweise ein Datum oder eine Uhrzeit darstellen. Bei der Verwendung mit anderen Datums- oder Zeitangaben wird das Ergebnis immer als Datum oder Uhrzeit ausgedrückt. Sie können auch eine Vielzahl von numerischen Informationen haben, die von booleschen Werten bis hin zu großen Gleitkommazahlen reichen. Diese verschiedenen Kategorien von Informationen, die in einer Variante enthalten sein können, werden als Subtypen bezeichnet. In den meisten Fällen können Sie einfach die gewünschte Art von Daten in eine Variante eingeben, und die Variante verhält sich so, wie es für die enthaltenen Daten am besten geeignet ist.

Die folgende Tabelle zeigt die Untertypen von Daten, die eine Variante enthalten kann.

Untertyp	Beschreibung
Leere	Variante ist uninitialisiert. Der Wert ist 0 für numerische Variablen oder ein String der Länge Null ("") für String-Variablen.
Null	Die Variante enthält absichtlich keine gültigen Daten.
Boolesche	Enthält entweder Wahr oder Falsch.
Byte	Enthält eine ganze Zahl im Bereich von 0 bis 255.
Integer	Enthält eine ganze Zahl im Bereich von -32,768 bis 32,767.
Währung	-922.337.203.685.477.5808 bis 922.337.203.685.477.5807.
Lang	Enthält eine ganze Zahl im Bereich von -2.147.483.648 bis 2.147.483.647.
Einzeln	Enthält eine Gleitkommazahl mit einfacher Genauigkeit im Bereich von -3,402823E38 bis -1,401298E-45 für negative Werte; 1,401298E-45 bis 3,402823E38 für positive Werte.
Doppelter	Enthält eine doppelgenaue Fließkommazahl im Bereich von -1,79769313486232E308 bis -4,94065645841247E-324 für negative Werte; 4,94065645841247E-324 bis 1,79769313486232E308 für positive Werte.
Datum (Uhrzeit)	Enthält eine Zahl, die ein Datum zwischen dem 1. Januar 100 und dem 31. Dezember 9999 darstellt.
Zeichenfolge	Enthält eine Zeichenkette variabler Länge, die bis zu etwa 2 Milliarden Zeichen lang sein kann.
Objekt	Enthält ein Objekt.
Fehler	Enthält eine Fehlernummer.

10.1.4 Betreiber

- Arithmetik

Beschreibung	Symbol
Potenzierung	^
Unäre Verneinung	-
Multiplikation	*
Abteilung	/

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Ganzzahlige Division	\

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Modulus-Arithmetik	Mod
Zusatz	+
Subtraktion	-
String-Verkettung	&

- Vergleich

Beschreibung	Symbol
Gleichstellung	=
Ungleichheit	<>
Weniger als	<
Größer als	>
Weniger als oder gleich	<=
Größer als oder gleich	>=
Äquivalenz der Objekte	Ist

- Logisch

Beschreibung	Symbol
Logische Verneinung	Nicht
Logische Konjunktion	Und
Logische Disjunktion	Oder
Logisch exklusiv	Xoder
Logische Gleichwertigkeit	Eqv
Logische Implikation	Kobold

10.1.5 Datum/Uhrzeit-Funktionen

- **Datum Funktion**

```
Beispiel Datum Funktion
DL.ClearCommWindows
DL.AddComment Datum    ' gibt das aktuelle Systemdatum aus.
```

- **Zeit Funktion**

```
Beispiel Zeitfunktion
DL.ClearCommWindows
DL.AddComment Zeit    ' gibt die aktuelle Systemzeit aus.
```

- **Timer-Funktion**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
Beispiel für eine Timer-Funktion
Die Funktion Timer gibt die Anzahl der verstrichenen Sekunden
  zurück.
'seit 12:00 AM
(Mitternacht). StartZeit =
Timer
For i = 1 To 1000
Nächste
DL.AddComment "Dauer [Millisekunden] = " & (Timer -
  StartTime) * 1000
```

- **Jetzt Funktion**

```
'Beispiel Now Function
Dim MyVar
MyVar = Now ' MyVar enthält das aktuelle Datum und die aktuelle Uhrzeit.
```

- **Tag Funktion**

```
Beispiel Day Function
DL.AddComment Day(Now)
```

- **Monat Funktion**

```
'Beispiel Monatsfunktion
DL.AddComment Month(Now)
```

- **Jahr Funktion**

```
'Beispiel Year
Function Dim MyDate
MeinDatum = #Dezember 7, 1968#      ' Zuweisen
eines Datums. DL.AddComment Jahr(MeinDatum)
```

- **Stundenfunktion**

```
'Beispiel
Stundenfunktion
DL.AddComment Hour(Now)
```

- **Minutenfunktion**

```
Beispiel Minutenfunktion
DL.AddComment Minute(Now)
```

- **Zweite Funktion**

```
Beispiel Zweite Funktion
DL.AddComment Second(Now)
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.1.6 Sonstiges

-InputBox-Funktion

Zeigt eine Eingabeaufforderung in einem Dialogfeld an, wartet darauf, dass der Benutzer Text eingibt oder auf eine Schaltfläche klickt, und gibt den Inhalt des Textfelds zurück.

TIPP: Verwenden Sie die Docklight-spezifische Methode DL.InputBox2 für ein Dialogfeld, das immer auf demselben Bildschirm wie das Docklight Scripting Hauptfenster erscheint. Oder lesen Sie in der Funktion DL.GetKeyState nach, wie Sie auf Tastatur- oder Mauseingaben warten und reagieren können.

Syntax

result **= InputBox (***prompt***[,** *title***][,** *default***][,** *xpos***][,** *ypos***][,** *helpfile***,** *context***])**

Teil	Beschreibung
Aufforderung	Erforderlich. String-Ausdruck, der als Meldung im Dialogfeld angezeigt wird. Die maximale Länge der Eingabeaufforderung beträgt ca. 1024 Zeichen, abhängig von der Breite der verwendeten Zeichen. Wenn die Eingabeaufforderung aus mehr als einer Zeile besteht, können Sie die Zeilen durch ein Wagenrücklaufzeichen (**Chr(**13**)**), ein Zeilenvorschubzeichen (**Chr(**10**)) oder eine** Kombination aus Wagenrücklauf- und Zeilenvorschubzeichen (**Chr(**13**) & Chr(**10**)**) zwischen jeder Zeile trennen.
Titel	Optional. String-Ausdruck, der in der Titelleiste des Dialogfelds angezeigt wird. Wenn Sie title weglassen, wird der Anwendungsname in der Titelleiste angezeigt.
Standard	Optional. String-Ausdruck, der im Textfeld als Standardantwort angezeigt wird, wenn keine anderen Eingaben gemacht werden. Wenn Sie default weglassen, wird das Textfeld leer angezeigt.
xpos	Optional. Numerischer Ausdruck, der den horizontalen Abstand der linken Kante des Dialogfelds von der linken Kante des Bildschirms in Twips angibt. Bildschirm. Wenn xpos weggelassen wird, wird das Dialogfeld horizontal zentriert.
ypos	Optional. Numerischer Ausdruck, der den vertikalen Abstand der Oberkante des Dialogfelds vom oberen Rand des Bildschirms in Twips angibt. Wenn ypos weggelassen wird, wird das Dialogfeld vertikal etwa ein Drittel des Bildschirms nach unten positioniert.
Hilfedatei	Optional. Zeichenkettenausdruck, der die Hilfedatei identifiziert, die verwendet werden soll, um kontextabhängige Hilfe für das Dialogfeld bereitzustellen. Wenn *helpfile* gleich zur Verfügung gestellt werden, muss auch der *Kontext* angegeben werden.
Kontext	Optional. Numerischer Ausdruck, der die vom Hilfeautor dem entsprechenden Hilfethema zugewiesene Hilfekontextnummer identifiziert. Wenn context angegeben wird, muss auch helpfile angegeben werden.

```
'Beispiel InputBox-
Funktion Dim MyInput
MyInput = InputBox("Bitte Text eingeben", "Mein Titel",
  "Beispieltext")
DL.AddComment MeineEingabe     ' Die aktuelle Eingabe als Kommentar hinzufügen
```

-MsgBox Funktion

Zeigt ein Nachrichtenfeld an, wartet darauf, dass der Benutzer auf eine Schaltfläche klickt, und gibt einen Wert zurück, der angibt, auf welche Schaltfläche der Benutzer

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

geklickt hat.

TIPP: Verwenden Sie die Docklight-spezifische Methode DL.MsgBox2 für eine Messagebox, die immer auf dem gleichen Bildschirm wie das Docklight Scripting Hauptfenster erscheint. Oder lesen Sie in der Funktion DL.SetUserOutput nach, wie Sie einen zusätzlichen Benutzerausgabebereich erstellen und zusätzliche Benutzerinformationen anzeigen können.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Syntax

result **= MsgBox (***prompt***[,** *buttons***][,** *title***][,** *xpos***][,** *ypos***][,** *helpfile***,** *context***])**

Teil	Beschreibung
Aufforderung	Erforderlich. Wie die obige Funktion InputBox.
Tasten	Optionale, gemeinsame Werte sind eine Kombination (Summe) der folgenden Konstanten: Konstante Wert Beschreibung vbOKOnly 0 Nur OK-Schaltfläche (Standard) vbOKCancel 1OK- und Abbrechen-Schaltflächen vbAbortRetryIgnore 2Schaltflächen Abbrechen, Wiederholen und Ignorieren vbYesNoCancel 3Ja-, Nein- und Abbrechen-Schaltflächen vbYesNo 4Ja- und Nein-Schaltflächen vbRetryCancel 5Schaltflächen für Wiederholung und Abbrechen vbCritical 16Kritische Meldung vbQuestion 32Warnungsabfrage vbExclamation 48Warnung Meldung vbInformation 64Informationsmeldung vbDefaultButton1 0 Erste Schaltfläche ist Standard (Default) vbDefaultButton2 256Zweite Schaltfläche ist standardmäßig vbDefaultButton3 512Die dritte Schaltfläche ist voreingestellt TIPP: Eine vollständige Liste aller verfügbaren Konstanten finden Sie in der Microsoft VBA-Dokumentation für MsgBox.
Titel	Optional. Wie die obige Funktion InputBox.
xpos, ypos, helpfile, Kontext	Optional. Wie die obige Funktion InputBox.
Ergebnis	Gibt die Benutzeraktion zurück: Konstante Wert Beschreibung vbOK 1OK-Taste gedrückt vbCancel 2 Abbrechen-Taste gedrückt vbAbort 3Abbruch-Taste gedrückt vbRetry 4Retry-Taste gedrückt vbIgnore 5Ignore-Taste gedrückt vbYes 6Ja-Taste gedrückt vbNo 7 Keine Taste gedrückt

```
'Beispiel MsgBox-Funktion
result = MsgBox("Diesen Test durchführen?", 3,
"Mein Titel") If result = 6 Then
     DL.AddComment "Ja-Taste gedrückt"
ElseIf result = 7 Then
     DL.AddComment "Keine Taste gedrückt"
Sonst
     DL.AddComment "Abgebrochen"
Ende wenn
```

-ScriptEngine Funktion

Gibt eine Zeichenkette zurück, die die verwendete Skriptsprache angibt.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Verwenden Sie das folgende Skriptbeispiel, um die vollständige Beschreibung der Skriptsprache und der Versionsnummer zu erhalten.

```
Beispiel für die Verwendung der
ScriptEngine-Funktion DL.AddComment
GetScriptEngineInfo

Function GetScriptEngineInfo
    Dim s
    s = "" ' String mit den erforderlichen
    Informationen erstellen. s = ScriptEngine
    & " Version "
    s = s & ScriptEngineMajorVersion & "."
    s = s & ScriptEngineMinorVersion & "."
    s = s & ScriptEngineBuildVersion
    GetScriptEngineInfo = s ' Rückgabe der Ergebnisse.
Funktion beenden
```

10.2 Docklight Script-Befehle - Das DL-Objekt

Das globale DL-Objekt wird verwendet, um aus einem VBScript-Programm heraus auf Docklight-spezifische Funktionen zuzugreifen.

DL Methoden

DL.AddComment
DL.ClearCommWindows
DL.GetReceiveCounter
DL.GetDocklightTimeStamp
DL.OpenProject
DL.Pause
DL.Beenden
DL.ResetReceiveCounter
DL.SendSequence
DL.StartCommunication
DL.StopCommunication
DL.StartLogging
DL.StopLogging
DL.WaitForSequence

DL-Methoden (Advanced)

DL.CalcChecksum
DL.ConvertSequenceData
DL.GetChannelSettings
DL.GetChannelStatus
DL.GetCommWindowData
DL.GetEnvironment
DL.GetHandshakeSignals
DL.GetKeyState
DL.GetReceiveComments
DL.InputBox2
DL.LoadProgramOptions
DL.PlaybackLogFile
DL.SaveProgramOptions
DL.SetChannelSettings
DL.SetContentsFilter
DL.SetHandshakeSignals
DL.SetUserOutput
DL.SetWindowOutput

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

DL.ShellRun
DL.UploadFile

DL-Eigenschaften

DL.NoOfSendSequences
DL.NoOfReceiveSequences

Zusätzliche Docklight-Scripting-Funktionen

OnSend / OnReceive Ereignisprozeduren
FileInput- / FileOutput-Objekte zum Lesen und Schreiben von Dateien Seitenkanäle - Verwendung mehrerer Datenverbindungen

10.2.1 Methoden

10.2.1.1 AddComment

Fügt dem Kommunikationsdatenfenster und der Protokolldatei einen benutzerdefinierten Text hinzu.

Rückgabewert

Leere

Syntax

DL.AddComment [*comment***] [,** *timeStampAfterComment***] [,** *lineBreakAndPadding***]**

Die Syntax der AddComment-Methode besteht aus diesen Teilen:

Teil	Beschreibung
Kommentar	Optional. Zeichenkette mit dem Kommentar, der dem/den Kommunikationsfenster(n) oder der/den Protokolldatei(en) hinzugefügt werden soll. Wenn der *Kommentar* ausgelassen wird, erzeugt **AddComment** nur einen Zeilenumbruch.
zeitStempelNachKomm e nt	Optionaler boolescher Wert. False (Standard) = Kein zusätzlicher Zeitstempel. True = Hinzufügen eines Zeitstempels nach dem Kommentar. Der Zeitstempel wird bei der Verarbeitung des nächsten seriellen Datenzeichens hinzugefügt, nicht unmittelbar nach dem Drucken des Kommentars. Dies ist vergleichbar mit der Option "Zusätzlicher Zeitstempel..." im Dialogfeld "Empfangsreihenfolge".
lineBreakAndPadding	Optionaler boolescher Wert. True (Standard) = Zusätzliche Leerzeichen werden vor und nach dem Text hinzugefügt, um ihn von den Kommunikationsdaten zu trennen. Nach dem Kommentar wird ein Zeilenumbruch eingefügt. False = Keine zusätzlichen Leerzeichen oder Zeilenumbrüche. Dies ist besonders nützlich in Kombination mit der Option Kommunikationsfilter, wenn Sie die eigentliche Bildschirmausgabe vollständig mit der Methode **AddComment** erstellen wollen.

Bemerkungen

AddComment unterstützt die Empfangssequenz-Kommentarmakros **%_S** (Glocke, "Piepssignal"), **%_L** (Zeilenumbruch) und **%_T** (Zeitstempel) zur Vereinfachung. Siehe Kommentar-Makros für Details.

Sie können ASCII-Steuerzeichen wie den Dezimalcode 08 (Backspace) nicht zur Emulation von Terminalfunktionen bzw. zur Formatierung der Anzeige verwenden. Die

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

einzige Ausnahme ist der Dezimalcode 07 (Glocke), der je nach Windows-Soundschema zur Erzeugung eines "Piepssignals" verwendet werden kann.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiel

```
' Beispiel AddComment

DL.ClearCommWindows
DL.AddComment "Hallo Welt!"
' Zusätzlicher Zeilenumbruch
DL.AddComment
Verwenden Sie den Operator '&', um Zeichenketten und andere
Variablen zu verketten.
r1 = 10
r2 = 20
DL.AddComment "Ergebnis 1 = " & r1 & "  Ergebnis2 = " &
r2 ' Die VBScript-Konstante vbCrLf kann für eine
' zusätzlicher Zeilenumbruch, auch
DL.AddComment
DL.AddComment "Ergebnis 1 = " & r1 &
vbCrLf DL.AddComment "Ergebnis 2 = " & r2

' Die Deaktivierung des Zeilenumbruchs und der Füllzeichen
gibt Ihnen ' eine bessere Kontrolle über die tatsächliche
Ausgabe
DL.AddComment vbCrLf + "Hier ein paar Infos", False, False
DL.AddComment "rmation. " + vbCrLf, False, False

' Ein "Piep"-Signal zur
Benachrichtigung des Benutzers
DL.AddComment Chr(7)
```

10.2.1.2 ClearCommWindows

Löscht den Inhalt des Kommunikationsfensters. Dies gilt für alle vier Darstellungen (ASCII, HEX, Dezimal, Binär) des Kommunikationsfensters.

Rückgabewert

Leere

Syntax

DL.ClearCommWindows

Beispiel

```
' Beispiel ClearCommWindows

' Neustart DL.ClearCommWindows
DL.AddComment "Testlauf
gestartet!"
```

10.2.1.3 GetReceiveCounter

Gibt den aktuellen Wert des Trefferzählers für die angegebene Empfangssequenz zurück. Der Zähler wird jedes Mal inkrementiert, wenn die Receive Sequence im eingehenden Datenstrom erkannt wird. Er kann mit dem Befehl ResetReceiveCounter zurückgesetzt werden. Die Befehle OpenProject und StartCommunication setzen den Trefferzähler ebenfalls auf Null zurück.

Rückgabewert

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Lang

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Syntax

Ergebnis **= DL.GetReceiveCounter(** *NameOderIndex* **)**

Die Syntax der Methode **GetReceiveCounter** besteht aus folgenden Teilen:

Teil	Beschreibung
nameOrIndex	Erforderlich. Zeichenfolge, die den Namen oder den Sequenzindex einer Empfangssequenz enthält.

Bemerkungen

Siehe auch WaitForSequence

Beispiel

Siehe WaitForSequence

10.2.1.4 GetDocklightTimeStamp

Liefert den aktuellen Docklight-Datums-/Zeitstempel gemäß den folgenden Einstellungen:

1. Das Format des Docklight-Datums-/Zeitstempels, das im Dialogfeld Optionen ausgewählt wurde:
   - Zeitstempel
   - Datumsstempel
   - Verwendung von Zeitstempeln mit einer Genauigkeit von 1/100 Sekunden
2. Die Windows-Einstellung für **Region und Sprache > Formate > Kurzes Datum** und **lange Uhrzeit**

Die Funktion **GetDocklightTimeStamp** ist besonders nützlich, um mit der Methode AddComment zusätzliche Zeitinformationen zu drucken.

Rückgabewe

rt String

Syntax 1

result **=**

DL.GetDocklightTimeStamp()

Bemerkungen (Syntax 1)

GetDocklightTimeStamp fügt der Datums-/Zeitangabe ein Leerzeichen am Ende hinzu. Dies hat historische Gründe und dient der Kompatibilität. Siehe Syntax 2 für eine gekürzte Version. Siehe auch die Methode AddComment.

Beispiel 1

```
' Beispiel GetDocklightTimeStamp

DL.ClearCommWindows
DL.StartCommunication
DL.AddComment "Kommunikation begann um " &
DL.GetDocklightTimeStamp()
DL.AddComment "Ich warte auf Daten..."
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' Endlosschleife, um zu verhindern, dass das Skript sofort
beendet wird
Do
    DL.Pause 1 ' (die Pause reduziert die CPU-Last im
Leerlauf) Schleife
```

Syntax 2

result **= DL.GetDocklightTimeStamp([** *myDateTime* **] [,** *milliseconds* **] [,** *trimmed* **])**

Teil	Beschreibung
myDateTime	Optional. eine VBScript-Datums-(Zeit)-Variable, die die Datums-/Zeitinformationen in der Auflösung "1 Sekunde" liefert. 0 (Standard) = Docklights eigene Zeitbasis verwenden.
Millisekunden	Optionaler ganzzahliger Wert mit entsprechenden Millisekunden von 0..999. -1 (Standard) = Docklights eigene Zeitbasis verwenden. -2 = Fügen Sie den Millisekundenteil NICHT hinzu.
getrimmt	Optional. True = Entfernen Sie das Leerzeichen am Ende (siehe Syntax 1). False (Standard) = aus Kompatibilitätsgründen das Originalformat verwenden.

Bemerkungen (Syntax 2)

Die erweiterte Syntax wird in der Regel für die Formatierung von Zeitinformationen über die Empfangssequenz verwendet, die innerhalb einer Sub DL_OnReceive()-Ereignisprozedur erhalten werden. Siehe das Beispiel 2.

Das Argument *milliseconds* = -2 ist nützlich bei der Erstellung einer Exportdatei für Software wie Excel, die ein Standard-Windows-Zeitformat ohne Millisekunden erwartet.

10.2.1.5 OpenProject

Öffnet eine bestehende Docklight-Projektdatei (.ptp-Datei).

Rückgabewert

Leere

Syntax

DL.OpenProject *filePathName*

Die Syntax der OpenProject-Methode besteht aus diesen Teilen:

Teil	Beschreibung
filePathName	Erforderlich. Zeichenfolge, die den Dateipfad (Verzeichnis und Dateiname) der zu öffnenden Docklight-Projektdatei (.ptp-Datei) enthält. Die Dateierweiterung .ptp kann weggelassen werden. Wenn kein Verzeichnis angegeben wird, verwendet Docklight das aktuelle Arbeitsverzeichnis.

Bemerkungen

Wenn *filePathName* keine gültige Docklight-Projektdatei ist oder nicht existiert, meldet Docklight einen Fehler und die Skriptausführung wird abgebrochen.

Wenn *filePathName* eine leere Zeichenfolge ist, wird ein Dateidialog zur Auswahl einer Projektdatei angezeigt.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Alle Zähler für die Empfangsreihenfolge werden beim (erneuten) Öffnen eines Docklight-Projekts zurückgesetzt, siehe die Funktion ResetReceiveCounter.

Beispiel

```
' Beispiel OpenProject

' Laden einer Docklight-Projektdatei
DL.OpenProject "D:\Meine Docklight-Dateien\Test.ptp"

' Laden Sie die Datei 'Test.ptp' aus dem aktuellen
Arbeitsverzeichnis DL.OpenProject "Test"
```

10.2.1.6 Pause

Pausiert die Ausführung des Skripts für eine bestimmte Anzahl von Millisekunden.

Rückgabewert

Leere

Syntax

DL.Pause *Millisekunden*

Die Syntax der Pause-Methode besteht aus diesen Teilen:

Teil	Beschreibung
Millisekunden	Erforderlich. Langer Wert für die Verzögerung in Millisekunden. Mindestwert ist 0 (**Pause** kehrt sofort zurück). Der Höchstwert ist 86000000 (23,88 Stunden).

Bemerkungen

Docklight im Allgemeinen und die Pause-Funktion bieten keine exakte Zeitmessung mit Millisekundengenauigkeit, so dass die tatsächliche Verzögerung vom Millisekundenwert abweichen kann.

Während einer **Pause** können keine DL_OnReceive()-Aufrufe verarbeitet werden. Wenn Sie während der Wartezeit **DL_OnReceive()**-Ereignisse verarbeiten müssen, sehen Sie sich den in Beispiel 2 beschriebenen **pauseWithEvents()**-Code an.

Beispiel

```
' Beispiel Pause

' Sende einen
Testbefehl
DL.SendSequence "Test1"
' 5 Sekunden
Verzögerung DL.Pause
5000
' Einen weiteren Befehl
senden DL.SendSequence
"Test2"

' Typische Hauptschleife zur
Verarbeitung von Daten Do
   DL.Pause 1 ' CPU-Belastung reduzieren
   countSomeThings = DL.GetReceiveCounter(1)
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
    ' ... weitere Dinge tun ...
Schleife
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.2.1.7 Beenden Sie

Stoppt das Docklight-Skript sofort.

Rückgabewert

Void

Syntax

DL.Quit

Bemerku

ngen

Wenn die Kommunikation mit einem Skriptbefehl gestartet wurde (siehe StartCommunication), wird die Kommunikation ebenfalls beendet. Wenn eine Protokolldatei mit StartLogging geöffnet wurde, wird die Datei geschlossen. Dateien, die mit FileInput oder FileOutput geöffnet wurden, werden ebenfalls geschlossen.

Die Verwendung der in VBScript eingebauten "Stop"-Anweisung oder anderer VBScript-Debugging-Funktionen, die den Programmablauf verändern, ist in Docklight Scripting nicht möglich. Verwenden Sie immer die Anweisung **DL.Quit**, um die Skriptausführung zu beenden.

10.2.1.8 ResetReceiveCounter

Setzt einen oder alle Empfangssequenz-Trefferzähler zurück. Setzt auch den Suchalgorithmus zurück, der den Zeichenstrom auf eine übereinstimmende Empfangssequenz prüft (siehe Beispielcode unten).

Rückgabewert

Leere

Syntax

DL.ResetReceiveCounter [*nameOrIndex***]**

Die Syntax der ResetReceiveCounter-Methode besteht aus folgenden Teilen:

Teil	Beschreibung
nameOrIndex	Optional. Zeichenfolge, die den Namen oder den Sequenzindex einer Empfangssequenz enthält. Wenn angegeben, wird nur der entsprechende Zähler zurückgesetzt. Wenn *nameOrIndex* weggelassen wird, werden alle Zähler zurückgesetzt.

Bemerkungen

Siehe auch GetReceiveCounter und WaitForSequence

Beispiel

Siehe WaitForSequence für ein einfaches Beispiel.

Eine zweite Anwendung wird weiter unten demonstriert - das Zurücksetzen der Empfangssequenzerkennung jedes Mal, wenn eine neue Sendesequenz übertragen wird. Dies ist besonders nützlich, wenn Docklight ein serielles Gerät testet und die Sequenzerkennung nicht durch unvollständige oder fehlerhafte Pakete verwirrt werden soll, die zuvor empfangen wurden. Siehe auch DL_OnSend().

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' Beispiel ResetReceiveCounter
' Sequenzerkennung zurücksetzen, wenn eine neue Sequenz gesendet wird
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' Endlosschleife, um zu verhindern, dass das Skript sofort
beendet wird
Do
    DL.Pause 1 ' (die Pause reduziert die CPU-Last im
Leerlauf) Schleife

Sub DL_OnSend()
    DL.ResetReceiveCounter
End Sub
```

10.2.1.9 SendSequence

Sendet eine Sendesequenz oder eine benutzerdefinierte Datensequenz. Startet die Kommunikation, wenn sie nicht bereits läuft (siehe StartCommunication).

Rückgabewert

Leere

Syntax 1

DL.SendSequence *nameOrIndex* **[,** *Parameter***] [,** *Darstellung***]**

Sendet die Sendesequenz, die *nameOrIndex* entspricht. Die **SendSequence**
Die Methodensyntax 1 besteht aus diesen Teilen:

Teil	Beschreibung
nameOrIndex	Erforderlich. Zeichenfolge, die den Namen der Sendesequenz enthält. Die erste Sendesequenz aus der Liste mit einem Namen, der mit *nameOrIndex* übereinstimmt, wird verwendet. Alternativ können Sie einen Ganzzahlwert übergeben, der den Sequenzindex angibt. Der gültige Bereich für den Sequenzindex reicht von 0 bis (NoOfSendSequences - 1).
Parameter	Optional. Zeichenfolge, die einen oder mehrere Parameterwerte für eine Sendesequenz mit Platzhaltern enthält/enthalten. Die Parameter werden standardmäßig in ASCII-Darstellung übergeben. Das Leerzeichen wird verwendet, um mehrere verschiedene Parameter für verschiedene Platzhalterbereiche zu trennen. Um Parameter in HEX-, Dezimal- oder Binärdarstellung zu übergeben, verwenden Sie das unten beschriebene optionale Darstellungsargument. In HEX-, Dezimal- oder Binärdarstellung wird das Komma (",") als Trennzeichen verwendet. Trennzeichen zwischen mehreren verschiedenen Parametern.
Vertretung	Optional. String-Wert zur Festlegung des Formats für die Parameterliste "A" = ASCII (Standard), "H" = Hex, "D" = Dezimal oder "B" = Binär.

Bemerkungen (Syntax 1)

Wenn das Argument *parameters die* falsche Anzahl von Parametern angibt oder die Parameterlänge nicht mit der entsprechenden Wildcard-Region übereinstimmt, gibt Docklight keinen Fehler aus, sondern wendet die folgenden Regeln an:

- Wenn zu wenige Parameter angegeben werden oder die Parameterzeichenfolge zu kurz ist, werden alle verbleibenden Platzhalter mit einem Leerzeichen aufgefüllt. Wenn Sie die *Darstellung* =
  "A" (ASCII) werden die Wildcards mit Leerzeichen (ASCII-Code 32) aufgefüllt. Bei allen anderen Formaten werden die Platzhalter mit dem ASCII-Code 0 aufgefüllt.
-

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Wenn zu viele Parameter angegeben werden oder die Parameterzeichenfolge zu lang ist, werden die Parameter abgeschnitten oder ignoriert.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Syntax 2

DL.SendSequence "", *customSequence* **[,** *representation* **]**

Sendet eine benutzerdefinierte Datensequenz aus. Die SendSequence-Methode Syntax 2 besteht aus den folgenden Teilen:

Teil	Beschreibung
benutzerdefinierteSequenc e	Erforderlich. Zeichenfolge, die die zu sendende Sequenz enthält. Die Sequenz wird standardmäßig in ASCII-Darstellung übergeben. Für HEX-, dezimale oder binäre Sequenzdaten verwenden Sie das optionale Darstellungsargument unten beschrieben.
Vertretung	Optional. String-Wert zur Festlegung des Formats für *customSequence*. "A" = ASCII (Standard), "H" = HEX, "D" = Dezimal oder "B" = Binär.

Beispiel

```
' Beispiel SendSequence

' Vordefinierte
Sendesequenzen ' (0) Test:
Test
' (1) Eins: Eins<#><#><##><CR><LF>
' (2) Zwei: Eins<?><?><?>Zwei<#><#><#>

DL.StartCommunication
DL.ClearCommWindows
' Sendesequenz ohne Parameter
DL.SendSequence "Test"
' Sendesequenz mit einem Parameter
DL.SendSequence "One", "100"
' Sendesequenz mit zwei Parametern
DL.SendSequence "Zwei", "100 20"
' Übergabe von zwei Parametern in HEX-Darstellung,
einschließlich Leer- und Steuerzeichen
DL.SendSequence "Zwei", "20 31 20, 30 0D 0A", "H"
' Benutzerdefinierte Sequenzdaten senden, nicht über eine
vordefinierte Sendesequenz
DL.SendSequence "", "Benutzerdefinierte Daten"

' Und nun eine Schleife und die
Schleifenvariable ' für die Sendesequenz-
Parameterwerte verwenden For i = 1 To 10
      parString = i & " " & i+1 ' zur Trennung der Parameter ein
Leerzeichen verwenden
      DL.SendSequence "Zwei", parString
Nächster

DL.StopCommunication
```

Nach der Ausführung des Skripts könnte das Docklight-Kommunikationsfenster wie folgt aussehen:

```
08/05/2008 13:50:35.622 [TX] - Test
08/05/2008 13:50:35.631 [TX] - One100<CR><LF>

08/05/2008 13:50:35.665 [TX] - One100Two20
08/05/2008 13:50:35.682 [TX] - Eins 1 Zwei0<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
08/05/2008 13:50:35.699 [TX]       -
                              Benutzerdefin
                               ierte Daten
08/05/2008 13:50:35.713 [TX] - Eins1 Zwei2
08/05/2008 13:50:35.745 [TX] - Eins2 Zwei3
08/05/2008 13:50:35.771 [TX] - Eins3 Zwei4
08/05/2008 13:50:35.807 [TX] - Eins4 Zwei5
08/05/2008 13:50:35.846 [TX] - Eins5 Zwei6
08/05/2008 13:50:35.878 [TX] - Eins6 Zwei7
08/05/2008 13:50:35.907 [TX] - Eins7 Zwei8
08/05/2008 13:50:35.922 [TX] - Eins8 Zwei9
08/05/2008 13:50:35.955 [TX]    - Eins9
                                 Zwei10
08/05/2008 13:50:35.987 [TX]    - Eins10
                                 Zwei11
```

10.2.1.10 StartCommunication

Öffnet die Kommunikationsschnittstelle(n) und aktiviert die Datenübertragung. Dies entspricht dem Docklight-Menü **Ausführen > Kommunikation starten**

Rückgabewert

Leere

Syntax

DL.StartCommunication

Bemerkungen

Die Methoden SendSequence, WaitForSequence und UploadFile öffnen automatisch den/die Kommunikationsport(s), wenn sie nicht zuvor mit der Methode **StartCommunication** geöffnet wurden.

Siehe auch StopCommunication.

10.2.1.11 StopCommunication

Stoppt die Datenübertragung und schließt die Kommunikationsschnittstelle(n). Dies entspricht dem Docklight-Menü **Run > Stop communication.**

Rückgabewert

Leere

Syntax

DL.StopCommunication

Bemerkungen

Weitere Informationen finden Sie in der Methode StartCommunication.

10.2.1.12 StartLogging

Legt neue Protokolldateien an und startet die Aufzeichnung der ein- und ausgehenden seriellen Daten. Dies entspricht dem Docklight-Menü **Kommunikationsprotokollierung starten ...**

Rückgabewert

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Leere

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Syntax

DL.StartLogging *baseFilePath* **[,** *appendData***] [,** *representations***] [,** *format***] [,** *highspeed***] [,** *noHeaders***]**

Die Syntax der StartLogging-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
baseFilePath	Erforderlich. Zeichenfolge, die das Verzeichnis und den Namen der Basisdatei für die Protokolldatei(en) enthält.
appendData	Optionaler boolescher Wert. True (Standard) = Anhängen der neuen Daten an die vorhandene(n) Protokolldatei(en). False = Vorhandene Protokolldatei(en) werden überschrieben. Zuvor gespeicherte Protokollierungsdaten gehen verloren.
Vertretungen	Optionale Zeichenfolge zur Auswahl der Darstellung der Protokolldatei. "A" (ASCII), "H" (HEX), "D" (Dezimal) und/oder "B" (Binär). Standardwert ist "AHDB". (erstellen Sie alle vier Darstellungen ASCII, HEX, Dezimal, Binär).
Format	Optional Ganzzahliger Wert. 0 (Standard) = einfache Textdateien (.txt) erstellen 1 = HTML-Dateien für Webbrowser erstellen (.htm) 2 = RTF Rich Text Format Dateien (.rtf) erstellen HINWEIS: Aus Kompatibilitätsgründen zu V2.2. und früher ist es auch möglich, zu verwenden: Falsch = einfacher Text (.txt) Wahr = HTML (.htm)
Highspeed	Optionaler boolescher Wert. False (Standard) = nicht verwendet True = Kommunikationsfenster während der Protokollierung deaktivieren (z. B. zur Überwachung von Hochgeschwindigkeitsverbindungen auf einem langsamen PC).
noHeaders	Optionaler boolescher Wert. False (Default) = erzeugt eine Standard-Kopfzeile "Docklight Log File started..." nach dem Öffnen der Datei. Beim Schließen der Datei wird eine Fußzeile "Docklight Log File stopped" erzeugt. True = Keine zusätzlichen Kopf- oder Fußzeileninformationen erstellen.

Bemerkungen

Weitere Informationen zu den oben beschriebenen Funktionen und Argumenten von **StartLogging** finden Sie unter Protokollierung und Analyse eines Tests und im Dialogfeld Protokolldatei(en) erstellen.

Wenn *baseFilePath* eine leere Zeichenfolge ist, wird ein Dateidialog angezeigt, in dem der Pfad zur Protokolldatei und der Name der Basisdatei ausgewählt werden können.

Wenn **StartLogging** aufgerufen wird, während eine andere Protokolldatei noch von einem früheren StartLogging-Aufruf geöffnet ist, wird die Datei geschlossen und die neue Datei erstellt/geöffnet. Dadurch kann der Name der Protokolldatei geändert werden, ohne dass Daten verloren gehen.

Das noHeaders-Flag ist besonders nützlich, wenn Sie Protokolldaten ohne Zeitstempel erstellen wollen. Sie können dann das Ergebnis leicht mit früheren Testläufen vergleichen, indem Sie ein Dateivergleichstool verwenden.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiel

```
' Beispiel StartLogging

DL.ClearCommWindows
DL.StartLogging "C:\DocklightLogging"
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' - Öffnet vier Protokolldateien:
'  'C:\DocklightLogging_asc.txt'
'  'C:\DocklightLogging_hex.txt'
'  'C:\DocklightLogging_dec.txt'
'  'C:\DocklightLogging_bin.txt'
' 5 Sekunden lang warten
DL.Pause 5000
'  Schließen Sie die vier
Protokolldateien
DL.StopLogging
```

Beispiel 2

Das folgende Beispiel zeigt, wie man einen Datums-/Zeitstempel in den Namen der Protokolldatei einfügt und jede Stunde eine neue Protokolldatei erstellt

```
Beispiel 'Eine Protokolldatei pro Stunde'.

Dies ist der Basispfad und der Ort, an dem die
Protokolldatei(en) gespeichert werden sollen
Const BASE_FILE_PATH = "logfile_"
' ASCII- und HEX-Protokolldateien
erstellen Const
LOG_REPRESENTATIONS = "AH"

currentLogFileName =
""
DL.StartCommunication
Do
    newLogFileName = getFileName()
    ' Zeit für den Start einer
    neuen Datei?
    If newLogFileName <> currentLogFileName Then
        DL.StartLogging newLogFileName, True,
LOG_REPRESENTATIONS
     currentLogFileName = newLogFileName
    End If
    DL.Pause 1 ' CPU-Last
reduzieren Schleife

Function getFileName()
    dt = Jetzt
    ' Verfassen Sie einen Dateinamen.
    ' Die Right()-Funktionen stellen sicher, dass alle
    Monate, Tage, ' Stunden mit zwei Nachkommastellen
    gedruckt werden
    getFileName = BASE_FILE_PATH & Year(dt) & "_" & Right("0" &
Month(dt), 2) & "_" & Right("0" & Day(dt), 2) & "_" & Right("0"
& Hour(dt), 2) & "H"
Funktion beenden
```

10.2.1.13 StopLogging

Stoppt die Protokollierung und schließt die aktuell geöffnete(n) Protokolldatei(en). Dies entspricht dem Docklight-Menü **Kommunikationsprotokollierung beenden**.

Rückgabewert

Leere

Syntax

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

DL.StopLogging

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Bemerkungen

Weitere Informationen zu Protokolldateien finden Sie in der Methode StartLogging.

10.2.1.14 WaitForSequence

Wartet auf ein oder mehrere Ereignisse einer Receive Sequence und gibt den entsprechenden Zählerwert zurück (siehe GetReceiveCounter). Startet die Kommunikation, sofern sie nicht bereits läuft (siehe StartCommunication).

Rückgabewert

Lang

Syntax

Ergebnis **= DL.WaitForSequence(** *nameOrIndex* **[,** *maxCounter***] [,** *timeout***])**

Die Syntax der WaitForSequence-Methode besteht aus diesen Teilen:

Teil	Beschreibung
nameOrIndex	Erforderlich. Zeichenfolge, die den Namen der zu zählenden Empfangsfolge enthält. Die erste Empfangsfolge aus der Liste mit einem Namen, der mit *nameOrIndex* übereinstimmt, wird verwendet. Alternativ können Sie einen Ganzzahlwert übergeben, der den Sequenzindex angibt. Gültiger Sequenzindex liegt im Bereich von 0 bis (NoOfReceiveSequences - 1).
maxCounter	Optional. Lange Zahl, die den Grenzwert des Zählers enthält, bis die Funktion zurückkehrt. Standardwert ist 1 (eins): **WaitForSequence** kehrt zurück, nachdem das erste Auftreten der Empfangssequenz festgestellt wurde. Der Rückgabewert ist in diesem Fall 1. Wenn *maxCounter* -1 ist, verwendet **WaitForSequence** keine Zählerbegrenzung. Es kehrt erst nach einem Timeout zurück (siehe unten). Verwenden Sie *maxCounter* = -1, um alle Vorkommnisse einer Empfangssequenz innerhalb eines begrenzten Zeitraums zu zählen der Zeit.
Timeout	Optional. Lange Zahl, die eine zusätzliche Zeitüberschreitung in Millisekunden angibt. Standardwert ist -1 (keine Zeitüberschreitung). Der Höchstwert ist 86000000 (23,88 Stunden).

Bemerkungen

Die Methode **WaitForSequence** prüft die Anzahl der "Treffer" für diese Empfangssequenz, seit die Kommunikation gestartet wurde (siehe StartCommunication) oder der Zähler zurückgesetzt wurde (siehe ResetReceiveCounter). **WaitForSequence** wartet, bis die durch den *maxCounter* angegebene Anzahl von "Treffern" festgestellt wurde.

Eine grundlegende Anwendung für **WaitForSequence** ist das Warten auf eine bestimmte Antwort, nachdem ein Testbefehl an Ihr serielles Gerät gesendet wurde. Um sicherzustellen, dass Sie eine sehr schnelle Antwort von Ihrem Gerät nicht verpassen, verwenden Sie die folgende Befehlsreihenfolge:
1. Setzen Sie den/die Zähler zunächst mit ResetReceiveCounter zurück.
2. Senden Sie Ihren Testbefehl mit SendSequence
3. Verwenden Sie nun **WaitForSequence**, um auf die erwartete Antwort zu warten

Es ist sehr wichtig, dass Sie **ResetReceiveCounter** vor **SendSequence** verwenden**.** **ResetReceiveCounter** setzt nicht nur den Erkennungszähler auf Null, sondern setzt auch den Zeichenabgleich zurück, so dass alle zuvor empfangenen Zeichen bei der Suche nach einer Sequenzübereinstimmung nicht berücksichtigt werden. Siehe auch

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

die Anmerkungen zur Wildcard-Suche für zusätzliche Informationen darüber, wie Docklight den Mustervergleich von Receive Sequence handhabt.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Während einer **WaitForSequence** können keine DL_OnReceive()-Aufrufe verarbeitet werden. Wenn Sie während des Wartens **DL_OnReceive()**-Ereignisse verarbeiten müssen, sehen Sie sich den **pauseWithEvents()**-Code an, der unter OnReceive Beispiel 2 beschrieben ist.

Wenn Sie warten müssen, bis *eine* der Empfangssequenzen ausgelöst wird, bietet die Prozedur DL_OnReceive() die Lösung. Siehe das OnReceive Beispiel 3.

Beispiel

```
' Beispiel WaitForSequence

' Zählen Sie die Anzahl der Vorkommen von
' die erste Empfangssequenz innerhalb eines 10-
Sekunden ' Intervalls.
Erfordert mindestens eine Definition einer Empfangssequenz

DL.StartCommunication
DL.ClearCommWindows
Ergebnis = DL.WaitForSequence(0 , -1, 10000)
DL.AddComment vbCrLf & vbCrLf & "Empfangssequenz #0, Trefferzahl
= " & Ergebnis
' Alternative Möglichkeit, den Zähler anschließend
auszulesen DL.AddComment "Receive Sequence #0, hit count = "
& DL.GetReceiveCounter(0)

' Senden Sie die erste Sendesequenz und warten Sie auf eine
Antwort des Geräts (kein Timeout)
DL.AddComment vbCrLf & vbCrLf & "Sending data and waiting for
Receive Sequence #0"
DL.ResetReceiveCounter
DL.SendSequence 0
DL.WaitForSequence 0
```

10.2.2 Methoden (Fortgeschrittene)

10.2.2.1 CalcChecksum

Gibt eine Prüfsumme oder einen CRC-Wert für eine gegebene Sequenz oder einen Teil einer Sequenz zurück.

Die CalcChecksum-Methode ist eine fortgeschrittene Docklight-Scripting-Funktion und erfordert einige Kenntnisse über Prüfsummen in seriellen Anwendungsprotokollen und darüber, wie Docklight mit Sende-/Empfangsdaten im Allgemeinen umgeht.

TIPP: Wir empfehlen den Abschnitt Berechnung und Validierung von Prüfsummen als Einführung. Wenn Ihnen die CRC-spezifischen Begriffe und Parameter verwirrend erscheinen, finden Sie im CRC-Glossar einige Hintergrundinformationen.

Rückgabewe

rt String

Syntax

result **= DL.CalcChecksum(** *checksumSpec*, *dataStr*, **[**, *representation***] [**, *startPos***]** **[**, *endPos***] [**, *bigEndian* **])**

Die Syntax der CalcChecksum-Methode besteht aus folgenden Teilen:

Teil	Beschreibung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

checksumSpec	Erforderlich. Zeichenkette, die den Prüfsummenalgorithmus und seine Parameter angibt. **CalcChecksum** unterstützt vordefinierte Namen für gängige Prüfsummenalgorithmen, oder Sie können eine generische CRC-Spezifikation für die Berechnung exotischerer CRCs übergeben. Vordefinierte Namen sind: "MOD256", "XOR", "CRC-8", "CRC-CCITT", "CRC-16", "CRC-MODBUS" und "CRC-32" Siehe checksumSpec Format für die vollständige Formatspezifikation.
dataStr	Erforderlich. Stringwert, der die Eingabesequenz für die Prüfsummenberechnung enthält, wie sie z. B. von der Funktion OnSend_GetData() zurückgegeben wird.
Vertretung	Optional. String-Wert zur Festlegung des Formats der *dataStr* Sequence: "H" = Hex (Standard), "A" = ASCII , "D" = Dezimal oder "B" = Binär.
startPos	Optional Ganzzahliger Wert. Gibt die Zeichenposition an, an der die Berechnung beginnen soll. Standardwert ist 1 (Beginn der *dataStr* Sequence). *startPos* akzeptiert auch negative Werte, z.B. -1 für "letztes Zeichen", -2 für "2. Zeichen vom Ende", -3 für "3. Zeichen vom Ende".
endPos	Optional Ganzzahliger Wert. Gibt das letzte Zeichen an, das in die Berechnung einbezogen werden soll. Standardwert ist die Größe der dataStr Sequence. *endPos* akzeptiert auch negative Werte, siehe *startPos* oben.
bigEndian	Optional. Boolescher Wert zur Festlegung der Bytereihenfolge für das *Ergebnis*. True (Standard): Big-Endian-Byte-Reihenfolge verwenden (erstes Zeichen ist höchstwertig) False: Little-Endian-Byte-Reihenfolge verwenden (erstes Zeichen ist niederwertig)

Bemerkungen

Der Rückgabewert ist eine Zeichenkette mit der CRC/Prüfsumme im Docklight-HEX-Sequenzformat, z. B. "CB F4 39 26". Die Anzahl der zurückgegebenen HEX-Bytes hängt von der Breite des Prüfsummenalgorithmus ab. Siehe das Beispielskript und die Ausgabe im Kommunikationsfenster unten.

Mit Hilfe von **CalcChecksum** können Sie spezifische Prüfsummen für Sendesequenzen on the fly generieren oder erweiterte Prüfsummenvalidierungen für empfangene Daten verwenden. Siehe die Sub DL_OnSend() Event Prozedur für Details.

Standard-Prüfsummen können bereits ohne Skriptcode verarbeitet werden, indem der Prüfsummenteil der Dialoge Sendesequenz bearbeiten / Empfangssequenz bearbeiten verwendet wird. Siehe auch das zugehörige Beispiel für das Modbus-Protokoll.

Beispiel

```
' Beispiel CalcChecksum

DL.ClearCommWindows

DL.AddComment
DL.AddComment "Einfache Prüfsumme (Mod 256) für '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("MOD256",
"123456789", "A")
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.AddComment
DL.AddComment "8-Bit-CRC (CRC-DOW) für '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-DOW",
"123456789", "A")
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.AddComment
DL.AddComment "16 Bit CRC (CRC-16) für '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-16",
"123456789", "A")

DL.AddComment
DL.AddComment "16 Bit CRC (CRC-Modbus) für '123456789' in
'LittleEndian' - unteres Byte zuerst"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-Modbus",
"123456789", "A", , ,False)

DL.AddComment
DL.AddComment "16 Bit CRC (CRC-CCITT) für '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-CCITT",
"123456789", "A")
DL.AddComment "Jetzt machen Sie dasselbe, aber geben Sie alle
CRC-Details selbst an..."
DL.AddComment "Prüfsumme = " &
DL.CalcChecksum("CRC:16,1021,FFFF,0000,No,No", "123456789",
"A")

DL.AddComment
DL.AddComment "32 Bit CRC (CRC-32) für '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-32",
"123456789", "A")

DL.AddComment
DL.AddComment "Ein 32-Bit-CRC (CRC-32) auf die ersten 5
Bytes der HEX-Sequenz 01 02 03 04 05 FF FF FF, Ergebnis ist
Little Endian / niedrigstes Byte zuerst"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-32", "01
02 03 04 05", "H", 1, 5, False)
```

Der obige Skriptcode erzeugt die folgende Ausgabe im Docklight-Kommunikationsfenster:

```
 Einfache Prüfsumme (Mod 256) für
 '123456789' CalcChecksum = DD

 8-Bit-CRC (CRC DOW) für '123456789'.
 CalcChecksum = A1

 16 Bit CRC (CRC-16) für '123456789'.
 Prüfsumme = BB 3D

 16 Bit CRC (CRC-Modbus) für '123456789' in 'LittleEndian' -
unteres Byte zuerst
 CalcChecksum = 37 4B

 16 Bit CRC (CRC-CCITT) für '123456789'.
 CalcChecksum = 29 B1
 Nun machen Sie dasselbe, geben aber alle CRC-Details selbst
 an... CRC-Prüfsumme = 29 B1

 32 Bit CRC (CRC-32) für '123456789'.
 CalcChecksum = CB F4 39 26
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
 Ein 32-Bit-CRC (CRC-32) auf die ersten 5 Bytes der HEX-
Sequenz 01 02 03 04 05 FF FF FF, Ergebnis ist Little Endian /
niedrigstes Byte zuerst
 Prüfsumme = F4 99 0B 47
```

10.2.2.2 ConvertSequenceData

Konvertiert Sequenzdaten in/aus einer Fließkommazahl, einer Ganzzahl oder anderen in technischen Anwendungen üblichen Datentypen.

Rückgabewe

rt String

Syntax

result **= DL.ConvertSequenceData(** *conversionType*, *source*, **[**, *representation***] [**, *bigEndian* **])**

Die Syntax der ConvertSequenceData-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
conversionType	Erforderlich. Zeichenfolge, die den Konvertierungstyp und die Richtung angibt. Siehe unten für die Liste der Konvertierungen und Beispiele.
Quelle	Erforderlich. Eingangsdaten-String für die Konvertierung. Dies kann eine Docklight-Sequenz sein, z. B. "4B 06 9E 3F", oder eine Zeichenkette mit dem Anwendungswert, z. B. "1,234567". Siehe unten für Details.
Vertretung	Optional. Format der Sequenzzeichenfolge (entweder *Quelle* oder *Ergebnis*, je nach *conversionType*): "H" = Hex (Standard), "D" = Dezimal oder "B" = Binär.
bigEndian	Optional. Boolescher Wert zur Festlegung der Bytereihenfolge bei Ganzzahl- oder Fließkomma-Umwandlungen. True (Standard): Big-Endian-Byte-Reihenfolge verwenden (erstes Zeichen ist höchstwertig) False: Little-Endian-Byte-Reihenfolge verwenden (erstes Zeichen ist niederwertig)

Das Argument *conversionType* unterstützt die folgenden Werte und Arten von Konvertierungen:

Wert	Beschreibung
"toSingle"	Konvertiert die *Quelle* in eine Fließkommazahl mit einfacher Genauigkeit. *Quelle*: IEEE-Sequenz mit einfacher Genauigkeit (32 Bit) *Ergebnis*: String mit Fließkommazahl im nicht lokalisierten Format, verwendet Punkt (".") als Dezimaltrennzeichen. Beispiel: `DL.ConvertSequenceData("toSingle", "3F 9E 06 4B")` Ertrag: `1,234567`
"fromSingle"	Konvertierung der *Quelle* in eine IEEE-Sequenz mit einfacher Genauigkeit (32 Bit) *Quelle*: String mit Fließkommazahl. Sowohl Punkt (".") als auch Komma (",") werden als Dezimaltrennzeichen akzeptiert. *Ergebnis*: 32-Bit-Sequenzdaten Beispiel: `DL.ConvertSequenceData("fromSingle", "1.234567")` zurück: `3F 9E 06 4B`

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

"toDouble"	Konvertiert die *Quelle* in eine Fließkommazahl mit doppelter Genauigkeit. *Quelle*: IEEE-Sequenz mit doppelter Genauigkeit (64 Bit) *Ergebnis*: String mit Fließkommazahl in nicht lokalisiertem Format (siehe oben) Beispiel:

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	`DL.ConvertSequenceData("toDouble", "103 154 149 160 081 161 036 075", "D", False)` Ertrag: `9,87987987987E+53`
"fromDouble"	Umwandlung in eine IEEE-Doppelpräzisionssequenz (64 Bit) *Quelle*: String mit Fließkommazahl. Sowohl Punkt (".") als auch Komma (",") werden als Dezimaltrennzeichen akzeptiert. *Ergebnis*: 64-Bit-Sequenzdaten Beispiel: `DL.ConvertSequenceData("fromDouble", "9.87987987987E+53", "D", False)` Erträge: `103 154 149 160 081 161 036 075`

"vonText"	Konvertiert einen Klartext in eine Hex-, Dezimal- oder Binärsequenz. z.B.
	`DL.AddComment DL.ConvertSequenceData("fromText", "Hallo Welt")`
	Rückgaben: `48 65 6C 6C 6F 20 57 6F 72 6C 64`
	
	bigEndian = false: Wenn diese Option für "fromText" verwendet wird, ist die resultierende Sequenz ohne Trennzeichen, z. B.
	`48656C6C6F20576F726C64`

"bisGanzzahl16"
"vonGanzzahl16"
	Konvertierung in/aus einem vorzeichenbehafteten 16-Bit-Ganzzahlwert Beispiele:
	`DL.ConvertSequenceData("toInteger16", "80 00")`
	Rückgabe: `-32768`
	`DL.ConvertSequenceData("fromInteger16", "-1")`
	zurück: `FF FF`

"toUnsigned16"
"fromUnsigned16"
	Wie "toInteger16" / "fromInteger16", aber für vorzeichenlose 16-Bit-Ganzzahldaten
	Beispiele:
	`DL.ConvertSequenceData("toUnsigned16", "80 00")`
	zurück: `32768`
	`DL.ConvertSequenceData("fromUnsigned16", "65535", "D")`
	Rückgabe: `255 255`

"toInteger32"
"fromInteger32"
	Konvertierung in/aus einem vorzeichenbehafteten 32-Bit-Ganzzahlwert Beispiele:
	`DL.ConvertSequenceData("toInteger32", "00 00 00 80", "H", False)`
	Rückgabe: `-2147483648`
	`DL.ConvertSequenceData("fromInteger32", "-2", "H", False)`
	zurück: `FE FF FF FF`

"toUnsigned32"
"fromUnsigned32"
	Wie "toInteger32" / "fromInteger32", aber für vorzeichenlose 32-Bit-Ganzzahldaten
	Beispiele:
	`DL.ConvertSequenceData("toUnsigned32", "FF 00 FF 00")`
	zurück: `4278255360`
	`DL.ConvertSequenceData("fromUnsigned32", "21121977", "D")`
	Rückgabe: `001 066 075 185`

"toBool"	Gibt "True" zurück, wenn das erste Quellzeichen <> 0 ist Beispiel:
	`DL.ConvertSequenceData("toBool", "00")`
	gibt zurück: `False`
	`DL.ConvertSequenceData("toBool", "01 00")`
	zurück: `True`

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

"bisText"	Konvertiert Sequenzdaten in eine Textzeichenfolge, die nur Druckzeichen enthält (siehe ASCII-Zeichensatz). Die ASCII-Codes 0 - 31 und 127 - 255 werden herausgefiltert und erscheinen nicht im Ergebnis. *Quelle*: Sequenz mit den Originaldaten, einschließlich nicht druckbarer Zeichencodes *Ergebnis*: der ASCII-Text, der nur die ASCII-Codes 32 - 126 verwendet Beispiel: `DL.ConvertSequenceData("toText", "FF 48 65 6C 6C 6F 21 0D 00 00")` zurück: `Hallo!`

Bemerkungen

Überprüfen Sie sorgfältig Ihre Protokollspezifikation zum Datenformat, einschließlich der Endianness (little endian / big endian).

Wenn Sie das Ergebnis einer "toSingle"- oder "toDouble"-Umwandlung für weitere Berechnungen verwenden, denken Sie daran, dass *das Ergebnis* eine nicht-numerische Zeichenfolge wie "NaN" (keine Zahl) oder "Inf" (Unendlich) sein kann.

Beachten Sie, dass "toText" nicht dasselbe ist wie das Auslesen einer Datenfolge in ASCII-Darstellung ("A"). Beispiel:

```
DL.AddComment DL.OnSend_GetData("A")
DL.AddComment DL.ConvertSequenceData("toText",
DL.OnSend_GetData("H"))
```

könnte das Folgende zurückgeben:

```
-Hallo!<CR><NUL><NUL>
Hallo!
```

10.2.2.3 GetChannelSettings

Gibt die aktuellen Einstellungen des Kommunikationskanals zurück (COM-Anschlussnummer oder TCP-Adresse, Einstellungen des seriellen Anschlusses).

HINWEIS: **GetChannelSettings** ist eine Ergänzung zur Methode SetChannelSettings und für fortgeschrittene Docklight-Scripting-Anwendungen gedacht, bei denen eine Kontrolle der Kommunikationskanaleinstellungen erforderlich ist.

Rückgabewe

rt String

Syntax

Ergebnis **= DL.GetChannelSettings([***channelNo***])**

Die Syntax der GetChannelSettings-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
channelNo	Optional. Ganzzahl, die den Kommunikationskanal angibt, wenn Kommunikationsmodus: Überwachung verwendet wird. Standardwert ist 1 (Kanal 1).

Bemerkungen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

GetChannelSettings gibt eine Zeichenfolge mit den aktuellen seriellen oder TCP-Einstellungen für den angegebenen Kommunikationskanal zurück.

Wenn es sich bei dem Kanal um eine serielle Schnittstelle handelt, hat der Rückgabewert folgendes Format: *COMxxx*: *BaudRate*, *Parität*, *DataBits*, *StopBits*, *FlowControl*, *ParityErrorChar*
e.g. "COM1:9600,NONE,8,1,OFF,63"

Handelt es sich bei dem Kanal um einen TCP-Client, ist der Rückgabewert die aktuelle IP-Adresse und TCP-Portnummer, z. B. "192.0.0.1:3001".

Handelt es sich bei dem Kanal um einen TCP-Server, ist der Rückgabewert die Zeichenkette "SERVER:" plus die TCP-Portnummer, z. B. "SERVER:3001".

Siehe auch die Methode SetChannelSettings für einen detaillierten Überblick über das Datenformat des Rückgabewerts und ein komplexeres Beispiel für die Manipulation von Kanaleinstellungen während der Skriptlaufzeit.

Beispiel

```
' Beispiel GetChannelSettings

DL.AddComment "Comm. Kanal 1 Einstellungen = " &
DL.GetChannelSettings()
Der folgende Befehl wird nur funktionieren,
' wenn der Docklight-Kommunikationsmodus 'Überwachung (nur
Empfang)' ist
DL.AddComment "Comm. Kanal 2 Einstellungen = " &
DL.GetChannelSettings(2)
```

Das Beispiel könnte die folgende Ausgabe im Docklight-Kommunikationsfenster erzeugen:

```
Komm. Kanal 1 Einstellungen =

COM1:9600,NONE,8,1,OFF,63 Komm. Kanal 2 Einstellungen

= SERVER:10001
```

10.2.2.4 GetChannelStatus

Gibt den aktuellen Status des Kommunikationskanals zurück (geschlossen, offen, wartet auf TCP-Verbindung oder Fehler).

Rückgabewe

rt Integer

Syntax

Ergebnis **= DL.GetChannelStatus([***channelNo***])**

Die Syntax der GetChannelStatus-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
channelNo	Optional. Ganzzahl, die den Kommunikationskanal angibt, wenn Kommunikationsmodus: Überwachung oder Seitenkanäle verwendet werden. Standardwert ist 1 (Kanal 1).

Bemerkungen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

GetChannelStatus gibt die folgenden Werte zurück:

Ergebnis	**Beschreibung**
0	Kanal wird geschlossen, Kommunikation wird gestoppt (siehe auch StopCommunication)
1	Der Kanal ist offen und bereit zum Senden/Empfangen von Daten. TCP-Server- oder TCP-Client-Modus: Verbindung hergestellt.
2	TCP-Server- oder TCP-Client-Modus: Wartet auf Verbindung. COM-Anschluss mit RTS/CTS-Hardware-Flusskontrolle: Warten auf Handshake-Signal.
3	Kanalfehler, z. B. nach einem SetChannelSettings-Befehl, der eine nicht existierende COM-Port-Nummer angegeben.

Siehe auch SetChannelSettings und GetChannelSettings.

Beispiel

```
' Beispiel GetChannelStatus
' (erfordert Docklight im Sende-/Empfangsmodus)

DL.ClearCommWindows

DL.AddComment "Zugriff auf COM-Anschluss"
DL.SetChannelSettings "COM3:9600,NONE,8,1", 1
DL.AddComment "GetChannelStatus vor StartCommunication = " &
DL.GetChannelStatus(1)
DL.StartCommunication
DL.AddComment "GetChannelStatus nach StartCommunication = " &
DL.GetChannelStatus(1)
DL.StopCommunication

DL.AddComment
DL.AddComment "TCP-Client-Modus"
DL.AddComment "Verbinden mit docklight.de ..."
DL.SetChannelSettings "docklight.de:80", 1
DL.StartCommunication
' warten bis verbunden
Do
    commStatus = DL.GetChannelStatus(1)
    DL.AddComment "GetChannelStatus = " & commStatus
    DL.Pause 10
Schleife bis commStatus <>
2 If commStatus = 1 Then
    DL.AddComment "Verbunden."
Else
    DL.AddComment
"Fehler!" End If
DL.StopCommunication
```

Nach dem Ausführen des Skripts auf einem Computer mit einem eingebauten COM3-Anschluss (z. B. Modem) und einer Internetverbindung könnte das Kommunikationsfenster wie folgt aussehen:

```
COM-Port-Zugang
GetChannelStatus vor StartCommunication = 0
GetChannelStatus nach StartCommunication = 1

TCP-Client-Modus
Verbinden mit docklight.de ...
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
GetChannelStatus = 2
GetChannelStatus = 2
GetChannelStatus = 2
GetChannelStatus = 2
GetChannelStatus = 1
Verbunden.
```

10.2.2.5 GetCommWindowData

Gibt den kumulierten Inhalt des Kommunikationsfensterpuffers zurück.

HINWEIS: Diese Methode ist für spezielle Anwendungen gedacht. Für viele Standardanwendungsfälle sind die Ereignisprozeduren OnSend / OnReceive oder die Methode GetReceiveComments die bevorzugte Lösung.

Rückgabewe

rt String

Syntax

Ergebnis = **DL.GetCommWindowData([***Darstellung***])**

Die Syntax der Methode **GetCommWindowData** besteht aus den folgenden Teilen:

Teil	Beschreibung
Vertretung	Erforderlich. String-Wert zur Definition des gewünschten Fensterpufferformats: "A" = ASCII (Standard), "H" = Hex, "D" = Dezimal oder "B" = Binär.

Bemerkungen

Es kann nur eine *Darstellung* verwendet werden, die in Docklight-Optionen - Kommunikationsfenster-Modi aktiviert ist. Standardmäßig sind dies ASCII, HEX und Dezimal. Laden Sie bei Bedarf andere Optionen mit LoadProgramOptions**.**

Die maximale Größe des GetCommWindowData-Puffers beträgt 128000 Zeichen. Wenn mehr Kommunikationsdaten anfallen, ohne dass **GetCommWindowData** aufgerufen wird, werden die ältesten Daten gelöscht.

10.2.2.6 GetEnvironment

Liefert den Wert einer Windows-Umgebungsvariablen im aktuellen Benutzerprofil oder den Wert einer der unten beschriebenen Docklight-spezifischen Umgebungsvariablen.

Rückgabewe

rt String

Syntax

Ergebnis **= DL.GetEnvironment(** *Name* **)**

Die Syntax der GetEnvironment-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
Name	Erforderlich. *Name* sein kann: 1) Der Name der Windows-Umgebungsvariable. (Ohne die %-Zeichen, die in der Windows-Befehlsshell verwendet werden **cmd.exe**).

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	2) Einer der unten aufgeführten Docklight-spezifischen Namen

Docklight Scripting Umgebungsvariablen

Name	Beschreibung
`DOCKLIGHT_VERSION`	Version der Docklight Scripting-Anwendung
`DOCKLIGHT_SCRIPTDIR`	den Ordner, in dem das Skript ausgeführt wird
`DOCKLIGHT_DIALOGDIR`	den Ordner, der für das zuletzt verwendete Skriptdateidialogfeld verwendet wurde
`DOCKLIGHT_PORTLIST`	Liste der auf diesem PC verfügbaren COM-Ports
`DOCKLIGHT_SENDSEQ`	Liste aller Sendesequenznamen im aktuellen Docklight-Projekt (.ptp-Datei)
`DOCKLIGHT_RECEIVESEQ`	Liste aller Namen von Empfangssequenzen
`DOCKLIGHT_SENDSEQDEF`	Liste aller Sendesequenzen **Name** und **Sequenz** im HEX-Format. **Name** und **Sequenz** werden in getrennten Textzeilen zurückgegeben
`DOCKLIGHT_SENDSEQDEF:` *SequenzName*	Listet die Definition nur für die Sequenznamen auf, die mit *SequenceName* übereinstimmen. *SequenceName* kann Wildcards enthalten, z.B. können Sie verwenden: `DOCKLIGHT_SENDSEQDEF:Test*`
`DOCKLIGHT_RECEIVESEQDEF`	wie `DOCKLIGHT_SENDSEQDEF`, aber für Receive Sequenzen

Bemerkungen

Die Liste der im folgenden Beispiel verwendeten Umgebungsvariablen ist nur ein

Beispiel. TIPP: Eine Liste der Variablen, die in Ihrem aktuellen Benutzerprofil

verfügbar sind, erhalten Sie, wenn Sie ein *Windows* Befehlsprozessor-Fenster (Windows-Taste + R, dann **cmd** eingeben), dann **SET** eingeben und drücken Sie Enter.

HINWEIS: In Docklight Scripting V2.0 und früher hieß diese Methode **GetEnvironmentVariable**. Der alte Name wird aus Kompatibilitätsgründen noch unterstützt. Er wurde geändert, um Verwechslungen mit der gleichnamigen Windows-API-Funktion zu vermeiden.

Beispiel

```
' Beispiel

GetEnvironment nameList

=
"ALLUSERSPROFILE,APPDATA,COMPUTERNAME,HOMEDRIVE,HOMEPATH,LOCALA
PPDATA,LOGONSERVER,ANZAHL_DER_PROZESSOREN,OS,PROZESSOR_ARCHITEKT
UR,PROZESSOR_IDENTIFIKATOR,PROZESSOR_LEVEL,PROZESSOR_REVISION,PU
BL IC,TEMP,TMP,USERDOMAIN,USERNAME,USERPROFILE"

DL.AddComment "Docklight Scripting läuft " &
DL.GetEnvironment("DOCKLIGHT_VERSION")
nameArray = Split(nameList , ",")
For i = 0 To UBound(nameArray)
   name = nameArray(i)
   DL.AddComment name & " = " & DL.GetEnvironment(name)
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
Next
```

Auf einem *Windows* 10-Notebook könnte die Ausgabe des Kommunikationsfensters wie folgt aussehen:

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
Ausführen von Docklight Scripting Docklight Scripting V2.4.5
ALLUSERSPROFILE = C:\ProgramData
APPDATA = C:\Benutzer\Docklight\AppData\Roaming
COMPUTERNAME = DOCK-OH
HOMEDRIVE = C:
HOMEPATH = \Benutzer\Docklight
LOCALAPPDATA =
C:\Benutzer\Docklight\AppData\Local LOGONSERVER
= \\DOCK-OH
ANZAHL_DER_PROZESSOREN = 4
OS = Windows_NT
PROCESSOR_ARCHITECTURE = x86
PROZESSOR_IDENTIFIER = Intel64 Familie 6 Modell 78 Stepping 3,
GenuineIntel
PROZESSOR_LEVEL = 6
PROZESSOR_REVISION = 4e03
PUBLIC = C:\Benutzer\Public
TEMP = C:\Users\docklight\AppData\Local\Temp
TMP = C:\Users\docklight\AppData\Local\Temp
USERDOMAIN = DOCK-OH
USERNAME = docklight
BENUTZERPROFIL =
C:\Benutzer\docklight
```

10.2.2.7 GetHandshakeSignals

Gibt die aktuellen Handshake-Signalzustände (CTS, DSR, DCD, RI) als Integer-Bitwert zurück, so wie es auch das Empfangssequenz-Funktionszeichen '!' tut.

Rückgabewe

rt Integer

Syntax

Ergebnis **= DL.GetHandshakeSignals()**

Bemerkungen

Das Ergebnis ist ein Bit-Wert mit den folgenden Komponenten:

Bit-Nr.	Dezimaler Wert	Handshake-Signal
0	001	CTS = Hoch
1	002	DSR = Hoch
2	004	DCD = Hoch
3	008	RI (Ring Indicator) = Hoch

In Tap Pro / Tap 485-Anwendungen gibt **GetHandshakeSignals** den folgenden erweiterten Satz von Handshake-Signalzuständen zurück:

Bit-Nr.	Dezimaler Wert	Handshake-Signal
0	001	CTS = High (DCE-Seite / Docklight-Empfangskanal 2)
1	002	DSR = Hoch (DCE-Seite / Kanal 2)
2	004	DCD = High (DCE-Seite / Kanal 2)
3	008	RI (Ring Indicator) = Hoch (DCE-Seite / Kanal 2)
4	016	RTS = High (DEE-Seite / Kanal 1)
5	032	DTR = High (DEE-Seite / Kanal 1)

Siehe auch SetHandshakeSignals zur Steuerung des Zustands der RTS- und DTR-Leitungen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiel

```
' Beispiel GetHandshakeSignals
DL.StartCommunication
Do
    DL.AddComment DL.GetDocklightTimeStamp() & " -
GetHandshakeSignals() = " & DL.GetHandshakeSignals()
    DL.Pause 200
Schleife
```

Beispiel für die Ausgabe im Kommunikationsfenster:

```
 6/23/2012 10:07:44.244 - GetHandshakeSignals() = 0
 6/23/2012 10:07:44.469 - GetHandshakeSignals() = 48
 6/23/2012 10:07:44.677 - GetHandshakeSignals() = 48
 6/23/2012 10:07:44.884 - GetHandshakeSignals() = 48
```

HINWEIS: Es kann 5-10 Millisekunden nach **StartCommunication** dauern, bis **GetHandshakeSignals** meldet den korrekten Signalzustand.

10.2.2.8 GetKeyState

Gibt den Zustand einer bestimmten Tastaturtaste, Maustaste oder einer ähnlichen Eingabe zurück.

Syntax

Ergebnis **= DL.GetKeyState(** *virtKey* **)**

Die Syntax der GetKeyState-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
virtKey	Erforderlicher ganzzahliger Wert. Ein ASCII- oder virtueller Schlüsselcode. Handelt es sich bei dem angeforderten Schlüssel um eine Ziffer von 0-9 oder einen Buchstaben von A-Z, wird der entsprechende ASCII-Code verwendet. Weitere Tasten und Schaltflächen finden Sie in der Tabelle mit **den Codes für virtuelle Tasten** in den *Microsoft Windows* Docs.

Bemerkungen

Das Argument **GetKeyState** und der Rückgabewert entsprechen der Windows-Funktion GetKeyState aus der Bibliothek **Winuser.h**.

Für virtuelle Tasten (z.B. F1-F12, Multimedia-Tasten, Maustasten usw.) können Sie die VBScript-Deklarationen in der Datei **GetKeyState_ImportFile_VirtualKeys.pts** aus dem Ordner ScriptSamples\Extras\GetKeyState_Example verwenden. Wir empfehlen, die Datei in Ihren eigenen Skriptordner zu kopieren und sie mit der #include-Direktive einzubinden.

Beispiel

```
' Beispiel GetKeyState

' Zwei virtuelle Schlüsseldeklarationen,
entnommen aus
GetKeyState_ImportFile_VirtualKeys.pts
Const VK_F2=     &h71
Konst. VK_LBUTTON= &h01
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Do

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
      DL.AddComment "Beispiel für die Funktionstaste F2:
DL.GetKeyState(VK_F2)=" & DL.GetKeyState(VK_F2)
      If keyDown(Asc("D")) Then
            DL.AddComment "Taste D ist
            gedrückt"
      Ende wenn
      Wenn TasteAb(VK_F2) Dann
            DL.AddComment "F2-Taste ist
      gedrückt" End If
      If keyDown(VK_LBUTTON) Then
            DL.AddComment "Linke Maustaste ist gedrückt"
      End If
      DL.Pause 500
Schleife

Function keyDown(virtKey)
      keyDown = (DL.GetKeyState(virtKey) And 128) > 0
Funktion beenden
```

Beispiel für die Ausgabe im Kommunikationsfenster:

```
Beispiel für die Funktionstaste F2:
DL.GetKeyState(VK_F2)=1 Beispiel für die
Funktionstaste F2: DL.GetKeyState(VK_F2)=1 Beispiel
für die Funktionstaste F2: DL.GetKeyState(VK_F2)=1
Linke Maustaste ist gedrückt
Beispiel für die Funktionstaste F2:
DL.GetKeyState(VK_F2)=1 Linke Maustaste ist gedrückt
Beispiel für die Funktionstaste F2:
DL.GetKeyState(VK_F2)=1 D Taste ist gedrückt
Beispiel für die Funktionstaste F2:
DL.GetKeyState(VK_F2)=1 Beispiel für die Funktionstaste
F2: DL.GetKeyState(VK_F2)=-128 F2-Taste ist gedrückt
Beispiel für die Funktionstaste F2:
DL.GetKeyState(VK_F2)=0 Beispiel für die
Funktionstaste F2: DL.GetKeyState(VK_F2)=0 Beispiel
für die Funktionstaste F2: DL.GetKeyState(VK_F2)=0
```

10.2.2.9 GetReceiveComments

Gibt eine chronologische Liste aller ausgegebenen Empfangsreihenfolgekommentare zurück, als Alternative zur Verarbeitung von Sub DL_OnReceive().

Rückgabewe

rt String

Syntax

Ergebnis **= DL.GetReceiveComments()**

Bemerkungen

result enthält alle Receive Sequence Comments in chronologischer Reihenfolge, getrennt durch einen Zeilenumbruch, seit dem letzten Aufruf von **GetReceiveComments**. Mit Hilfe von Receive Sequence Comment Makros können Sie einen Parser für alle eingehenden Receive Sequence Daten implementieren, als Alternative zu Sub DL_OnReceive().

HINWEIS: Es werden maximal 10000 Empfangssequenzereignisse gespeichert und von

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

GetReceiveComments, was für alle praktischen Anwendungen ausreichend sein sollte.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.2.2.10 Eingabefeld2

Alternative zur ursprünglichen VBScript InputBox-Methode.

Zeigt eine Eingabeaufforderung in einem Dialogfeld an, wartet darauf, dass der Benutzer Text eingibt oder auf eine Schaltfläche klickt, und gibt den Inhalt des Textfelds zurück. Dieser Dialog erscheint immer auf demselben Bildschirm wie das Docklight Scripting-Hauptfenster. Er unterstützt nicht die (selten nützlichen) optionalen Argumente *xpos*, *ypos*, *helpfile* und *context* der VBScript-Methode InputBox.

TIPP: Als Alternative können Sie auch die Funktion DL.GetKeyState verwenden, um auf Tastatur- oder Mauseingaben zu warten und zu reagieren.

Rückgabewe

rt String

Syntax

Ergebnis **= DL.InputBox2 (***Eingabeaufforderung***[,** *Titel***][,** *Standard***])**

Teil	Beschreibung
Aufforderung	Erforderlich. String-Ausdruck, der als Meldung im Dialogfeld angezeigt wird. Die maximale Länge der Eingabeaufforderung beträgt ca. 1024 Zeichen, abhängig von der Breite der verwendeten Zeichen. Wenn die Eingabeaufforderung aus mehr als einer Zeile besteht, können Sie die Zeilen durch ein Wagenrücklaufzeichen (**Chr(**13**)**), ein Zeilenvorschubzeichen (**Chr(**10**)) oder eine** Kombination aus Wagenrücklauf- und Zeilenvorschubzeichen (**Chr(**13**) & Chr(**10**)**) zwischen jeder Zeile trennen.
Titel	Optional. String-Ausdruck, der in der Titelleiste des Dialogfelds angezeigt wird. Wenn Sie title weglassen, wird der Anwendungsname in der Titelleiste angezeigt.
Standard	Optional. String-Ausdruck, der im Textfeld als Standardantwort angezeigt wird, wenn keine anderen Eingaben gemacht werden. Wenn Sie default weglassen, wird das Textfeld leer angezeigt.

```
Beispiel DL.InputBox2 Funktion
MyInput = DL.InputBox2("Bitte Text eingeben", "Mein Titel",
  "Beispieltext")
DL.AddComment MeineEingabe      ' Druck der Benutzereingabe
```

10.2.2.11 MsgBox2

Alternative zur ursprünglichen VBScript-MSgBox-Methode.

Zeigt ein Meldungsfenster an, wartet darauf, dass der Benutzer eine Schaltfläche anklickt, und gibt einen Wert zurück, der angibt, welche Schaltfläche der Benutzer angeklickt hat. Dieser Dialog erscheint immer auf demselben Bildschirm wie das Docklight Scripting Hauptfenster. Er unterstützt nicht die optionalen Argumente *helpfile* und *context* der VBScript-Methode MsgBox.

TIPP: Alternativ können Sie auch die Funktion DL.SetUserOutput verwenden, um einen zusätzlichen Benutzerausgabebereich zu erstellen und zusätzliche Benutzerinformationen anzuzeigen.

Rückgabewe

rt Integer

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Syntax

Ergebnis **= DL.MsgBox2 (***Eingabeaufforderung***[,** *Schaltflächen***][,** *Titel***])**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Teil	Beschreibung
Aufforderung	Erforderlich. String-Ausdruck, der als Meldung im Dialogfeld angezeigt wird. Die maximale Länge der Eingabeaufforderung beträgt ca. 1024 Zeichen, abhängig von der Breite der verwendeten Zeichen. Wenn die Eingabeaufforderung aus mehr als einer Zeile besteht, können Sie die Zeilen durch ein Wagenrücklaufzeichen (**Chr(**13**)**), ein Zeilenvorschubzeichen (**Chr(**10**)) oder eine** Kombination aus Wagenrücklauf- und Zeilenvorschubzeichen (**Chr(**13**) & Chr(**10**)**) zwischen jeder Zeile trennen.
Tasten	Optionale, gemeinsame Werte sind eine Kombination (Summe) der folgenden Konstanten: Konstante Wert Beschreibung vbOKOnly 0 Nur OK-Schaltfläche (Standard) vbOKCancel 1OK- und Abbrechen- Schaltflächen vbAbortRetryIgnore 2Schaltflächen Abbrechen , Wiederholen und Ignorieren vbYesNoCancel 3Ja-, Nein- und Abbrechen- Schaltflächen vbYesNo 4Ja- und Nein-Schaltflächen vbRetryCancel 5Schaltflächen für Wiederholung und Abbrechen vbCritical 16Kritische Meldung vbQuestion 32Warnungsabfrage vbExclamation 48Warnung Meldung vbInformation 64Informationsmeldung vbDefaultButton1 0 Erste Schaltfläche ist Standard (Default) vbDefaultButton2 256Zweite Schaltfläche ist standardmäßig vbDefaultButton3 512Die dritte Schaltfläche ist voreingestellt TIPP: Eine vollständige Liste aller verfügbaren Konstanten finden Sie in der Microsoft VBA-Dokumentation für MsgBox.
Titel	Optional. String-Ausdruck, der in der Titelleiste des Dialogfelds angezeigt wird. Wenn Sie title weglassen, wird der Anwendungsname in der Titelleiste angezeigt.
Ergebnis	Gibt die Benutzeraktion zurück: Konstante Wert Beschreibung vbOK 1OK-Taste gedrückt vbCancel 2 Abbrechen-Taste gedrückt vbAbort 3Abbruch-Taste gedrückt vbRetry 4Retry-Taste gedrückt vbIgnore 5Ignore-Taste gedrückt vbYes 6Ja-Taste gedrückt vbNo 7 Keine Taste gedrückt

```
'Beispiel MsgBox2 Funktion
result = DL.MsgBox2("Diesen Test ausführen?", 3,
"Mein Titel") If result = 6 Then
      DL.AddComment "Ja-Taste gedrückt"
ElseIf result = 7 Then
      DL.AddComment "Keine Taste gedrückt"
Sonst
      DL.AddComment "Abgebrochen"
Ende wenn
```

10.2.2.12 LoadProgramOptions

Lädt die Docklight-Programmoptionen aus einer mit SaveProgramOptions erstellten Datei.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Rückgabewert

Leere

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Syntax

DL.LoadProgramOptions *filePathName*

Die Syntax der LoadProgramOptions-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
filePathName	Erforderlich. Zeichenkette, die den Dateipfad (Verzeichnis und Dateiname) der zu ladenden Docklight-Einstellungsdatei enthält. Wenn kein Verzeichnis angegeben wird, verwendet Docklight das aktuelle Arbeitsverzeichnis. Wenn *filePathName* ein leer ist, wird ein Dateidialog zur Auswahl einer Datei angezeigt.

Bemerkungen

Weitere Informationen zum Speichern und Laden von Docklight-Programmoptionen finden Sie in der Methode SaveProgramOptions.

10.2.2.13 PlaybackLogFile

Öffnet eine bestehende Docklight-Logdatei (HEX-, Dezimal- oder Binärdarstellung) und gibt die Daten einer Kommunikationsrichtung dieser Logdatei wieder (sendet sie erneut).

Startet die Kommunikation, wenn sie nicht bereits läuft (siehe StartCommunication).

Rückgabewert

Leere

Syntax

DL.PlaybackLogFile *filePathName* **[,** *dataDirection***] [,** *timeInterval* **]**

Die Syntax der PlaybackLogFile-Methode besteht aus diesen Teilen:

Teil	Beschreibung
filePathName	Erforderlich. Zeichenkette, die den Dateipfad (Verzeichnis und Dateiname) der Protokolldatei enthält. Wenn kein Verzeichnis angegeben wird, verwendet Docklight das aktuelle Arbeitsverzeichnis. Wenn *filePathName* ein leerer String ist, wird ein Dateidialog angezeigt, um eine Datei auszuwählen.
dataDirection	Optionaler String-Wert. Gibt an, welcher der beiden aufgezeichneten Kommunikationskanäle (TX oder RX? COM1 oder doch COM2?) wiedergegeben werden soll. Wenn *dataDirection* eine leere Zeichenkette ist, wird der erste Kanal, der in der Protokolldatei erscheint, wird verwendet.
timeInterval	Optionaler Integer-Wert. Verwendet eine Pausenzeit in Millisekunden zwischen zwei Meldungen anstelle des ursprünglichen Timings aus der Protokolldatei (siehe Anmerkungen unten).

Bemerkungen

Die Wiedergabe ist nur im Kommunikationsmodus Senden/Empfangen und nur für Protokolldateien in HEX-, Dezimal- oder Binärdarstellung möglich. Für die Wiedergabe können sowohl HTML- (.htm) als auch reine Textdateien (.txt) verwendet werden.

Wenn *filePathName* nicht existiert, meldet Docklight einen Fehler und die Ausführung des Skripts wird abgebrochen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Die verwendete Protokolldatei muss Datums- und Zeitstempel für die beiden Kommunikationsrichtungen enthalten.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

filePathName muss die ursprüngliche Namenserweiterung im Docklight-Stil enthalten, um den Typ der Protokolldatei zu bestimmen, z. B. "log1_hex.txt", "log1_dec.txt" oder "log1_bin.txt". Wenn *filePathName* ein anderes Format hat, wird eine HEX-Protokolldatei angenommen.

PlaybackLogFile wertet die Datums-/Zeitstempel aus der Protokolldatei aus und emuliert das Timing der ursprünglich protokollierten Kommunikation. Wenn Sie dies ändern möchten, z. B. um die Dinge für Debugging-Zwecke zu verlangsamen, können Sie das optionale Argument *timeInterval* verwenden.

Beispiel

```
' Beispiel PlaybackLogFile

' Wiedergabe der ersten Datenrichtung aus einer Beispiel-Logdatei
DL.AddComment "Playback TX side"
DL.PlaybackLogFile "modbus_logfile_hex.txt"

' Dieselbe Datei, aber jetzt werden die Antworten von der RX-
Seite abgespielt DL.AddComment
DL.AddComment
DL.AddComment "Wiedergabe RX-Seite"
DL.PlaybackLogFile "modbus_logfile_hex.txt", "RX"

Dieselbe Datei, aber mit einem festen Zeitabstand zwischen den
einzelnen Sequenzen.
DL.AddComment
DL.AddComment
DL.AddComment "Wiedergabe TX mit festem 500-Millisekunden-
Intervall"
DL.PlaybackLogFile "modbus_logfile_hex.txt", "", 500
```

Wir gehen davon aus, dass die Protokolldatei modbus_logfile_hex.txt während einer früheren Modbus-Kommunikationssitzung erstellt wurde und die folgenden Informationen enthält:

```
8/29/2006 18:45:23.19 [TX] - 01 04 00 00 00 01 31 CA
8/29/2006 18:45:23.34 [RX] - 01 04 02 FF FF B8 80
8/29/2006 18:45:33.14 [TX] - 02 04 00 00 00 01 31 F9
8/29/2006 18:45:33.29 [RX] - 02 04 02 27 10 E7 0C
8/29/2006 18:45:43.23 [TX] - 03 04 00 00 00 01 30 28
8/29/2006 18:45:43.39 [RX] - 03 04 02 00 00 C0 F0
8/29/2006 18:45:58.72 [TX] - 04 04 00 00 00 01 31 9F
8/29/2006 18:45:58.87 [RX] - 04 04 02 04 00 77 F0
```

Nach der Ausführung des Beispielskripts könnte das Kommunikationsfenster wie folgt aussehen:

```
Wiedergab TX-Seite
        e

4/26/2009 13:29:15.841 [TX] - 01 04 00 00 00 01 31 CA
4/26/2009 13:29:25.788 [TX] - 02 04 00 00 00 01 31 F9
4/26/2009 13:29:35.879 [TX] - 03 04 00 00 00 01 30 28
4/26/2009 13:29:51.367 [TX] - 04 04 00 00 00 01 31 9F

Wiedergab RX-Seite
        e

4/26/2009 13:29:51.545 [TX] - 01 04 02 FF FF B8 80
4/26/2009 13:30:01.495 [TX] - 02 04 02 27 10 E7 0C
4/26/2009 13:30:11.596 [TX] - 03 04 02 00 00 C0 F0
4/26/2009 13:30:27.075 [TX] - 04 04 02 04 00 77 F0
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
TX-Wiedergabe mit festem 500-Millisekunden-Intervall
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
4/26/2009 13:30:27.095 [TX] - 01 04 00 00 00 01 31 CA
4/26/2009 13:30:27.595 [TX] - 02 04 00 00 00 01 31 F9
4/26/2009 13:30:28.096 [TX] - 03 04 00 00 00 01 30 28
4/26/2009 13:30:28.596 [TX] - 04 04 00 00 00 01 31 9F
```

10.2.2.14 SaveProgramOptions

Speichert die aktuellen Docklight-Programmoptionen (alles, was im Dialog Optionen eingestellt werden kann) und den aktiven Kommunikationsfenstermodus (ASCII, HEX, Dezimal oder Binär) in eine Datei.

Rückgabewert

Leere

Syntax

DL.SaveProgramOptions *filePathName*

Die Syntax der SaveProgramOptions-Methode besteht aus den folgenden Teilen:

Teil	Beschreibung
filePathName	Erforderlich. Zeichenkette, die den Dateipfad (Verzeichnis und Dateiname) der zu erstellenden Docklight-Einstellungsdatei enthält. Wenn kein Verzeichnis angegeben wird, verwendet Docklight das aktuelle Arbeitsverzeichnis. Wenn *filePathName* ein leer ist, wird ein Dateidialog zur Auswahl einer Datei angezeigt.

Bemerkungen

Eine mit **SaveProgramOptions** erstellte Datei kann mit LoadProgramOptions geladen werden. **SaveProgramOptions** erstellt XML-Dateien (.xml-Dateierweiterung).

SaveProgramOptions und **LoadProgramOptions** sind sehr nützlich, um sicherzustellen, dass Docklight bestimmte Anzeige- und Zeitstempel-Einstellungen für die Ausführung Ihres Docklight-Skripts verwendet. Dies eignet sich hervorragend für automatisierte Testwerkzeuge, die für andere Benutzer bestimmt sind, die mit Docklight nicht vertraut sind. Sie können die entsprechende Display-Darstellung vorbereiten (z.B. nur HEX-Modus) und sicherstellen, dass andere Benutzer die gleiche Display-Ausgabe erhalten wie Sie selbst.

HINWEIS: Die Kommunikation muss gestoppt werden (siehe StopCommunication), bevor Sie **SaveProgrammOptionen** oder **LoadProgrammOptionen**.

Beispiel

```
' Beispiel SaveProgramOptions
DL.StopCommunication DL.SaveProgramOptions
"myFavoriteSettings" DL.Quit
```

Nehmen Sie nun einige Änderungen in den Docklight-Optionen vor, oder ändern Sie das Kommunikationsfenster, indem Sie z.B. die Registerkarte Dezimal wählen. Führen Sie dann das folgende Skript aus:

```
' Beispiel LoadProgramOptions
DL.LoadProgramOptions "myFavoriteSettings"
```

Docklight kehrt nun zu den vorherigen Anzeigeeinstellungen zurück.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.2.2.15 SetChannelSettings

Ändern Sie die aktuellen Einstellungen des Kommunikationskanals: Geben Sie eine neue COM-Port-Nummer oder TCP/IP-Adresse an, oder ändern Sie die Einstellungen des seriellen Anschlusses (Baudrate, Paritätseinstellungen, ...).

Die Einstellungen des seriellen Anschlusses können während des laufenden Betriebs geändert werden, während der Kommunikationskanal geöffnet ist. Für andere Änderungen, z.B. der COM-Port-Nummer selbst, muss StopCommunication aufgerufen werden, bevor **SetChannelSettings** verwendet wird.

HINWEIS:

Rückgabewe

rt Boolean

Syntax

Ergebnis **= DL.SetChannelSettings(** *newSettings* **[,** *channelNo***] [,** *dontTest***])**

Die Syntax der Methode **SetChannelSettings** besteht aus den folgenden Teilen:

Teil	Beschreibung
newSettings	Erforderlich. Zeichenfolge mit dem neuen Kommunikationskanal und/oder den seriellen Einstellungen. Siehe unten für detaillierte Angaben
channelNo	Optional. Ganzzahliger Wert, der den Kommunikationskanal angibt, wenn Kommunikationsmodus: Überwachung verwendet wird. Standardwert ist 1 (Kanal 1).
dontTest	Optional. Boolescher Wert. Wenn *dontTest* auf True gesetzt ist, öffnet und schließt **SetChannelSettings** den Kommunikationskanal nicht zu Testzwecken. Siehe den Abschnitt "Bemerkungen" weiter unten. Standardwert ist False (der Kanal wird getestet, um den Rückgabewert zu bestimmen).

Das Argument *newSettings* akzeptiert die folgenden Werte:

Wert	Beschreibung
"*COMxxx*"	Wählen Sie einen neuen seriellen Kommunikationsanschluss, z. B. "COM7".
"*EntfernterHost*:*EntfernterPort*"	Machen Sie diesen Kanal zu einem TCP-Client und stellen Sie eine Verbindung mit der angegebenen IP-Adresse und TCP-Portnummer her, z. B. "192.0.0.1:3001" (siehe Projekteinstellungen)
"SERVER:*LocalPort*"	Diesen Kanal zu einem TCP-Server machen, der die Verbindungen über den angegebenen TCP-Port, z.B. "SERVER:3001" (siehe Projekteinstellungen)
"UDP:*RemoteHost*:*Port*"	Macht diesen Kanal zu einem UDP-Peer und überträgt Daten an *RemoteHost*:*Port* und Abhören des lokalen *Ports* (siehe Projekteinstellungen)
"USBHID:*vendorId*:*productId"*	Zugriff auf USB-HID-Eingangs-/Ausgangsberichte (siehe Projekteinstellungen).
"PIPE:*myNamedPipe"*	Client-Verbindung zu einer Named Pipe mit Lesen/Schreiben Zugang (siehe Projekteinstellungen).

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

"COMxxx:*BaudRate,Parity, DataBits*,*StopBits*"	Wählen Sie einen neuen seriellen Anschluss und Einstellungen für die serielle Kommunikation *Die Parität* kann NONE, EVEN, ODD, MARK, SPACE sein. Beispiel: "COM18:9600,EVEN,8,1"
"BaudRat,*Parity*,*DataBits*, *StopBits*"	Ändern der seriellen Einstellungen ohne Kenntnis/Änderung des aktuellen seriellen Anschlusses. Beispiel: "38400,KEINE,8,1"
"BaudRat,*Parity*,*DataBits*, *StopBits,FlowControl ParityErrorChar*"	Erweiterte Syntax zur zusätzlichen Änderung der Hardware-Flusssteuerungsoptionen:

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	FlowControl kann OFF, RTSCTS, XONXOFF, RTSSEND sein *ParitätsFehlerZeichen*: Der dezimale ASCII-Code für das Paritätsfehlerzeichen (siehe Projekteinstellungen). Der Standardwert ist 63. Beispiel: "9600,NONE,8,2,RTSCTS,35"
">"	Suchen Sie den nächsten auf diesem PC verfügbaren seriellen COM-Anschluss. Wenn der derzeit ausgewählte Anschluss COM1 ist, beginnt **SetChannelSettings** die Suche bei COM2.

Bemerkungen

Für die meisten Anwendungen ist es nicht notwendig, **SetChannelSettings** oder sein Gegenstück GetChannelSettings zu verwenden. Die Kommunikationsparameter können im Dialog Projekteinstellungen ausgewählt und in der Docklight-Projektdatei gespeichert werden (siehe Speichern und Laden Ihrer Projektdaten und die Methode Projekt öffnen).

Die Methode **SetChannelSettings** ist für fortgeschrittene Docklight-Scripting-Anwendungen gedacht, bei denen die Steuerung der Kommunikationskanaleinstellungen während der Skriptlaufzeit erforderlich ist. Sie ermöglicht es Ihnen, Skripte zu erstellen, die auf verschiedene COM-Ports zugreifen (siehe Beispiel unten) oder eine Liste von IP-Adressen durchlaufen.

SetChannelSettings-Methode erzeugt einen Fehler, wenn ein unzulässiger Wert mit *newSettings*.

Wenn das Argument *newSettings* gültig ist (und das Flag *dontTest* nicht gesetzt ist), wird der Kommunikationskanal für einen Test sofort geöffnet und wieder geschlossen.

Wenn *dontTest* True ist, wird **SetChannelSettings** den Kanal nicht zum Testen öffnen/schließen und immer True zurückgeben. Dies ist nützlich bei Netzwerkanwendungen, bei denen zusätzliche Verbindungs-/Trennungsversuche den anderen Host/das andere Gerät verwirren könnten. Probleme sind zum Beispiel bei Telnet-Server-Anwendungen aufgetreten.

Der Rückgabewert von **SetChannelSettings** ist True, wenn der Kanal erfolgreich geöffnet werden konnte (oder die neuen Einstellungen in Ordnung sind und *dontTest* true ist).
Der Rückgabewert ist False, wenn die Einstellungen ungültig sind oder beim Versuch, auf den Port zuzugreifen, ein Fehler auftrat (z.B. der COM-Port ist bereits in Gebrauch, oder die Firewall hat den TCP/IP-Zugriff verweigert).

HINWEIS: Die Änderung des Parameters *FlowControl*, wenn Project Settings: Flow Control einen anderen Wert als "Aus" hat, kann zu undefiniertem Verhalten führen.

Siehe auch GetChannelSettings und GetChannelStatus.

Beispiel

```
' Beispiel SetChannelSettings /
GetChannelSettings ' (erfordert Docklight im
Sende-/Empfangsmodus)

DL.ClearCommWindows

DL.AddComment "Suche nach dem ersten verfügbaren COM-Anschluss an
diesem PC..."
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
portAvailable = DL.SetChannelSettings("COM1:9600,NONE,8,1")
While Not portAvailable
    oldPort = DL.GetChannelSettings()
    ' nächsten COM-Anschluss
    ausprobieren
    portAvailable = DL.SetChannelSettings(">")
    newPort = DL.GetChannelSettings()
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
    ' bereits alle COM-Ports auf diesem PC
    ausprobiert? If (oldPort = newPort) Then
        DL.AddComment "Kein COM-Anschluss
        verfügbar" DL.Quit
    Ende
Wenn Wend
DL.AddComment "COM-Anschluss verwenden " & DL.GetChannelSettings()

' Probieren Sie einige verschiedene
Baudraten aus baudRatesStr =
"9600,14400,57600,115200"
baudRatesArray = Split(baudRatesStr, ",")
For i = 0 To UBound(baudRatesArray)
    ' Die Einstellungen der seriellen Schnittstelle anpassen
    DL.SetChannelSettings(baudRatesArray(i) + ",NONE,8,1")
    DL.AddComment
    DL.AddComment
    DL.AddComment "Testen mit Einstellungen " &
DL.GetChannelSettings()
    ' Senden Sie einen Modemtestbefehl und warten Sie einige Zeit
auf die Antwort
    DL.StartCommunication
    DL.SendSequence "", "ATI3" + Chr(13) + Chr(10)
    DL.Pause 200
    DL.StopCommunication
Weiter
```

Nachdem das Skript auf einem Computer mit eingebautem Modem an COM3 ausgeführt wurde, könnte das Docklight-Kommunikationsfenster wie folgt aussehen:

```
 Suche nach dem ersten verfügbaren COM-Port auf diesem
 PC... Verwendung des COM-Anschlusses
 COM3:9600,NONE,8,1


 Test mit Einstellungen COM3:9600,NONE,8,1

28.01.2008 16:28:36.26 [TX] - ATI3<CR><LF>

28.01.2008 16:28:36.26 [RX] - ATI3<CR>
<CR><LF>
Agere SoftModem Version 2.1.46<CR><LF>
<CR><LF>
OK<CR><LF>


 Test mit Einstellungen

COM3:14400,NONE,8,1 28.01.2008 16:28:37.46

[TX] - ATI3<CR><LF>

28.01.2008 16:28:37.46 [RX] - ATI3<CR>
<CR><LF>
Agere SoftModem Version 2.1.46<CR><LF>
<CR><LF>
OK<CR><LF>


 Test mit Einstellungen
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
COM3:57600,NONE,8,1 28.01.2008 16:28:38.60

[TX] - ATI3<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
28.01.2008 16:28:38.60 [RX] - ATI3<CR>
<CR><LF>
Agere SoftModem Version 2.1.46<CR><LF>
<CR><LF>
OK<CR><LF>


 Test mit Einstellungen COM3:115200,NONE,8,1

28.01.2008 16:28:39.73 [TX] - ATI3<CR><LF>

28.01.2008 16:28:39.73 [RX] - ATI3<CR>
<CR><LF>
Agere SoftModem Version 2.1.46<CR><LF>
<CR><LF>
OK<CR><LF>
```

10.2.2.16 SetContentsFilter

Verwenden Sie eine andere Einstellung für den **Inhaltsfilter** als die, die im Dialogfeld Projekteinstellungen - Kommunikationsfilter festgelegt wurde.

Rückgabewert

Leere

Syntax

DL.SetContentsFilter *newContentsFilter*

Die Syntax der Methode **SetContentsFilter** besteht aus den folgenden Teilen:

Teil	Beschreibung
newContentsFilter	Erforderlich. Ganzzahliger Wert zur Auswahl des neuen Filters: 0 = Alle ursprünglichen Kommunikationsdaten anzeigen (Kanal 1 und Kanal 2) 1 = Nur Kanal 1 oder [TX]-Daten anzeigen 2 = Nur Kanal 2 oder [RX]-Daten anzeigen 3 = Alle ursprünglichen seriellen Daten ausblenden, nur zusätzliche Kommentare anzeigen

Bemerkungen

Nach Beendigung der Skriptausführung wird der **Inhaltsfilter** auf die ursprüngliche Projekteinstellung gesetzt, die unter Projekteinstellungen - Kommunikationsfilter definiert ist.

Beispiel

```
' Benötigt das Docklight-Basisbeispielprojekt "PingPong" und
einen Loopback auf dem gewählten
' Kommunikationskanal
DL.OpenProject "PingPong"
DL.ClearCommWindows
DL.SendSequence "Ping"
DL.Pause 50
DL.AddComment vbCrLf + "SetContentsFilter(1) " :
DL.SetContentsFilter(1)
DL.Pause 50
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.AddComment vbCrLf + "SetContentsFilter(2) " :
DL.SetContentsFilter(2)
DL.Pause 50
DL.AddComment vbCrLf + "SetContentsFilter(3) " :
DL.SetContentsFilter(3)
DL.Pause 50
```

Nach der Ausführung des Skripts könnte das Docklight-Kommunikationsfenster wie folgt aussehen:

```
7/30/2012 17:42:31.322 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
7/30/2012 17:42:31.326 [RX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
 "Ping" empfangen
7/30/2012 17:42:31.350 [TX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
7/30/2012 17:42:31.352 [RX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Pong" erhalten
7/30/2012 17:42:31.499 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
SetContentsFilter(1)
 "Ping" empfangen
7/30/2012 17:42:31.523 [TX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Pong" erhalten
7/30/2012 17:42:31.547 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
 "Ping" empfangen
7/30/2012 17:42:31.572 [TX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Pong" erhalten
7/30/2012 17:42:31.594 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
 "Ping" empfangen
7/30/2012 17:42:31.619 [TX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
SetContentsFilter(2)

7/30/2012 17:42:31.621 [RX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Pong" erhalten 2D 2D 2D 2D 6F 20 50 69 6E 67 "Ping" erhalten
6F 2D 2D 2D 2D 20 50 6F 6E 67 "Pong" erhalten 2D 2D 2D 2D 6F
20 50 69 6E 67 "Ping" empfangen 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Pong" erhalten
SetContentsFilter(3)
 "Ping" empfangen "Pong" empfangen "Ping" empfangen "Pong"
empfangen "Ping" empfangen "Pong" empfangen
```

10.2.2.17 SetHandshakeSignals

Setzt die RTS- und DTR-Handshake-Signale. Nur erlaubt, wenn Flow Control: Manuell verwendet wird.

Syntax

DL.SetHandshakeSignals *rts*,

dtr

Die Syntax der Methode **SetHandshakeSignals** besteht aus den folgenden Teilen:

Teil	Beschreibung
rts	Erforderlich. Boolescher Wert, um RTS = Hoch (True) oder RTS = Niedrig (False) zu setzen
dtr	Erforderlich. Boolescher Wert zum Setzen von DTR = High (True) oder DTR = Low (Falsch)

Bemerkungen

Siehe auch die Funktion GetHandshakeSignals zum Auslesen des aktuellen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Zustands der Handshake-Signale.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

SetHandshakeSignals kann vor dem Öffnen des Kommunikationskanals verwendet werden, um einen bestimmten Zustand der RTS- und DTR-Leitungen bei der Initialisierung sicherzustellen.

Beispiel

```
' Beispiel SetHandshakeSignals
DL.SetHandshakeSignals true, false
DL.StartCommunication
DL.Pause 1000
DL.SetHandshakeSignals false, true
DL.Pause 1000
```

10.2.2.18 SetUserOutput

Erstellen Sie eine zusätzliche Registerkarte für Benutzerausgaben im Dokumentations-/Skriptbereich des Docklight-Hauptfensters. Fügen Sie einfache Text- oder formatierte RTF-Ausgaben hinzu, um visuelles Benutzerfeedback und/oder menüähnliche Interaktion (mit GetKeyState) zu ermöglichen.

Syntax

DL.SetUserOutput *text* **[,** *rtfFormat***] [,** *append***] [,** *readFromFile***]**

Die Syntax der Methode **SetUserOutput** besteht aus folgenden Teilen:

Teil	Beschreibung
Text	Erforderlich. Ihre Ausgabe, die angezeigt oder hinzugefügt werden soll. Ist der *Text* eine leere Zeichenkette, wird die Ausgabelasche wieder aus dem Docklight-Hauptfenster entfernt.
rtfFormat	Optionaler boolescher Wert. False (Standard) = *Text* ist reiner Text True = *Text* ist ein gültiges RTF-Dokument, z. B. der Inhalt einer RTF-Dokumentendatei.
anhängen.	Optionaler boolescher Wert. True (Standard) = *Text* zum bereits vorhandenen Inhalt hinzufügen. False = den vorhandenen Inhalt durch *Text* ersetzen.
readFromFile	Optionaler boolescher Wert. False (Standard) = *Text* enthält den eigentlichen Text oder das RTF-Dokument True = *Text* ist eine Datei und ihr Inhalt soll angezeigt werden. HINWEIS: Wenn eine Datei nicht existiert oder nicht geladen werden kann, wird **SetUserOutput** verwendet stattdessen eine leere Zeichenkette und weist nicht auf einen Fehler hin.

Bemerkungen

SetUserOutput kann Ihnen helfen, eine benutzerdefinierte Ausgabe zu erstellen oder menüähnliche Benutzerinteraktionen anzubieten. Ein Beispiel könnte die Auswahl aus verschiedenen vordefinierten Tests in einem automatisierten Gerätetest sein.

Siehe das Beispielskript **SetUserOutput_Example.pts** im Ordner ScriptSamples\Extras\SetUserOutput_Example für die verschiedenen Möglichkeiten zum Laden / Hinzufügen von formatiertem Text.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiel

```
DL.SetUserOutput "" ' Ausgabe löschen
DL.Pause 1000
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.SetUserOutput "Ein gutes altes 'Hello
World!'" DL.Pause 5000
DL.SetUserOutput "... mehr Text ..."
DL.Pause 5000

Sie können Ihre RTF-Dokumente mit Windows Wordpad erstellen
und
Öffnen Sie die resultierende .rtf-Datei mit Windows Notepad,
um den RTF-Code zu sehen.
rtfExample =
"{\rtf1\ansi\ansicpg1252\deff0{\fonttbl{\f0\fnil\fcharset0
Harlow Solid Italic;}}{\colortbl ;\red75\green172\blue198;}
\cf1\f0\fs32 A Curly Hello!\par}" DL.SetUserOutput
"... etwas bunter ..." DL.SetUserOutput
rtfExample, True

DL.Pause 5000
DL.SetUserOutput "Das Ende. Ausgabe-Registerkarte wird bald
 entfernt...", False, False
DL.Pause 5000
DL.SetUserOutput ""
```

Beispiel 2 (RTF-Datei lesen und anzeigen)

```
DL.SetUserOutput "C:\mytext.rtf", True, False, True
```

10.2.2.19 SetWindowLayout

Steuert das Aussehen des Docklight-Hauptfensters, ähnlich wie das Menü **Extras > Minimieren/Wiederherstellen...** und die F12-Hotkeys.

Syntax

DL.SetWindowLayout [*layout* **]**

Die Syntax der Methode **SetWindowLayout** besteht aus folgenden Teilen:

Teil	Beschreibung
Layout	Optionales String-Argument. Eine Kombination aus den folgenden Buchstaben: S - Sequenzbereich sichtbar D - Dokument-/Skriptbereich sichtbar Standardwert ist "SD" - zeigt sowohl die Sequenzlisten als auch den Dokument-/Skriptbereich an.

Bemerkungen

SetWindowLayout ist nützlich, wenn Sie ein fertiges Skript/Projekt haben und der Endbenutzer nicht durch Details Ihres Docklight-Projekts und Ihrer Skriptdefinitionen abgelenkt werden soll. Indem Sie einige Bereiche des Docklight-Hauptfensters ausblenden, können Sie den Fokus auf die Ausgabe des Kommunikationsfensters oder auf eine von Ihnen erstellte SetUserOutput-Anzeige legen.

Beispiel

```
' Beispiel SetWindowLayout
' zeigt nur das Kommunikationsfenster. Kein Dokument-
/Skriptbereich, kein Sende-/Empfangsreihenfolgebereich
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.SetWindowLayout ""
DL.Pause 4000
' Kommunikationsfenster und Sende-/Empfangsfolgelisten
anzeigen, aber keinen Dokument-/Skriptbereich
DL.SetWindowLayout "S"
DL.Pause 4000
' alles anzeigen (Standard, gleich wie <Layout> = "SD")
DL.SetWindowLayout
```

10.2.2.20 ShellRun

Startet eine externe Anwendung oder führt einen *Windows* Shell-Vorgang aus.

Rückgabewe

rt Integer

Syntax 1

result **= DL.ShellRun(***operation*, *file* **[,** *parameters***] [,** *directory***] [,** *showCmd***])**

Die ShellRun-Methode Syntax 1 besteht aus folgenden Teilen:

Teil	Beschreibung
Betrieb	Erforderlich. Zeichenkette, die die Art der auszuführenden Aktion angibt, entsprechend dem *Windows* ShellExecute-Parameter lpOperation. Üblicherweise verwendete Werte sind: "Öffnen" - öffnet eine Datei oder einen Ordner "Drucken" - druckt eine Datei "Bearbeiten" - startet einen Editor und öffnet die Datei zur Bearbeitung Weitere Werte und Dokumentation finden Sie in der Windows-Dokumentation ShellExecute.
Datei	Erforderlich. Zeichenfolge mit dem Pfad zu der Datei, die ausgeführt werden soll *Betrieb*.
Parameter	Optional. String-Wert. Wenn die *Datei* eine ausführbare Datei ist, können hier Befehlszeilenargumente übergeben werden.
Verzeichnis	Optional. String-Wert. Legt das Arbeitsverzeichnis für die Aktion fest. Wenn nicht verwendet, wird das aktuelle Docklight-Arbeitsverzeichnis verwendet.
showCmd	Optionaler Integer-Wert. Gibt das Erscheinungsbild der Anwendung an, das dem Parameter *Windows* ShellExecute nShowCmd entspricht. Der Standardwert ist: 10 - SW_SHOWDEFAULT andere gängige Werte sind: 0 - SW_HIDE 1 - SW_SHOWNORMAL 2 - SW_SHOWMINIMIZED 3 - SW_SHOWMAXIMIERT

Bemerkungen (Syntax 1)

result enthält den Rückgabewert von *Windows* ShellExecute.

Bei Erfolg ist das *Ergebnis* **> 32**.

Siehe das folgende Beispiel für eine Definition der relevanten Fehlerkonstanten für das *Ergebnis*.

Syntax 2

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

result **= DL.ShellRun(**"wait<timeout>**",** [, *file*] [, *parameters*] [, *directory*] [, *showCmd*]**)**

Die ShellRun-Methode Syntax 1 besteht aus folgenden Teilen:

Teil	Beschreibung
"wait<timeout>"	Anstelle von Operation wie in Syntax 1 verwenden Sie: "warten" - eine externe Anwendung starten und warten, bis sie beendet wird oder "wait<timeout>", z.B. "wait60000" - startet eine externe Anwendung und wartet darauf, dass sie beendet wird, oder dass die angegebene Zeitspanne in Millisekunden abläuft.
Datei *Parameter* *Verzeichnis* *showCmd*	wie in Syntax 1

Bemerkungen (Syntax 2):

Der Befehl "wait<timeout>" startet und überwacht eine externe Anwendung über Windows ShellExecuteEx, ähnlich wie der Befehl "open" in Syntax 1.

Die Rückgabewerte in Syntax 2 unterscheiden sich von denen in Syntax 1:

Ergebnis	Beschreibung
0	erfolgreich
-1	Fehler beim Starten der Anwendung
-2	Timeout

Beispiel

```
' Beispiel
ShellRun '
Hier ist eine Liste von Fehlercodes, entsprechend der originalen
C-Header-Datei shellapi.h aus Windows Kit 8.1:
' reguläre WinExec()-Codes */
const SE_ERR_FNF = 2          ' Datei nicht gefunden
const SE_ERR_PNF = 3          ' Pfad nicht
gefunden const SE_ERR_ACCESSDENIED = 5 '
Zugriff verweigert const SE_ERR_OOM = 8     '
kein Speicher vorhanden const
SE_ERR_DLLNOTFOUND = 32
Fehlerwerte für ShellExecute() über die normalen WinExec()-Codes
hinaus
const SE_ERR_SHARE = 26
const SE_ERR_ASSOCINCOMPLETE = 27
const SE_ERR_DDETIMEOUT = 28
const SE_ERR_DDEFAIL = 29
const SE_ERR_DDEBUSY = 30
const SE_ERR_NOASSOC = 31

' Beispiel für Syntax 2
DL.AddComment "Notepad öffnen und auf das Ende der Anwendung
warten. Nach maximal 60 Sekunden Timeout zwangsweise schließen:"
DL.AddComment "DL.AddComment DL.ShellRun(" & Chr(34) &
"wait60000" & Chr(34) & ", " & Chr(34) & "notepad.exe" &
Chr(34) & ", " & Chr(34) & "test.txt" & Chr(34) & ")"
DL.AddComment "Rückgabe = " & DL.ShellRun("wait60000",
"notepad.exe", "test.txt")
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.Pause 1000
' Beispiele für Syntax 1
DL.AddComment "Windows auffordern, die gleiche Datei mit der
Standardanwendung für die Bearbeitung von .txt-Dateien zu
öffnen, sofort fortfahren:" DL.AddComment "DL.ShellRun(" &
Chr(34) & "edit" & Chr(34) & ", " & Chr(34) & "test.txt" &
Chr(34) & ")"
DL.AddComment "Rückgabe = " & DL.ShellRun("edit", "test.txt")
DL.Pause 3000
DL.AddComment "Öffnen Sie die Docklight-Website im
Standardbrowser:"
DL.AddComment "DL.ShellRun(" & Chr(34) & "open" & Chr(34) & ",
" & Chr(34) & "https://docklight.de" & Chr(34) & ")"
DL.AddComment "Rückgabe = " & DL.ShellRun("open",
"https://docklight.de")
DL.Pause 3000
DL.AddComment "Windows Geräte-Manager öffnen:"
DL.AddComment "DL.ShellRun(" & Chr(34) & "open" & Chr(34) & ",
" & Chr(34) & "devmgmt.msc" & Chr(34) & ")"
DL.AddComment "Rückgabe = " & DL.ShellRun("open", "devmgmt.msc")
DL.Pause 1000
DL.AddComment "erledigt"
```

10.2.2.21 UploadDatei

Öffnet eine vorhandene Datei und sendet ihren Inhalt. Startet die Kommunikation, wenn sie nicht bereits läuft (siehe StartCommunication).

Rückgabewert

Leere

Syntax

DL.UploadFile *filePathName* **[,** *representation***]**

Die Syntax der UploadFile-Methode besteht aus diesen Teilen:

Teil	Beschreibung
filePathName	Erforderlich. Zeichenkette, die den Dateipfad (Verzeichnis und Dateiname) der zu sendenden Datei enthält. Wenn kein Verzeichnis angegeben wird, verwendet Docklight das aktuelle Arbeitsverzeichnis. Wenn *filePathName* eine leere Zeichenkette ist, wird eine Datei wird ein Dialogfeld zur Auswahl einer Datei angezeigt.
Vertretung	Optional. String-Wert, um das Format der Datei *filePathName zu* definieren. "A" = ASCII (Standard): *filePathName* ist eine Textdatei, die direkt gesendet wird, ohne weiteres Parsing. "H" = HEX: *filePathName* enthält HEX-Sequenzdaten, z. B. 5F 54 65 73 74 ... "D" = Dezimal: *filePathName* enthält dezimale Sequenzdaten, z. B. 095 084 101 115 ... "B" = Binär: *filePathName* enthält binäre Sequenzdaten, z. B. 01011111 01010100 ... "R" = Rohdaten: *filePathName* ist eine Binärdatei, die unverändert gesendet werden

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	muss.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Bemerkungen

Das Hochladen von Dateien ist nur im Kommunikationsmodus Senden/Empfangen möglich.

Wenn *filePathName* nicht existiert, meldet Docklight einen Fehler und die Ausführung des Skripts wird abgebrochen.

Die Standard-ASCII-Darstellung "A" ermöglicht das Senden von Textdateien ohne weitere Änderungen. Für unbearbeitete Binärdateien, die unverändert gesendet werden müssen, verwenden Sie die
Option "R" (Rohdaten). Sie ist nicht zu verwechseln mit der "B"-Darstellung (Binary), die von Docklight verwendet wird, um Daten nur mit 0 und 1 darzustellen.

Sie können die UploadFile-Methode verwenden, um den Inhalt einer Docklight-Logdatei zu übertragen. Bitte stellen Sie sicher, dass Ihre Protokolldatei im reinen Textmodus vorliegt (siehe Protokolldateieinstellungen) und die Datei nur die Rohdaten enthält, ohne zusätzliche Kommentare und ohne Datums-/Zeitstempel (siehe Optionen).

Die UploadFile-Methode unterstützt keine spezifischen Compiler-Ausgabedateiformate, wie z.B. "Intel HEX File". Wenn Sie spezielle Anforderungen haben, wenden Sie sich bitte an unseren E-Mail-Support.

HINWEIS: Die Daten werden in Blöcken von maximal 512 Bytes gesendet. 512 Bytes gesendet. Wenn Sie während eines Datei-Uploads manuell eine Sendesequenz senden, wird die Sequenz zwischen einem dieser Blöcke gesendet und beschädigt die Datenübertragung.

Beispiel

```
' Beispiel Upload

Datei ' Senden einer

Textdatei
DL.UploadFile "helloworld.txt", "A"

' Binäre Rohdaten direkt senden
DL.UploadFile "test.dat", "R"

' Parsen und Senden einer HEX-Datei
DL.UploadFile "hexfile.txt", "H"
```

10.2.3 Eigenschaften

10.2.3.1 NoOfSendSequences

Gibt die Anzahl der im aktuellen Docklight-Projekt definierten Sendesequenzen zurück.

Rückgabewe

rt Integer

Syntax

Ergebnis = **DL.NoOfSendSequences**

Bemerkungen

Die Eigenschaft **NoOfSendSequences** ist sehr nützlich, um Schleifenstrukturen zu erstellen, die alle verfügbaren Sendesequenzen nutzen. Siehe das folgende Beispiel.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Beispiel

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' Beispiel NoOfSendSequences

' Alle definierten Sendesequenzen mit einer Verzögerung von 1
Sekunde ' zwischen den einzelnen Sequenzen aussenden
For i = 0 To (DL.NoOfSendSequences - 1)
   DL.SendSequence i
   DL.Pause 1000
Weiter
```

10.2.3.2 NoOfReceiveSequences

Gibt die Anzahl der im aktuellen Docklight-Projekt definierten Empfangssequenzen zurück.

Rückgabewe

rt Integer

Syntax

Ergebnis =

DL.NoOfReceiveSequences

Bemerkungen

Siehe NoOfSendSequences.

10.3 OnSend / OnReceive Ereignisprozeduren

Docklight Scripting unterstützt zwei spezielle Prozeduren, die von der Docklight Scripting Engine vor dem Senden einer neuen Sendesequenz oder nach dem Erkennen einer Receive Sequence aufgerufen werden.

Definition des Verfahrens	Beschreibung
Sub DL_OnSend() *... mein Skriptcode ...* **End Sub**	**DL_OnSend()** wird aufgerufen, nachdem ein neuer Sendevorgang ausgelöst wurde (manuelles Senden oder DL.SendSequence). Es stehen spezielle Manipulationsfunktionen zur Verfügung, mit denen die Daten ausgelesen und verändert werden können, bevor sie tatsächlich übertragen werden. Siehe Manipulation von Sendesequenzdaten.
Sub DL_OnReceive() *... mein Skriptcode ...* **End Sub**	**DL_OnReceive()** wird aufgerufen, nachdem eine Empfangssequenz erkannt wurde. Zum Auslesen und Weiterverarbeiten der empfangenen Daten stehen spezielle Manipulationsfunktionen zur Verfügung. Siehe Auswertung der empfangenen Sequenzdaten.

Die Prozeduren können an beliebiger Stelle im Skriptcode auf Modulebene (nicht innerhalb einer Klasse) definiert werden. Ein Beispiel finden Sie unter Manipulation von Sendesequenzdaten.

HINWEIS: Der Code **DL_OnSend()** und **DL_OnReceive()** wird nur ausgeführt, während das Skript läuft. Das Senden einer Sendesequenz führt nicht automatisch den zugehörigen **DL_OnSend()-Code** aus. Das Skript muss manuell über das Menü **Scripting > Run Script** gestartet werden. Jeder Fehler während der Skriptausführung stoppt das Skript und verhindert, dass weitere **DL_OnSend()** / **DL_OnReceive()** Prozeduraufrufe gemacht werden.

HINWEIS: Die Ereignisse **DL_OnSend()** und **DL_OnReceive() stehen** in einer

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Warteschlange und können zu einem späteren Zeitpunkt verarbeitet werden. Weitere Informationen finden Sie unter Timing und Programmablauf.

TIPP: Wenn Ihr Skript nur aus den Prozeduren **DL_OnSend()** und **DL_OnReceive()** und sonst nichts besteht, verwenden Sie eine einfache Endlosschleife auf Modulebene, um zu verhindern, dass das Skript sofort beendet wird. Siehe das Beispiel für die Manipulation von Sendesequenzdaten.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.3.1 Sub DL_OnSend() - Manipulation der Sendesequenzdaten

Um zusätzliche Berechnungen und Algorithmen (z.B. Prüfsummen) auf Sendesequenzdaten zu ermöglichen, kann das folgende Verfahren in einem Docklight-Skript definiert werden:

Sub DL_OnSend()
... mein Skriptcode ...
End Sub

Vor dem Versenden einer neuen Sendesequenz wird die Prozedur **DL_OnSend()** von der Docklight Script Engine aufgerufen. Innerhalb der Prozedur **DL_OnSend()** stehen die folgenden Funktionen zur Verfügung, um die aktuellen Sequenzdaten zu lesen und zu manipulieren:

Funktion	Beschreibung
Ergebnis = **DL.OnSend_GetSize()**	Gibt die Größe der gesendeten Daten / Anzahl der Zeichen zurück
Ergebnis = **DL.OnSend_GetName()**	Gibt den Namen der zu übertragenden Sendesequenz zurück. Handelt es sich um eine benutzerdefinierte Datensequenz, die durch einen DL.SendSequence-Befehl erstellt wurde, ist der Rückgabewert eine leere Zeichenkette ("”).
Ergebnis = **DL.OnSend_GetIndex()**	Gibt ihren Index innerhalb der Sendesequenzliste zurück. Handelt es sich um eine benutzerdefinierte Datensequenz, die durch einen DL.SendSequence-Befehl erstellt wurde, lautet der Rückgabewert -1.
Ergebnis = **DL.OnSend_GetData(** **[***Repräsentation***])** Syntax 2: *Ergebnis* = **DL.OnSend_GetData([***Darstellung***]** **[,** *Start***] [,** *Länge***])**	Gibt eine Zeichenkette zurück, die die tatsächlichen Sendedaten enthält. *Die Darstellung* gibt das Format des *Ergebnisses* an: "A" = ASCII (Standard), "H" = HEX, "D" = Dezimal oder "B" = Binär. Die zurückgegebenen Daten enthalten keine Wildcards. Alle Platzhalterpositionen sind bereits durch tatsächliche Zeichen ersetzt worden. HINWEIS: Wenn die ursprüngliche Sendesequenz '#' Platzhalter (null oder ein Zeichen) enthält, kann die Länge der **DL.OnSend_GetData()**-Sequenz kürzer sein als die ursprüngliche Sequenz mit Platzhaltern. Syntax 2: Gibt eine Zeichenkette zurück, die eine bestimmte Anzahl von Zeichen aus den empfangenen Daten enthält. *Start:* Bereich: 1 ... **DL.OnSend_GetSize()**, oder -1 = Start beim letzten Zeichen, -2 = Start beim vorletzten Zeichen, ... Standardwert ist 1. *length:* Anzahl der Zeichen, oder -1 = bis zum letzten Zeichen, -2 = bis zum vorletzten Zeichen, ... Der Standardwert ist -1.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

DL.OnSend_SetData *newData* **[,** *Vertretung***]**	Ersetzt die zu übertragenden Daten durch die in der Zeichenkette *newData* angegebenen Daten. *representation* gibt das Format von *newData* an "A" = ASCII (Standard), "H" = HEX, "D" = Dezimal oder "B" = Binär. Nach dem Beenden der Prozedur **DL_OnSend()** sendet Docklight *newData*, unabhängig von

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	wie die ursprüngliche Sendesequenz aussah. Die Länge von *newData* kann sich von der Länge der ursprünglichen Sendesequenz unterscheiden. HINWEIS: Wenn *newData* eine leere Zeichenfolge ist, wird die Übertragung der ursprünglichen Sendesequenz effektiv unterdrückt.
DL.OnSend_Poke *charNo*, *Wert*	Setzt das Zeichen an der Position *charNo* auf *value*. *value* ist das neue Zeichen als Integer-Zahl von 0..255. Siehe auch **DL.OnSend_Peek(...)**
Ergebnis = **DL.OnSend_Peek(** *charNo* **)** Syntax 2: *result* = **DL.OnSend_Peek(** *charNo*, *representation* **)**	Gibt ein Zeichen der Sendedaten als Ganzzahlwert von 0..255 zurück. *charNo* ist die Position innerhalb der Sendedaten. Gültiger Bereich für *charNo*: 1 .. **DL.OnSend_GetSize()**, oder -1 = Start beim letzten Zeichen, -2 = Start beim vorletzten Zeichen, ... Syntax 2: Gibt eine Zeichenkette anstelle eines Integer-Wertes zurück. *Die Darstellung* gibt das Format an: "A" = ASCII, "H" = HEX, "D" = Dezimal oder "B" = Dezimal Binär.

Bemerkungen

Mit den Funktionen **DL.OnSend_GetSize()**, **DL.OnSend_Peek(..)** und **DL.OnSend_Poke** können Prüfsummenberechnungen und andere Algorithmen leicht implementiert werden. Siehe das Beispiel unten.

Die Prozedur **DL_OnSend()** wird nur ausgeführt, während das Skript läuft. Während der Ausführung des Codes **DL_OnSend()** werden keine weiteren Kommunikationsverarbeitungen und Anzeigeaktualisierungen durchgeführt. Um Leistungs- und Zeitprobleme zu vermeiden, sollten Sie die Ausführungszeit gering halten. Vermeiden Sie z. B. verschachtelte Schleifen und führen Sie keine zeitaufwändigen Berechnungen durch.

Wie Docklight Sendedaten-Ereignisse behandelt und den Codeabschnitt **DL_OnSend()** ausführt, erfahren Sie unter Zeitplanung und Programmablauf.

Beispiel

```
' Beispiel DL_OnSend()-

Ereigniscode ' Vordefinierte

Sendesequenzen
' (0) Test: TestX<CR><NUL>

' Endlosschleife, um zu verhindern, dass das Skript sofort
beendet wird
Do
    DL.Pause 1 ' (die Pause reduziert die CPU-Last im
Leerlauf) Schleife

Sub DL_OnSend()
    ' Einfache Prüfsumme: Letztes Byte der Sequenz
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' ist eine PrÜfsumme Über alle vorherigen Bytes,
mod 256 seqSize = DL.OnSend_GetSize()
' benÖtigen wir mindestens eine Drei-Byte-Sequenz
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
    Wenn seqSize > 2 Dann
        ' anstelle des "X" nach Test ein zufälliges Zeichen
        einfügen DL.OnSend_Poke seqSize - 2, 65 + Rnd * 25
        ' eine einfache Prüfsumme über die neue Sequenz berechnen
        chksumHex = DL.CalcChecksum("MOD256",
DL.OnSend_GetData("H"), "H", 1, seqSize -1)
        ' Überschreiben des letzten Zeichens der
Sendesequenz mit dem aktuellen Prüfsummenwert
        DL.OnSend_Poke seqSize, CInt("&h" + chkSumHex)
        ' Verwendung der Funktion Peek für zusätzliche
        Dokumentation DL.AddComment vbCrLf & vbCrLf
        DL.AddComment "Prüfsumme ein", False, False
        For i = 1 To seqSize - 1
            DL.AddComment " " & DL.OnSend_Peek(i, "H"), False,
Falsch
        Weiter
        DL.AddComment " ist " & DL.OnSend_Peek(seqSize, "H") &
"(Hex), " & DL.OnSend_Peek(seqSize, "D") & "(Decimal)"
    End If
End Sub
```

Nach dem Start des Skripts und dem zweimaligen manuellen Senden der "Test"-Sequenz konnte das ASCII-Kommunikationsfenster von Docklight die folgende Ausgabe anzeigen:

```
 Prüfsumme auf 54 65 73  53 0D  ist 00(Hex), 000(Dezimal)
           74

23.06.2015 11:28:31.695 [TX] - 54 65 73 74 53 0D 00

 Prüfsumme auf 54 65 73  4E 0D  ist FB(Hex), 251(Dezimal)
           74

23.06.2015 11:28:32.568 [TX] - 54 65 73 74 4E 0D FB
```

HINWEIS: Die Berechnung und Validierung von Prüfsummen und das Beispiel für das Modbus-Protokoll beschreiben, wie gängige CRCs und andere Prüfsummen ohne DL_OnSend() / DL_OnReceive()-Code berechnet und validiert werden können. Diese Verarbeitung erfolgt, bevor die Sequenzdaten an die DL_OnSend()-Prozedur übergeben werden. Wenn Sie jedoch Ihre Sendesequenzdaten vor dem Senden ändern wollen *und* eine Prüfsumme für die geänderten Daten benötigen, ist das obige Beispiel die richtige Lösung.

10.3.2 Sub DL_OnReceive() - Auswertung der empfangenen Sequenzdaten

Um die Receive Sequence-Daten zu analysieren (z.B. die tatsächlich empfangenen Werte für einen Wildcard-Bereich zu überprüfen) oder zusätzliche Aufgaben nach dem Empfang der Sequenz durchzuführen, kann die folgende Prozedur in einem Docklight-Skript definiert werden:

Sub DL_OnReceive()
... mein Skriptcode ...
End Sub

Nach dem Erkennen einer neuen Empfangssequenz und dem Ausführen der vordefinierten Aktionen (Kommentar hinzufügen, Sequenz senden, ...) wird die Prozedur **DL_OnReceive()** von der Docklight Script Engine aufgerufen. Innerhalb der Prozedur **DL_OnReceive() stehen** die folgenden Funktionen zur Verfügung, um die Daten der Receive Sequence auszulesen:

Funktion	Beschreibung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Ergebnis = **DL.OnReceive_GetSize()**	Gibt die Größe der empfangenen Daten / Anzahl der Zeichen zurück

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Ergebnis = **DL.OnReceive_GetName()**	Gibt den Namen der entsprechenden Empfangssequenz zurück.
Ergebnis = **DL.OnReceive_GetIndex()**	Gibt den Index innerhalb der Liste der Empfangssequenzen zurück
Ergebnis = **DL.OnReceive_GetData(** [*Darstellung*] **)** Syntax 2: *Ergebnis* = **DL.OnReceive_GetData(** [*Repräsentation*] **[,** *Start***] [,** *Länge***])**	Gibt eine Zeichenkette zurück, die die tatsächlich empfangenen Daten enthält. *representation* gibt die Darstellung des *Ergebnisses* an: "A" = ASCII (Standard), "H" = HEX, "D" = Dezimal oder "B" = Binär. Die zurückgegebenen Daten enthalten keine Platzhalter. An Platzhalterpositionen werden die tatsächlich empfangenen Zeichen zurückgegeben. HINWEIS: Wenn die ursprüngliche Empfangssequenz '#' Platzhalter (null oder ein Zeichen) enthält, kann die Länge der DL.OnReceive _GetData()-Sequenz kürzer sein als die ursprüngliche Sequenz mit Platzhaltern. Syntax 2: Gibt eine Zeichenkette zurück, die eine bestimmte Anzahl von Zeichen aus den empfangenen Daten enthält. *Start:* Bereich: 1 .. **DL.OnReceive_GetSize()**, oder - 1 = Start beim letzten Zeichen, -2 = Start beim vorletzten Zeichen, ... Standardwert ist 1. *length:* Anzahl der Zeichen, oder -1 = bis zum letzten Zeichen, -2 = bis zum vorletzten Zeichen, ... Der Standardwert ist -1.
Ergebnis = **DL.OnReceive_GetChannel()**	Gibt die Nummer des Kommunikationskanals zurück, auf dem diese Sequenz erkannt wurde. Im Kommunikationsmodus "Überwachung" ist der Rückgabewert 1 oder 2. Im Kommunikationsmodus "Senden/Empfangen" ist der Rückgabewert immer 2 (RX-Kanal).
Ergebnis = **DL.OnReceive_Peek(** *charNo* **)** Syntax 2: *Ergebnis* = **DL.OnReceive_Peek(** *charNo*, *Darstellung* **)**	Gibt ein Zeichen der empfangenen Daten als Ganzzahlwert von 0..255 zurück *charNo* ist die Position innerhalb der empfangenen Daten. Gültiger Bereich für *charNo*: 1 .. **DL.OnReceive_GetSize()**, oder -1 = Start beim letzten Zeichen, -2 = Start beim vorletzten Zeichen, ... Syntax 2: Gibt eine Zeichenkette anstelle eines Integer-Wertes zurück. *Die Darstellung* gibt das Format an: "A" = ASCII, "H" = HEX, "D" = Dezimal oder "B" = Dezimal Binär.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

myDateTime = **DL.OnReceive_GetDateTime()** *Millisekunden* = **DL.OnReceive_GetMilliseconds()**	Diese Funktionen geben den tatsächlichen Docklight-Datums-/Zeitstempel zurück, zu dem diese Empfangssequenz ausgelöst wurde. Das Ergebnis wird in zwei separaten VBScript-Standard-Datentypen gespeichert: *myDateTime:* VBScript-Datumswert mit der Datums-/Zeitangabe in 1-Sekunden-Auflösung *milliseconds:* Integer-Wert mit der entsprechenden Millisekunden-Information aus 0..999

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Bemerkungen

Die Methode **DL.OnReceive_GetData()** ist eine gute Möglichkeit, die tatsächlich empfangenen Daten zu analysieren, wenn Sie ASCII-Protokolle nur mit Druckzeichen verwenden. Wenn Sie den HEX- oder Dezimalwert einzelner Zeichen benötigen, können Sie die Funktion **DL.OnReceive_Peek(** .. **)** als praktische Alternative verwenden. Siehe die Ereignisprozedur DL_OnSend() für ein entsprechendes Beispiel.

Die Prozedur **DL_OnReceive()** wird nur ausgeführt, während das Skript läuft. Während der Ausführung des Codes **DL_OnReceive()** werden keine weiteren Kommunikationsverarbeitungen und Anzeigeaktualisierungen durchgeführt. Um Leistungs- und Zeitprobleme zu vermeiden, sollten Sie die Ausführungszeit gering halten. Vermeiden Sie z. B. verschachtelte Schleifen und führen Sie keine zeitaufwändigen Berechnungen durch.

DL_OnReceive()-Prozeduren werden nicht ausgeführt, während eine Pause- oder eine WaitForSequence-Methode den Programmablauf blockiert. Wenn eine Empfangssequenz erkannt wird, wird der DL_OnReceive()-Aufruf in die Warteschlange gestellt und nach der Rückkehr von **Pause** (oder **WaitForSequence**) ausgeführt. Siehe Beispiel 2 unten für eine Abhilfe für dieses Problem.

Siehe auch Timing und Programmablauf für einen Einblick, wie Docklight Empfangsdatenereignisse behandelt und den Codeabschnitt **DL_OnReceive()** ausführt.

Beispiel

```
' Beispiel DL_OnReceive()
Ereigniscode '
' Vordefinierte
Sendesequenz ' (0)
Sendewert:
' VALUE=<?><?><CR><LF>
'
' Vordefinierte
Empfangssequenz ' (0)
Empfangener Wert:
' VALUE=<?><?><CR><LF>
'
' Führen Sie diesen Test an einem COM-Anschluss mit
einem Loopback-Anschluss durch ' (TX mit RX desselben
Anschlusses verbunden).

finished = False
DL.ClearCommWindows
Do
    DL.Pause 1 ' (die Pause reduziert die CPU-Last im
Leerlauf) Schleife bis zum Ende

Sub DL_OnReceive()
    If DL.OnReceive_GetName() = "Wert erhalten" Then
        DL.AddComment "Wert erhalten = " &
DL.OnReceive_GetData("A", 7, -3)
        ' Lesen Sie den Wert aus den Empfangsdaten, aber nur den
sich ändernden "Wert"-Teil
        myValue = Mid(DL.OnReceive_GetData(), 7, 2)
        ' Sicherstellen, dass dies ein numerischer
        Wert ist
        If IsNumeric(myValue) Then
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
            ' erhöhen
            myValue = myValue + 1
            If myValue < 100 Then
                Wenn der Wert immer noch unter 100 liegt, senden Sie ihn aus.
wieder
            newValueStr = CStr(myValue)
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
                DL.SendSequence "Wert senden", newValueStr
            Else
                DL.AddComment "VALUE=99, stoppt..." finished
                = True
            End If
        End If
    End If
End Sub
```

Nach dem Starten des Skripts und dem manuellen Senden einer "Send Value"-Sequenz mit dem Parameterwert "95" könnte das Kommunikationsfenster wie folgt aussehen:

```
7/29/2012 15:43:43.823 [TX] - WERT=95

7/29/2012 15:43:43.826 [RX] - WERT=95
 Erhaltener Wert = 95

7/29/2012 15:43:43.879 [TX] - VALUE=96

7/29/2012 15:43:43.880 [RX] - WERT=96
 Erhaltener Wert = 96

7/29/2012 15:43:43.926 [TX] - VALUE=97

7/29/2012 15:43:43.927 [RX] - WERT=97
 Erhaltener Wert = 97

7/29/2012 15:43:43.977 [TX] - VALUE=98

7/29/2012 15:43:43.978 [RX] - WERT=98
 Erhaltener Wert = 98

7/29/2012 15:43:44.025 [TX] - WERT=99

7/29/2012 15:43:44.026 [RX] - WERT=99
 Empfangener Wert = 99
 VALUE=99, Anhalten...
```

Beispiel 2

```
' Beispiel für die Verwendung von DL_OnReceive() in Code mit

Pause-Anweisungen ' Vordefinierte Sendesequenz
' (0) Hallo:
' Hallo<CR><LF>
'
' Vordefinierte
Empfangssequenz ' (0) Hallo:
' Hallo<CR><LF>
'
' Führen Sie diesen Test an einem COM-Anschluss mit
einem Loopback-Anschluss durch ' (TX mit RX desselben
Anschlusses verbunden).

DL.ClearCommWindows
' Starten der Kommunikation started =
True
DL.SendSequence "Hallo"
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
' Warten Sie etwa 1 Sekunde, aber stellen Sie sicher, dass
die DL_OnReceive()-Ereignisse
' werden in der Zwischenzeit
verarbeitet pauseWithEvents 1000
' Senden stoppen und warten, bis alle Daten ordnungsgemäß
zurückkommen started = False
DL.Pause 20
' Datendurchsatz?
DL.AddComment
DL.AddComment "Anzahl der erkannten 'Hello'-Sequenzen: " &
DL.GetReceiveCounter("Hallo")

Sub DL_OnReceive()
    Wenn gestartet Dann
        myDate = DL.OnReceive_GetDateTime()
        msec = DL.OnReceive_GetMilliseconds()
        DL.AddComment " Zeitstempel empfangen =
        " &
DL.GetDocklightTimeStamp(myDate, msec)
        ' Senden Sie die gleiche Sequenz, die Sie gerade
empfangen haben
        DL.SendSequence DL.OnReceive_GetIndex()
    End If
End Sub

Sub pauseWithEvents(milliseconds)
    Anders als der Befehl DL.Pause erlaubt diese Funktion
DL_OnReceive()
    ' Anweisungen, die während der Wartezeit
    verarbeitet werden sollen startTime = Timer
    While (Timer - startTime) < milliseconds / 1000
        Mitternachtssprung' / Zurücksetzen der Timer-
Variablen berücksichtigen
        If Timer < (startTime - 1) Then startTime = startTime -
86400
        DL.Pause 1
    Wend
End Sub
```

Nach dem Start des Skripts sendet und empfängt Docklight etwa 1 Sekunde lang die "Hello"-Sequenz. Die Gesamtzahl der gesendeten und empfangenen Sequenzen hängt von den COM-Port-Einstellungen (Baudrate), der PC-Geschwindigkeit und den Docklight-Anzeigeeinstellungen ab. Das Kommunikationsfenster könnte wie folgt aussehen:

```
8/1/2012 11:00:41.830 [TX] - Hallo<CR><LF>

8/1/2012 11:00:41.834 [RX] - Hallo<CR><LF>
  Zeitstempel des Empfangs = 8/1/2012

11:00:41.834 8/1/2012 11:00:41.846 [TX] -

Hallo<CR><LF>

8/1/2012 11:00:41.849 [RX] - Hallo<CR><LF>
  Zeitstempel des Empfangs = 8/1/2012

11:00:41.849 8/1/2012 11:00:41.861 [TX] -

Hallo<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
...
8/1/2012 11:00:42.825 [TX] - Hallo<CR><LF>
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
8/1/2012 11:00:42.827 [RX] - Hallo<CR><LF>
  Zeitstempel empfangen = 8/1/2012 11:00:42.827

8/1/2012 11:00:42.839 [TX] - Hallo<CR><LF>

8/1/2012 11:00:42.841 [RX] - Hallo<CR><LF>
  Zeitstempel des Empfangs = 8/1/2012

11:00:42.841 8/1/2012 11:00:42.852 [TX] -

Hallo<CR><LF>

8/1/2012 11:00:42.855 [RX] - Hallo<CR><LF>

Anzahl der erkannten "Hello"-Sequenzen: 70
```

Beispiel 3

```
' Beispiel mit Sub DL_OnReceive(), um auf JEDE gefundene

Sequenz zu warten = False
foundName = ""
foundDate =
Jetzt foundMSec
= 0

Do
    DL.Pause 1 ' (die Pause reduziert die CPU-Last im Leerlauf)
Schleife bis gefunden

DL.AddComment
DL.AddComment "Sequenz empfangen: " & foundName
DL.AddComment "Datum/Zeit empfangen: " &
DL.GetDocklightTimeStamp(foundDate, foundMSec)

Sub DL_OnReceive()
    If Not found Then
        found = True
        foundName = DL.OnReceive_GetName()
        foundDate = DL.OnReceive_GetDateTime()
        foundMSec = DL.OnReceive_GetMilliseconds()
    End If
End Sub
```

10.3.3 OnSend / OnReceive - Zeitablauf und Programmfluss

Sub DL_OnSend() Timing

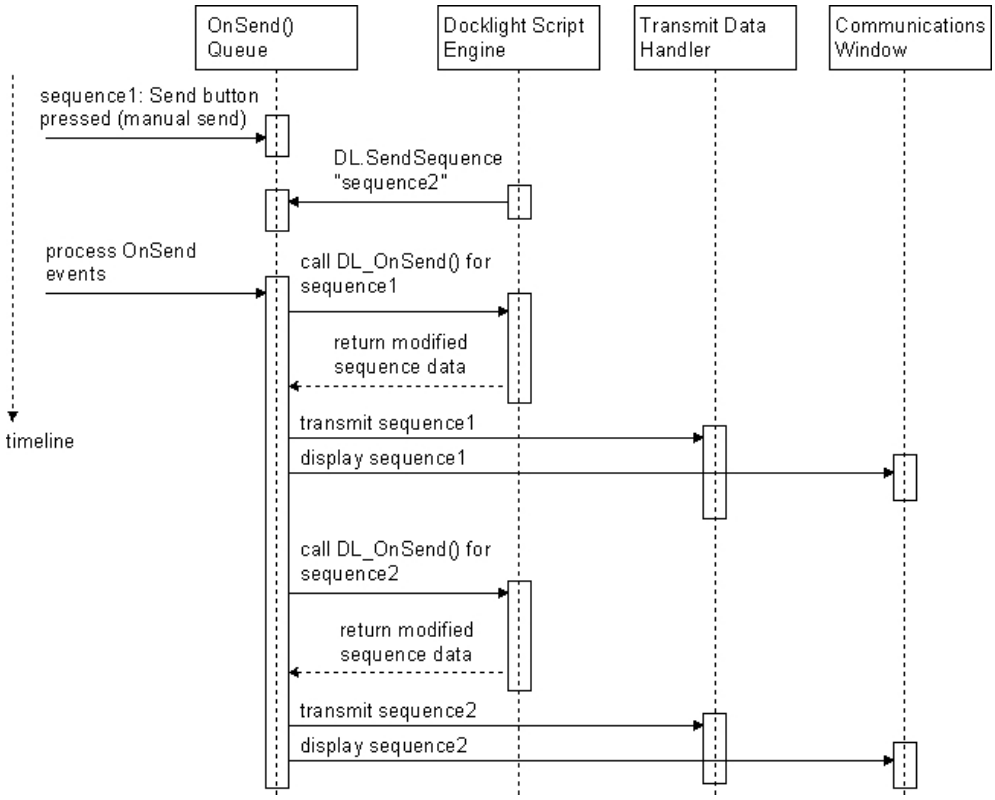
Während ein Skript läuft, wird die Ereignisprozedur DL_OnSend() einmal für jede neue Sendesequenz ausgeführt. Dies gilt sowohl für Sequenzen, die durch Klicken auf die Schaltfläche "Senden" gesendet werden, als auch für DL.SendSequence-Aufrufe.

Die Ereignisprozedur **DL_OnSend()** wird erst eingegeben, nachdem die aktuelle Skriptcodezeile ausgeführt wurde. In der Zwischenzeit werden "Sende"-Anfragen gepuffert.

Das folgende Sequenzdiagramm zeigt das resultierende Zeitverhalten für ein Beispiel mit einem "manuellen" Sendeauftrag (sequence1) und einer zweiten, durch Skriptcode ausgelösten Sendesequenz (DL.SendSequence "sequence2").

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

HINWEIS: Parameterplatzhalter und Prüfsummenfelder werden verarbeitet, bevor die Sequenzdaten in die OnSend()-Warteschlange gelangen.



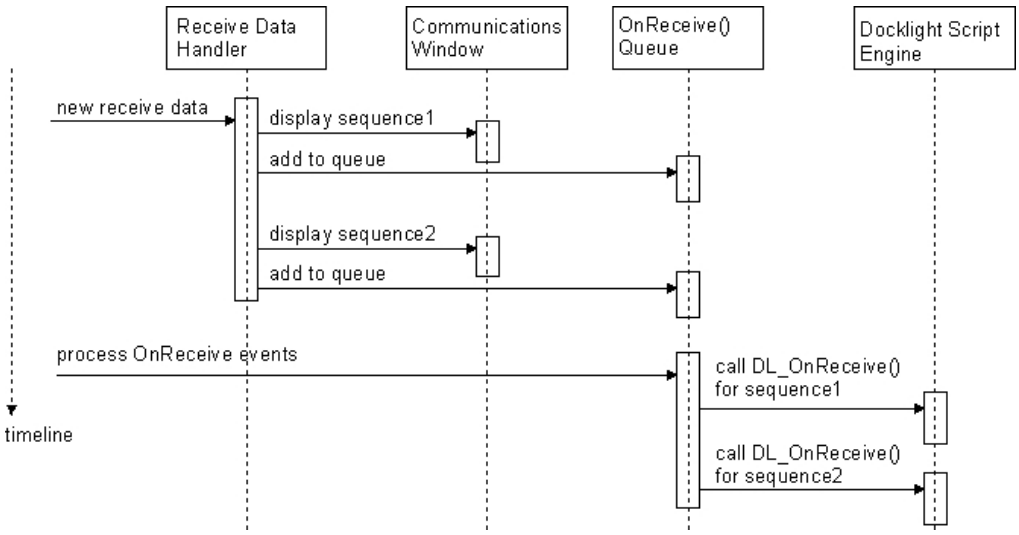
Sub DL_OnReceive() Timing

Ähnlich wie **DL_OnSend()** wird die Ereignisprozedur DL_OnReceive() nicht sofort ausgeführt, nachdem Docklight eine neue Übereinstimmung der Empfangssequenz festgestellt hat. Stattdessen werden die Ereignisse gepuffert und ausgeführt, nachdem die aktuelle Skriptcodezeile ausgeführt wurde.

Das nachstehende Sequenzdiagramm zeigt das Timing für ein Beispiel, bei dem zwei verschiedene Empfangssequenzen auf einmal erkannt werden und der Code **DL_OnReceive() zu einem** späteren Zeitpunkt ausgeführt wird.

HINWEIS: Prüfsummenfelder werden im Receive Data Handler verarbeitet, bevor die Sequenzdaten in die OnReceive()-Warteschlange gelangen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de



10.4 FileInput- / FileOutput-Objekte zum Lesen und Schreiben von Dateien

Docklight Scripting bietet zusätzliche Objekte, mit denen Sie eine Datei mit Text- oder Binärdaten lesen oder eine eigene Ausgabedatei erstellen können.

Objekt Name	Beschreibung
FileInput	Vorhandene Dateien für die sequentielle Eingabe öffnen, wobei die Datei entweder Zeichen für Zeichen oder Zeile für Zeile gelesen wird. Siehe Dateien lesen.
FileOutput	Erstellen Sie eine neue Datei oder fügen Sie Daten an eine bestehende Datei an. Es können sowohl binäre Daten als auch Textdateien erstellt werden. Siehe Schreiben Dateien.

10.4.1 FileInput - Dateien lesen

Das globale FileInput-Objekt bietet eine einfache Schnittstelle, um vorhandene Dateien zu bearbeiten, z.B. um sie mit zusätzlichen Prüfsummen und Formatierungen über die serielle Leitung zu übertragen.

Verfügbare Methoden und Eigenschaften für **FileInput**:

Methode / Eigenschaft	Beschreibung
FileInput.OpenFile *filePathName* **[,** *rawData***]**	Öffnet eine vorhandene Datei zur Eingabe. *rawData* = False (Standard): Als Textdatei öffnen. *rawData* = True: Öffnen als binäre Rohdaten-Datei.
FileInput.CloseFile	Schließt die Datei.
Ergebnis = **FileInput.GetLine()**	Gibt eine Zeichenkette mit der nächsten Textzeile zurück. Das *Ergebnis* enthält nicht die Zeilenumbruchzeichen (CR / LF). Die Methode **GetLine** kann nur für Text verwendet werden Dateien (*rawData* = False).
Ergebnis = **FileInput.GetByte()**	Gibt das nächste Byte zurück.
Ergebnis = **FileInput.IsOpen**	Gibt True zurück, wenn eine Datei geöffnet ist, False, wenn nicht.
Ergebnis = **FileInput.EndOfFile**	Gibt True zurück, wenn alle Daten gelesen wurden und die End-of-File-Marke erreicht wurde.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

result = **FileInput.Dialog([***caption***,] [,** *defaultPath***] [,** *fileFilter***])**	Zeigt einen "Datei öffnen"-Dialog an und gibt den gewählten Dateipfad oder eine leere Zeichenkette zurück, wenn der Vorgang abgebrochen wird. Der Standardwert für *fileFilter* ist:

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	Alle Dateien (*.*)|*.*
Ergebnis = **FileInput.FileExists**(*filePath*)	Gibt True zurück, wenn *filePath* existiert.

Bemerkungen

Wenn eine Datei nicht geöffnet werden kann, erzeugt **FileInput.OpenFile** einen Fehler und die Skriptausführung stoppt. Verwenden Sie **FileInput.FileExists**, um zu prüfen, ob die Datei verfügbar ist.

Siehe auch das FileOutput-Objekt.

Beispiel

```
' FileInput / FileOutput Beispiel

DL.ClearCommWindows

' Erstellen Sie eine einfache Textdatei
FileOutput.CreateFile "C:\test.txt"
FileOutput.WriteLine "Hello World!"
FileOutput.WriteLine "Auf Wiedersehen,
Welt!"
FileOutput.CloseFile

Öffnen Sie die Datei und drucken Sie ihren Inhalt aus.
path = FileInput.Dialog("Einen Text für Docklight öffnen",
"C:\", "Textdateien (*.txt)|*.txt")
Wenn Len(Pfad) > 0 Dann
    DL.AddComment "Textdatei lesen..."
    FileInput.OpenFile Pfad
    Do Until FileInput.EndOfFile
        DL.AddComment FileInput.GetLine()
    Schleife
    FileInput.CloseFile
Ende wenn

' Versuchen Sie nun eine Rohdaten-Datei
FileOutput.CreateFile "C:\test.bin", True
For i = 0 To 255
    FileOutput.WriteByte i
Nächste
FileOutput.CloseFile

' Und laden Sie es...
DL.AddComment
DL.AddComment "Rohdaten-Datei lesen..."
FileInput.OpenFile "C:\test.bin", True
Do Until FileInput.EndOfFile
    DL.AddComment Right("0" + Hex(FileInput.GetByte()), 2) + "
", False, False
Schleife
FileInput.CloseFile
```

Der obige Skriptcode erzeugt die folgende Ausgabe im Docklight-Kommunikationsfenster:

```
Textdatei C:\test.txt erstellen
Textdatei lesen: C:\test.txt
Hallo Welt!
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
Auf Wiedersehen, Welt!
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
 Binäre Datendatei C:\test.txt

 erstellen Rohdatendatei lesen...
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14
15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29
2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E
3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53
54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67 68
69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D
7E 7F 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92
93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7
A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC
BD BE BF C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1
D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6
E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB
FC FD FE FF
```

10.4.2 FileOutput - Dateien schreiben

Das globale FileOutput-Objekt bietet eine einfache Schnittstelle zum Erstellen von Dateien, z. B. zum Schreiben von benutzerdefinierten Datenformaten.

Verfügbare Methoden und Eigenschaften für **FileOutput**:

Methode / Eigenschaft	Beschreibung
FileOutput.CreateFile *filePathName* **[,** *rawData***] [,** *appendData***]**	Erstellen Sie eine neue Ausgabedatei. *rawData* = False (Voreinstellung): Als Textdatei öffnen. *rawData* = True: Als binäre Rohdaten-Datei öffnen. *appendData* = False (Voreinstellung): Überschreibt die Datei, falls vorhanden. *appendData* = True: Daten an eine bestehende Datei anhängen.
FileOutput.CloseFile	Schließt die Datei.
FileOutput.WriteLine *Daten* **[,** *appendLineBreak***]**	Schreiben der *Zeichenkettendaten* in die Datei. *appendLineBreak* = True (Standard): Einen CR/LF-Zeilenumbruch nach dem Datenstring anhängen *appendLineBreak* = False: keinen Zeilenumbruch erzeugen Die Methode **WriteLine** kann nur für Text verwendet werden Dateien (*rawData* = False).
FileOutput.WriteByte *Daten*	Schreiben Sie die *Byte-Daten* in die Datei.
Ergebnis = **FileOutput.IsOpen**	Gibt True zurück, wenn eine Datei geöffnet ist, False, wenn nicht.
result = **FileOutput.Dialog([***caption***] [,** *defaultPath***] [,** *fileFilter***])**	Zeigt einen "Datei speichern"-Dialog an und gibt den gewählten Dateipfad oder eine leere Zeichenkette zurück, wenn der Vorgang abgebrochen wird. Der Standardwert für *fileFilter* ist: `Alle Dateien (*.*)|*.*`
Ergebnis = **FileOutput.FileExists(***filePath***)**	Gibt True zurück, wenn *filePath* existiert.

Bemerkungen

Wenn **FileInput.CreateFile** die Datei nicht öffnen kann, wird ein Fehler ausgegeben und die Ausführung des Skripts angehalten.

Siehe das FileInput-Objekt für ein Beispiel.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

10.4.3 Mehrere Eingabedateien / mehrere Ausgabedateien

Wenn Sie mehr als eine Datei gleichzeitig lesen oder schreiben müssen, können Sie bis zu 4 Eingabedateien und 4 Ausgabedateien gleichzeitig öffnen, indem Sie neben **FileInput** und **FileOutput** weitere globale Objekte verwenden. Die Liste der verfügbaren Objekte lautet:

Objekt Name	Beschreibung
DateiEingabe **DateiEingabe2** **DateiEingabe3** **DateiEingabe4**	Öffnen Sie bis zu 4 verschiedene Dateien zum Lesen. Siehe Dateien lesen.
DateiAusgang **DateiAusgang2** **DateiAusgang3** **DateiAusgang4**	Öffnen Sie bis zu 4 verschiedene Dateien zum Schreiben. Siehe Dateien schreiben.

Beispiel

```
' Ausgabe mehrerer

Dateien ' Erstellen

von 4 Textdateien
DL.AddComment "4 Textdateien gleichzeitig
schreiben..." FileOutput.CreateFile "file1.txt"
FileOutput2.CreateFile "file2.txt"
FileOutput3.CreateFile "file3.txt"
FileOutput4.CreateFile "file4.txt"
' Gleichzeitig schreiben
For i = 1 To 10
    FileOutput.WriteLine "Datei 1: Textzeile " & CStr(i)
    FileOutput2.WriteLine "Datei 2: Textzeile " & CStr(i)
    FileOutput3.WriteLine "Datei 3: Textzeile " & CStr(i)
    FileOutput4.WriteLine "Datei 4: Textzeile " & CStr(i)
Weiter
' Alle 4 Dateien
schließen
FileOutput.CloseFile
FileOutput2.CloseFile
FileOutput3.CloseFile
FileOutput4.CloseFile
DL.AddComment "Done!"
```

10.5 Seitenkanäle - Verwendung mehrerer Datenverbindungen

Docklight Scripting bietet eine Reihe von fortgeschrittenen Skriptmethoden für einfache Mehrkanalanwendungen. Es erlaubt Ihnen, bis zu 8 sekundäre Datenverbindungen zum Senden und Empfangen von Daten zu erstellen, zusätzlich zu den Hauptkommunikationskanälen von Docklight.

Die zusätzlichen / sekundären Datenverbindungen werden als Seitenkanäle bezeichnet. Sie werden über die Methoden OpenSideChannel / CloseSideChannel gesteuert. Um

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Daten auf einem Seitenkanal zu übertragen, wird die Methode DirectSend verwendet.

10.5.1 OpenSideChannel / CloseSideChannel - Verwaltung mehrerer Kanäle

Die Methoden **OpenSideChannel / CloseSideChannel** ermöglichen die Verwendung mehrerer zusätzlicher Datenverbindungen in einer Docklight Scripting-Instanz. Eingehende Daten aus Seitenkanälen können mit dem Argument *rxChannelTag* unterschieden / gekennzeichnet werden Das Senden von Daten auf Seitenkanälen ist über die Methode DirectSend möglich.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

DL Methoden für die Verwaltung von Seitenkanälen/Mehrkanälen:

Methode	Beschreibung
Ergebnis = **DL.OpenSideChannel(** *newSettings* **[,** *channelNo***] [,** *rxChannelTag***])**	Öffnen Sie einen Seitenkommunikationskanal unter Verwendung der Kommunikationskanaleinstellungen aus *newSettings*. *channelNo* = 3 (Standard) oder 4-10 *rxChannelTag* = "Kanal3" (Voreinstellung): Die Empfangsdaten dieses Kanals werden wie folgt gekennzeichnet: `<Kanal3>`*... Daten auf diesem Seitenkanal empfangen...* `</Kanal3>` *rxChannelTag* = "norx": Die empfangenen Kommunikationsdaten werden überhaupt nicht angezeigt. *rxChannelTag* = "" (leere Zeichenkette): Es werden keine Kanal-Tags verwendet und Sie können die eingehenden Daten nicht von "normalen" Docklight RX-Daten unterscheiden. *result* = True: Der Kanal wurde erfolgreich geöffnet. *Ergebnis* = Falsch: Der Kanal konnte nicht geöffnet werden, z. B. weil die Einstellungen ungültig oder der COM-Port nicht verfügbar ist.
DL.CloseSideChannel [*channelNo***]**	Schließt den Seitenkanal.

Bemerkungen

Die Seitenkanäle hängen vom Status der Hauptverbindung ab und umgekehrt:
- OpenSideChannel führt automatisch eine StartCommunication aus, falls erforderlich.
- CloseSideChannel schließt nur den angegebenen Seitenkanal. Andere Kommunikationskanäle setzen die Datenübertragung fort.
- StopCommunication schließt die Hauptkommunikationskanäle und alle offenen Nebenkanäle.

Siehe auch StartCommunication / StopCommunication.

Beispiel

```
DL.SetChannelSettings
"LOCALHOST:10001" DL.StartCommunication
DL.OpenSideChannel "SERVER:10001"
DL.ResetReceiveCounter
DL.SendSequence "", "Test", "A"
DL.Pause 1000
DL.AddComment "Stoppen Sie den Channel3. Dies sollte einen
TCP-Client-Verbindungsfehler verursachen..."
DL.CloseSideChannel
DL.Pause 4000
' alle Kanäle schließen
DL.StopCommunication
```

Die Ausgabe des Kommunikationsfensters könnte wie folgt aussehen:

```
02.10.2019 12:44:23.622 [TX] - Test
02.10.2019 12:44:23.634 [RX] - <Channel3>Test</Channel3> [Channel3]

 Stoppen Sie den Channel3. Dies sollte einen TCP-Client-
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Verbindungsfehler verursachen...

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
02.10.2019 12:44:24.179 DOCKLIGHT meldet: Fehler auf Kanal
LOCALHOST:10001:
TCP/IP-Verbindung vom entfernten Computer geschlossen
```

10.5.2 DirectSend

Die Methode **DirectSend** ist eine Alternative zu SendSequence für bestimmte Anwendungen. Die Syntax ist ähnlich der SendSequence-Syntax 2 und wird für die folgenden Zwecke verwendet:

- Senden von Daten über Seitenkanäle - sekundäre Datenverbindungen, die mit OpenSideChannel geöffnet werden.
- Übertragung / Einspeisung zusätzlicher Daten unter Umgehung der OnSend-Datenwarteschlange und ohne Hinzufügung der Ausgabe im Kommunikationsfenster. Siehe **Bemerkungen** und **Beispiel 2** unten für ein praktisches Beispiel.

Syntax

Ergebnis **= DL.DirectSend** *channelNo*, *customSequence* **[**, *representation* **]**

Die Syntax der DirectSend-Methode besteht aus diesen Teilen:

Teil	Beschreibung
channelNo	*channelNo* = 3 - 10: Daten auf einem Seitenkanal senden (siehe OpenSideChannel). *channelNo* = 1: Senden auf Docklight-Kanal 1 (Senden/Empfangen oder Überwachungsmodus) *channelNo* = 2: Senden auf Docklight Kanal 2 (nur im Überwachungsmodus)
benutzerdefinierteSequenc e	Erforderlich. Zeichenfolge, die die zu sendende Sequenz enthält. Die Sequenz wird standardmäßig in ASCII-Darstellung übergeben. Für HEX-, dezimale oder binäre Sequenzdaten verwenden Sie das optionale Darstellungsargument unten beschrieben.
Vertretung	Optional. String-Wert zur Festlegung des Formats für *customSequence*. "A" = ASCII (Standard), "H" = HEX, "D" = Dezimal oder "B" = Binär.

Bemerkungen

Ergebnis ist true, wenn *channelNo* gültig ist und die Daten übertragen werden konnten.

Die Hauptanwendung für **DirectSend** ist die Kombination mit OpenSideChannel. Darüber hinaus kann **DirectSend** nützlich sein, wenn Sie eine Datenverbindung überwachen und zusätzliche Daten in den Datenstrom zwischen den beiden Geräten einspeisen müssen. Sie können den "passiven Überwachungsansatz" in Docklight effektiv in eine "aktive Überwachung" umwandeln, bei der Docklight z.B. zusätzliche Fehlerzustände erzeugen kann, die in der ursprünglichen Kommunikation nicht vorkommen.

HINWEIS: **DirectSend** erzeugt keine Ausgaben im Kommunikationsfenster und verwendet nicht die OnSend-Datenwarteschlange. Es überträgt Ihre Text- oder Binärdaten einfach "wie sie sind".

Beispiel 1

```
' Vorbedingung: Docklight Kommunikationsmodus = Senden/Empfangen
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.StopCommunication
' Eine UDP-Rückleitung für Kanal 1 verwenden
DL.SetChannelSettings "UDP:LOCALHOST:10001", 1
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
DL.StartCommunication DL.OpenSideChannel
"UDP:LOCALHOST:10002"
DL.DirectSend 1, "TX Data on Channel 1" + vbCrLf
DL.DirectSend 3, "TX Data on side channel" + vbCrLf
' und auf die Reaktionen warten
DL.Pause 100
```

Die Ausgabe des Kommunikationsfensters könnte wie folgt aussehen:

```
02.10.2019 11:51:26.749 [RX] - TX Daten auf Kanal 1<CR><LF>
<Kanal3>TX-Daten auf dem Seitenkanal<CR><LF>
</Kanal3>
```

HINWEIS: Die Daten erscheinen nur auf der RX-Datenanzeige. Es wird keine TX-Kommunikationsausgabe erzeugt.

Beispiel 2

```
' Vorbedingung: Docklight Kommunikationsmodus = Überwachung
DL.StopCommunication
DL.SetChannelSettings "UDP:LOCALHOST:10001", 1
DL.SetChannelSettings "UDP:LOCALHOST:10002", 2
DL.StartCommunication
DL.DirectSend 2, "Bounce"
' und lassen Sie das eine Weile so
weitergehen DL.Pause 1000
```

Die Ausgabe des Kommunikationsfensters könnte wie folgt aussehen:

```
02.10.2019 12:03:08.840 [UDP:LOCALHOST:10002] - Bounce
02.10.2019 12:03:08.840 [UDP:LOCALHOST:10001] - Bounce
02.10.2019 12:03:08.841 [UDP:LOCALHOST:10002] - Bounce
02.10.2019 12:03:08.848 [UDP:LOCALHOST:10001] - Bounce
02.10.2019 12:03:08.851 [UDP:LOCALHOST:10002] - Bounce
02.10.2019 12:03:08.864 [UDP:LOCALHOST:10001] - Bounce
```

HINWEIS: Die Daten werden wiederholt zwischen Kanal 1 und Kanal 2 gespiegelt, da wir auf beiden Seiten UDP-Loopbacks verwenden und der Docklight-Überwachungsmodus standardmäßig Data Forwarding verwendet.

10.6 Debug-Objekt/Skript-Debugging

Docklight Scripting bietet zusätzliche Debugging-Funktionen über das Debug-Objekt.

Methode / Eigenschaft	Beschreibung
Debug.Modus = *newValue*	Legt den Skript-Debug-Modus fest: *newValue* = 0: Kein Debugging, alle Debug-Methoden werden ignoriert. *newValue* = 1: Debug-Modus. Die unten beschriebenen Debug-Methoden werden ausgeführt.
Debug.Assert *assertCondition*	Bricht die Skriptausführung ab, wenn *assertCondition* False ist. Die Skriptausführung kann manuell fortgesetzt werden mit die Symbolleiste **Skript fortsetzen**.
Debug.Break	Bricht die Skriptausführung bedingungslos ab.
Debug.PrintMsg *debugMsg*	Fügt der Anzeige des Kommunikationsfensters einen zusätzlichen Debug-Text hinzu, einschließlich eines Datums-/Zeitstempels und der

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

	aktuelle Zeile des Skriptcodes.

Bemerkungen

Die Methoden **PrintMsg** und **Assert** sind sehr nützlich, um Variablenwerte an verschiedenen Punkten der Ausführung zu drucken und zu überwachen.

Damit die Debug-Methoden eine Wirkung haben, müssen Sie zunächst den Debug-Modus aktivieren, indem Sie die Eigenschaft **Modus** auf eins setzen:

```
Debug.Modus = 1
```

Beispiel

```
' Beispiel Debug-Objekt

Debug.Mode = 1

Count = 0
Do
    Anzahl = Anzahl + 1
    ' einige Debug-Informationen ausgeben: den Wert der Variablen
count
    Debug.PrintMsg "Anzahl = " & Anzahl
    ' Skriptausführung bei Erreichen von 5
    unterbrechen Debug.Assert (Count <> 5)
Schleife bis Anzahl = 10

' jetzt das Gleiche mit Debug-Modus 'off' - Debug-Methoden
haben keine Wirkung

Debug.Modus = 0
Debug.PrintMsg "dies wird nie gedruckt"
Debug.Break ' dies wird nie ausgeführt

DL.AddComment "Debug-Test beendet"
```

Nach der Ausführung dieses Skripts könnte das Kommunikationsfenster wie folgt aussehen:

```
07.04.2009 15:45:06.078 Zeile #9 Debug: Anzahl = 1

07.04.2009 15:45:06.100 Zeile #9 Debug: Anzahl = 2

07.04.2009 15:45:06.119 Zeile #9 Debug: Anzahl = 3

07.04.2009 15:45:06.131 Zeile #9 Debug: Anzahl = 4

07.04.2009 15:45:06.145 Zeile #9 Debug: Anzahl = 5

07.04.2009 15:45:06.158 Zeile #11 Debug: Assert ist Falsch
```

(hier fährt der Benutzer manuell mit der Schaltfläche **Skript fortsetzen** fort)

```
07.04.2009 15:45:07.781 Zeile #9 Debug: Anzahl = 6

07.04.2009 15:45:07.805 Zeile #9 Debug: Anzahl = 7

07.04.2009 15:45:07.830 Zeile #9 Debug: Anzahl = 8

07.04.2009 15:45:07.853 Zeile #9 Debug: Anzahl = 9
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

```
07.04.2009 15:45:07.881 Zeile #9 Debug: Anzahl = 10
 Debug-Test beendet
```

10.7 #include Richtlinie

Weist den Präprozessor des Docklight-Skripts an, den Inhalt der angegebenen Datei an der Stelle einzufügen, an der die #include-Direktive erscheint.

Syntax

#include *filePathName*

Die #include-Syntax besteht aus diesen Teilen:

Teil	Beschreibung
filePathName	Erforderlich. Zeichenfolge, die den Dateipfad (Verzeichnis und Dateiname) der einzubindenden Docklight-Skriptdatei (.pts-Datei) enthält. Die Dateierweiterung .pts kann weggelassen werden. Wenn kein Verzeichnis angegeben wird, verwendet Docklight das aktuelle Arbeitsverzeichnis Verzeichnis.

Bemerkungen

Wenn *filePathName* keine gültige Docklight-Skriptdatei ist oder nicht existiert, meldet Docklight einen Fehler und das Skript wird nicht gestartet.

Die #include-Direktive weist den Präprozessor an, den Inhalt einer angegebenen Datei so zu behandeln, als ob dieser Inhalt im Quellprogramm an der Stelle, an der die Direktive erscheint, enthalten wäre.

Sie können Konstantendeklarationen und Funktionsdefinitionen in Include-Dateien organisieren und dann #include-Direktiven verwenden, um diese Definitionen zu jedem Skript hinzuzufügen. Include-Dateien sind auch nützlich, um Deklarationen von externen Variablen und komplexen Datentypen einzubinden.

Beispiel

```
Beispiel #include-Anweisung
'
#include "myIncludeFile.pts"
DL.AddComment " Pi = " & conPi
```

Die **Datei myIncludeFile.pts** enthält die folgende Definition:

```
Const conPi = 3.14159265358979
```

Die Ausgabe des Kommunikationsfensters würde dann wie folgt aussehen:

```
  Pi = 3,14159265358979
```

10.8 Befehlszeilen-Syntax

Die Docklight Scripting-Anwendung unterstützt Kommandozeilenargumente, um vordefinierte Projekt- oder Skriptdateien zu laden (und auszuführen). Verwenden Sie die folgende Befehlssyntax:

Docklight_Scripting.exe [-r] [-m] [-i] [*ProjektPfadName.ptp* **]**
[*ScriptPfadName.pts* **]**

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Die Docklight-Befehlszeile besteht aus diesen Teilen:

Teil	Beschreibung
-r	Optionales Argument, das in Kombination mit *scriptPathName.pts* verwendet wird. Führt das Skript sofort aus. Wenn kein Laufzeitfehler oder Benutzerstopp auftritt, wird die Docklight Scripting-Anwendung nach Beendigung der Skriptausführung geschlossen.
-m	Optionales Argument. Minimiert das Fenster der Docklight-Scripting-Anwendung beim Starten.
-i	Optionales Argument. Unsichtbare Operation / kein Hauptfenster. Nützlich in Kombination mit der Option **-r** und *scriptPathName.pts.*
ProjektPfadName.ptp	Optional. Lädt die Docklight-Projektdatei *projectPathName.ptp*
SkriptPfadName.pts	Optional. Lädt die Docklight-Skriptdatei *scriptPfadName.pts*

Bemerkungen

Wenn Ihr Skript die Schnittstelle StartLogging oder FileInput / FileOutput verwendet und Sie nur einen Dateinamen, aber keinen vollständigen Verzeichnispfad als Parameter angeben, verwendet Docklight Scripting das aktuelle Skript/Projektverzeichnis.

Beispiel

```
Docklight_Scripting.exe -r C:\myScript.pts
```

Lädt die Docklight-Skriptdatei **C:\myScript.pts** und führt sie aus.

10.9 Dialog: Anpassen / Externer Editor

Menü Scripting > Anpassen / Externer Editor

Externes Programm als Docklight Script Editor

verwenden

Aktivieren Sie diese Option, um den eingebauten Skript-Editor zu deaktivieren und eine externe Editor-Anwendung für diesen Zweck zu starten.

Eine flexible Konfigurationssyntax ermöglicht es Ihnen, mit fast jedem Editor zu arbeiten, der zumindest das Öffnen einer Datei über eine Befehlszeile wie `meinEditor.exe tempScriptFile.vbs`

Kontrolle der Anwendung

Diese Konfigurationsdatei legt fest, wie Docklight Scripting den externen Editor steuert.

Voreinstellung laden für...

Vordefinierte Konfigurationsdateien für drei weit verbreitete Editoren.

TIPP: Wir empfehlen den Editor Notepad++, den Sie unter https://notepad-plus-plus.org/ finden. Das Windows-Notepad-Beispiel dient nur zur Veranschaulichung und erklärt, wie die Konfigurationsdateien funktionieren. Sie können es als Ausgangspunkt für die Einbindung Ihres eigenen Editors verwenden.

So integrieren Sie Ihren eigenen Lieblingseditor

Sie können den Anwendungspfad am Anfang der Konfigurationsdatei festlegen, indem Sie die Option

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

path=-Syntax. Beispielzeile:

```
path=C:\Programmdateien\Notepad++
```


Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Alle folgenden Zeilen der Konfigurationsdatei haben die folgende Syntax:
<Bearbeitungsaktion> <Anwendungssteuerung>

Beispielzeile:
`öffnen: notepad.exe "%DATEI%"`

<Edit Action> kann eine der folgenden Docklight-Bearbeitungsaktionen sein:

Aktion bearbeiten	Beschreibung
`offen:`	Öffnen Sie eine neue Skriptcode-Datei
`goto:`	Sprung zu einer Zeilennummer innerhalb der Skriptdatei
`sparen:`	Speichern der aktuell geöffneten Datei
`schließen:`	Die aktuell geöffnete Datei schließen

<Anwendungssteuerung> kann einer der folgenden Vorgänge sein:

Kontrolle der Anwendung	Beschreibung
`Sendeschlüssel`	Sendet eine oder mehrere Tastenanschläge an den externen Editor. Sie verwendet die gleiche Argumentensyntax wie die SendKeys-Methode des Windows Script Hosts. Einzelheiten finden Sie in der entsprechenden Microsoft-Dokumentation. Beispiel: `goto: sendkeys +^{HOME}{DOWN %LINE%}+{UP}`
`endtask`	Beenden Sie die externe Anwendung. Beispiel: `schließen: endtask`
`aktivieren.`	Aktivieren Sie das Fenster der externen Anwendung. Beispiel: `goto: aktivieren`
`schlafen`	Warten Sie bis zu 500 Millisekunden, um der externen Anwendung etwas mehr Zeit zu geben, die Dinge zu klären. Dies kann notwendig sein, wenn Sie mit der oben beschriebenen Operation `sendkeys:` arbeiten. Beispiel: `öffnen: schlafen 100`
Befehlszeile	Neben den oben genannten Operationen können Sie jede beliebige Windows-Befehlszeile ausführen, z. B. zum Starten Ihres externen Editors. Beispiel: `öffnen: notepad++.exe -nosession -lvb -n%LINE% "%DATEI%"`

Für jede *<Edit Action>* können Sie mehrere Befehlszeilen definieren, z.B.
```
goto: sendkeys +^{HOME}{DOWN %LINE%}+{UP}
goto: activate
```

Die folgenden Platzhalter sind für *<Anwendungssteuerung>* verfügbar

Platzhalter	Beschreibung
`%FILE%`	Pfad zu einer temporären Datei, die den zu bearbeitenden Skriptcode enthält. Docklight Scripting erstellt und verwaltet die temporäre Datei.
`%FILE_UNIX%`	Wie `%FILE%`, verwendet aber ein '/' im UNIX-Stil als Pfadseparator. Dies ist nützlich für einige Open-Source-Editor-Pakete, die Probleme mit dem Windows-Backslash ("\") als Trennzeichen haben.
`%FILE_ESC%`	Wie `%FILE%`, aber mit doppeltem Backslash ('Escape-Sequenz') für das Pfad-Trennzeichen. Dies ist z.B. bei der Arbeit mit dem freien Quellcode-Editor SciTE notwendig.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

`%LINE%`	Die aktuelle Quellcode-Zeilennummer. Diese wird für den `goto` verwendet: Aktion.

Bemerkungen

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Der External Editor Support ist eine flexible und offene Lösung für unsere Benutzer, die mit großen Skriptprojekten arbeiten und lieber mit einem vollwertigen Bearbeitungspaket arbeiten möchten.

Die oben beschriebene Schnittstelle zur Anwendungskontrolle bietet Ihnen Flexibilität, aber wir sind uns der Einschränkungen bei der Steuerung von Drittanbieteranwendungen bewusst, die eigentlich nicht für eine Steuerung von außen gedacht sind.

Wenn Sie eine intelligente Konfigurationsdatei für Ihren persönlichen Lieblingseditor finden oder Probleme mit der oben genannten Schnittstelle haben, würde sich unser Kundendienst freuen, davon zu hören.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Unterst
ützung

11 Unterstützung

11.1 Web-Support und Fehlerbehebung

Aktuelle FAQs und Informationen zur Fehlerbehebung finden Sie auf unseren Online-Support-Seiten, die unter

www.docklight.de/support/

Neuigkeiten über Docklight und Informationen über kostenlose Wartungsupdates finden

Sie unter: www.docklight.de/news.htm

11.2 E-Mail-Unterstützung

Unseren registrierten Kunden bieten wir individuellen E-Mail-Support. Bitte geben Sie bei Ihrer Anfrage Ihre Docklight-Lizenzschlüsselnummer an. Wir werden uns so schnell wie möglich mit Ihnen in Verbindung setzen, um eine Lösung für Ihr Problem zu finden. Senden Sie Ihre Support-Anfrage an

support@docklight.de

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Anhang

12 Anhang

12.1 ASCII-Zeichensatztabellen

Steuerzeichen

Dez	HexASCII	Zeichen.	Bedeutung
0	00	NUL	Null
1	01	SOH	Beginn der Überschrift
2	02	STX	Beginn des Textes
3	03	ETX	Pause/Ende des Textes
4	04	EOT	Ende der Übertragung
5	05	ENQ	Anfrage
6	06	ACK	Positive Rückmeldung
7	07	BEL	Glocke
8	08	BS	Rücktaste
9	09	HT	Horizontale Registerkarte
10	0A	LF	Zeilenvorschub
11	0B	VT	Vertikale Registerkarte
12	0C	FF	Formularvorschub
13	0D	CR	Rückführung des Wagens
14	0E	SO	Herausschieben
15	0F	SI	Shift in/XON (Ausgang fortsetzen)
16	10	DLE	Datenübertragungsfehler
17	11	DC1	XON - Gerätesteuerzeichen 1
18	12	DC2	Gerätesteuerzeichen 2
19	13	DC3	XOFF - Gerätesteuerzeichen 3
20	14	DC4	Gerätesteuerzeichen 4
21	15	NAK	Negative Quittierung
22	16	SYN	Synchroner Leerlauf
23	17	ETB	Ende des Übertragungsblocks
24	18	CAN	Abbrechen
25	19	EM	Ende des Mediums
26	1A	SUB	Ersatz/Ende der Datei
27	1B	ESC	Flucht
28	1C	FS	Datei-Trennzeichen
29	1D	GS	Gruppentrennzeichen
30	1E	RS	Satz-Trennzeichen
31	1F	US	Einheitenabscheider

Zeichen drucken

Dez	HexASCII	Zeichen.	Bedeutung
32	20		Weltr aum
33	21	!	!
34	22	"	"
35	23	#	#
36	24	$	$
37	25	%	%
38	26	&	&
39	27	'	'
40	28	(	(
41	29	)	)

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

42	2A	*	*
43	2B	+	+
44	2C	,	,
45	2D	-	-
46	2E	.	.
47	2F	/	/
48	30	0	Null
49	31	1	Eine
50	32	2	Zwei
51	33	3	Drei
52	34	4	Vier
53	35	5	FÜnf
54	36	6	Sechs
55	37	7	Siebe n
56	38	8	Acht
57	39	9	Neun
58	3A	:	:
59	3B	;	;
60	3C	<	<
61	3D	=	=
62	3E	>	>
63	3F	?	?
64	40	@	@
65	41	A	A
66	42	B	B
67	43	C	C
68	44	D	D
69	45	E	E
70	46	F	F
71	47	G	G
72	48	H	H
73	49	I	I
74	4A	J	J
75	4B	K	K
76	4C	L	L
77	4D	M	M
78	4E	N	N
79	4F	O	O
80	50	P	P
81	51	Q	Q
82	52	R	R
83	53	S	S
84	54	T	T
85	55	U	U
86	56	V	V
87	57	W	W
88	58	X	X
89	59	Y	Y
90	5A	Z	Z
91	5B	[	[
92	5C	\	\
93	5D	]	]
94	5E	^	^
95	5F	_	_
96	60	`	`
97	61	a	a
98	62	b	b

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

99	63	c	c
100	64	d	d
101	65	e	e
102	66	f	f
103	67	g	g
104	68	h	h
105	69	i	i
106	6A	j	j
107	6B	k	k
108	6C	l	l
109	6D	m	m
110	6E	n	n
111	6F	o	o
112	70	p	p
113	71	q	q
114	72	r	r
115	73	s	s
116	74	t	t
117	75	u	u
118	76	v	v
119	77	w	w
120	78	x	x
121	79	y	y
122	7A	z	z
123	7B	{	{
124	7C	|	|
125	7D	}	}
126	7E	~	Tilde
127	7F	DEL	Lösche n

12.2 Tastenkombinationen

Allgemeine Tastenkombinationen

Gilt für

- Kommunikationsfenster (ASCII, HEX, Dezimal, Binär)
- Dialogfeld "Sendesequenz bearbeiten" / "Empfangssequenz bearbeiten
- Bereich Dokumentation

Funktion	Tastenkombination
Kontextspezifische Hilfe	F1
Schnitt	Strg+X
Kopieren	Strg+C
Kleister	Strg+V
Löschen	Del
Alle auswählen	Strg+A

Kontextspezifische Hot Keys

Docklight-Menü

Menü	Funktion	Tastenkombination
Datei	Neues Projekt	Strg+N

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Datei	Offenes Projekt	Strg+O
Datei	Projekt speichern	Strg+S
Datei	Gedruckte Kommunikation	Strg+P
bearbeiten	Sequenz in Comm.Window suchen	Strg+F
Laufen lassen	Start Kommunikation	F5
Laufen lassen	Kommunikation stoppen	F6
Werkzeuge	Start Comm. Protokollierung	F2
Werkzeuge	Stop Comm. Protokollierung	F3
Werkzeuge	Tastatur Konsole Ein	Strg+F5
Werkzeuge	Tastatur Konsole Aus	Strg+F6
Werkzeuge	Dokumentation/Skriptbereich minimieren/wiederherstellen	F12
Werkzeuge	Minimieren/Wiederherstellen von Sequenzlisten	Umschalttaste+F12
Skripting	Skript ausführen	Umschalt+F5
Skripting	Skript anhalten	Umschalt+F6
Skripting	Pause Skript	Umschalt+F7
Skripting	Skript fortsetzen	Umschalt+F8
Skripting	Skript speichern	Strg+T

Fenster Kommunikation

Funktion	Tastenkombination
Eine Sequenz finden	Strg+F
Alle Kommunikationsfenster löschen	Strg+W
Umschalten zwischen ASCII-, HEX-, Dezimal- und Binärdarstellung	Strg+Tab

Liste der Sendesequenzen / Empfangssequenzen

Funktion	Tastenkombination
Diese Sequenz löschen	Del
Diese Sequenz bearbeiten	Strg+E
Diese Sequenz senden *- Nur Sendesequenzen Liste -*	Weltraum

Dialog Sendesequenz bearbeiten / Dialog Empfangssequenz bearbeiten

Funktion	Tastenkombination
Abbrechen	Esc
Platzhalter '?' (passt auf ein Zeichen)	F7
Platzhalter '#' (entspricht einem oder null Zeichen)	F8
Funktion Zeichen '&' (Verzögerung für *x* * 0,01 sec.)	F9
Funktionszeichen '%' - (Unterbrechungszustand)	F10
Funktion Zeichen '!' (Handshake-Signale)	F11

Bereich Dokumentation

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Funktion	Tastenkombination
Standard-Schriftart	`Strg+D`

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

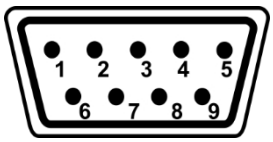
12.3 RS232-Anschlüsse / Pinout

Die gebräuchlichsten Anschlüsse für die RS232-Kommunikation sind

- 9-poliger SUB D9 (EIA/TIA 574-Norm). Eingeführt von IBM und weit verbreitet. Siehe unten.
- 25-poliger SUB D25 (RS232-C). Dies ist der ursprüngliche Stecker, der für den RS232-Standard eingeführt wurde. Er bietet einen sekundären
- Kommunikationskanal.
  8-poliger RJ45 (unterschiedliche Pinbelegungen für Cisco/Yost-Verkabelung, EIA/TIA-561 und andere herstellerspezifische Pinbelegungen).

RS232 SUB D9 (D-Sub DB9) Pinbelegung

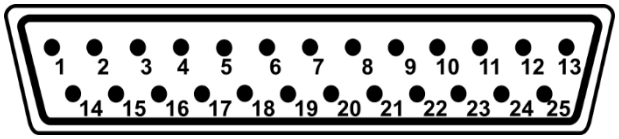
Ansicht: Blick in die Stiftleiste.
Belegung: Aus der Sicht der DEE (die DEE sendet Daten auf der Leitung TX Transmit Data, während die DÜE Daten auf dieser Leitung empfängt)



Stift Nr.	Signal Name	Beschreibung	DTE ein/aus
1	DCD	Datenträgererkennung	Eingabe
2	RX	Empfangen von Daten	Eingabe
3	TX	Daten übermitteln	Ausgabe
4	DTR	Datenendgerät bereit	Ausgabe
5	SGND	Signalmasse	-
6	DSR	Datensatz bereit	Eingabe
7	RTS	Anfrage zum Senden	Ausgabe
8	CTS	Zum Senden freigeben	Eingabe
9	RI	Ring-Indikator	Eingabe

RS232 SUB D25 (D-Sub DB25) Pinbelegung

Ansicht: Blick in die Stiftleiste.
Belegung: Aus der Perspektive einer DTE.



Stift Nr.	Signal Name	Beschreibung
1	-	Schutz-/Schirmerde
2	TX	Daten übermitteln
3	RX	Empfangen von Daten
4	RTS	Anfrage zum Senden
5	CTS	Zum Senden freigeben
6	DSR	Datensatz bereit
7	SGND	Signalmasse

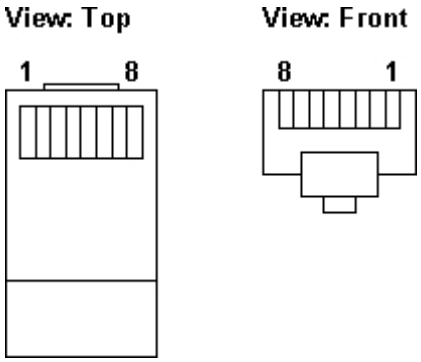
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

8	DCD	Datenträgererkennung

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

9	-	Reserviert
10	-	Reserviert
11	-	Nicht zugewiesen
12	SDCD	Sekundäre Datenträgererkennung
13	SCTS	Sekundäre Freigabe zum Senden
14	STx	Sekundäres Senden von Daten
15	TxCLK	Uhr senden
16	SRx	Sekundär empfangene Daten
17	RxCLK	Empfangsuhr
18	LL	Lokales Loopback
19	SRTS	Sekundäre Anfrage zum Senden
20	DTR	Datenendgerät bereit
21	RL/SQ	Remote Loopback / Signalqualifizierungsdetektor
22	RI	Ring-Indikator
23	CH/CI	Wahlschalter für die Signalrate
24	ACLK	Hilfsuhr
25	-	Nicht zugewiesen

RJ45 8-polige Pinbelegung



Für RJ45-Steckverbinder in der RS232-Kommunikation gibt es mehrere widersprüchliche Pinbelegungen, die auch verwendet werden:

Cisco Konsole / Yost Kabel / Rollover Kabel Anwendungen

Belegung: Aus der Sicht der DEE (die DEE überträgt Daten auf der Leitung TX Transmit Data)

Stift Nr.	Signal Name	Beschreibung
1	CTS	Zum Senden freigeben
2	DCD	Datenträgererkennung
3	RX	Empfangen von Daten
4	SGND	Signalmasse
5	SGND	Signalmasse
6	TX	Daten übermitteln
7	DTR	Datenendgerät bereit
8	RTS	Anfrage zum Senden

HINWEIS: Die Cisco/Yost-Pinbelegung wird bei Kabeln verwendet, die an einem Ende "spiegelbildlich" verdrahtet sind, ähnlich wie ein Nullmodemkabel mit Handshaking.
Jedes Gerät hat den gleichen RJ45

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Buchse und überträgt die Daten auf demselben Pin. Siehe auch den Yost Serial Device Wiring Standard .

EIA/TIA-561-Standard für modulare RJ45/8P8C-Steckverbinder

Stift Nr.	Signal Name	Beschreibung
1	DSR / RI	Datensatzbereitschafts-/Ring-Anzeige
2	DCD	Datenträgererkennung
3	DTR	Datenendgerät bereit
4	SGND	Signalmasse
5	RX	Empfangen von Daten
6	TX	Daten übermitteln
7	CTS	Zum Senden freigeben
8	RTS	Anfrage zum Senden

HINWEIS: Obwohl dies ein offizieller Standard ist, ist es wahrscheinlicher, dass Sie RS232-RJ45-Produkte mit einer anderen Pinbelegung finden, entweder die obige Cisco/Yost-Variante oder herstellerspezifische Pinbelegungen, z. B. MOXA Nport.

12.4 Standard RS232-Kabel

Klassische RS232-Verbindungen

Beim Anschluss von zwei seriellen Geräten müssen je nach den Eigenschaften des seriellen Geräts und der verwendeten Kommunikationsart unterschiedliche Kabeltypen verwendet werden.

Übersicht der RS232 SUB D9 (D-Sub DB9) Verbindungen

Serielles Gerät 1	Serielles Gerät 2	Flusssteuerung (Handshaking)	empfohlenes Kabel
DTE (Datenend gerät) Ausrüstung)	DTE	keine Handshake-Signale	einfaches Nullmodemkabel
DTE	DTE	DTE/DCE kompatible Hardware-Flusskontrolle	Nullmodemkabel mit partiellem Handshaking
	DCE (Daten Kommunikations einrichtungen)	keine Handshake-Signale	einfaches gerades Kabel
DTE	DCE	Hardwarefluss Kontrolle	volles gerades Kabel
DCE	DCE	keine Handshake-Signale	einfaches Nullmodemkabel, jedoch mit SUB-D9-Steckern an beiden Enden
DCE	DCE	Hardware-Flusskontrolle	Nullmodemkabel mit teilweiser Handshaking, aber mit SUB-D9-Steckern an beiden Enden

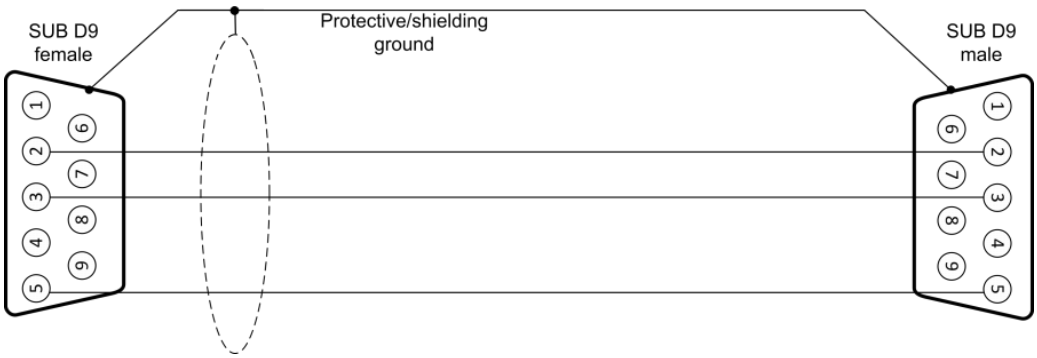
HINWEIS: Eine gute Alternative, um die korrekte Verbindung zwischen verschiedenen DTE- und DCE-Geräten herzustellen, ist die Verwendung des Yost Serial Device Wiring Standards von Dave Yost.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

SUB D9 Einfaches gerades Kabel

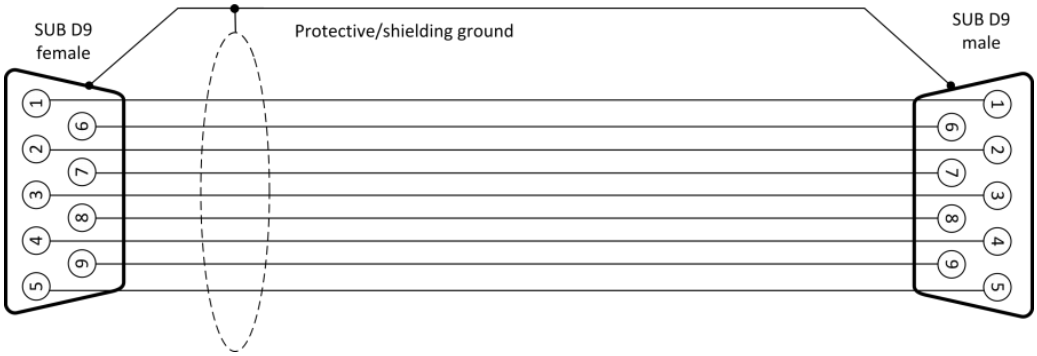
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Anwendungsbereich: DTE-DCE-Kommunikation, bei der keine zusätzlichen Handshake-Signale verwendet werden.



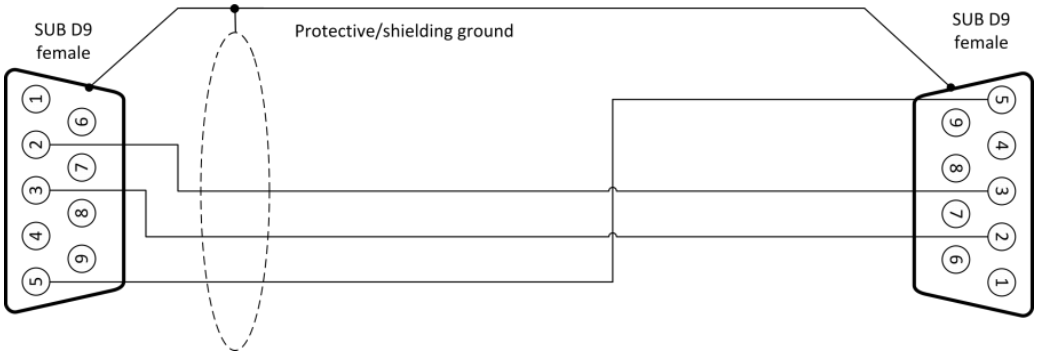
SUB D9 Volles gerades Kabel

Anwendungsbereich: DTE-DCE Kommunikation mit Hardware-Flusskontrolle durch zusätzliche Handshake-Signale.



SUB D9 Einfaches Nullmodemkabel ohne Handshaking

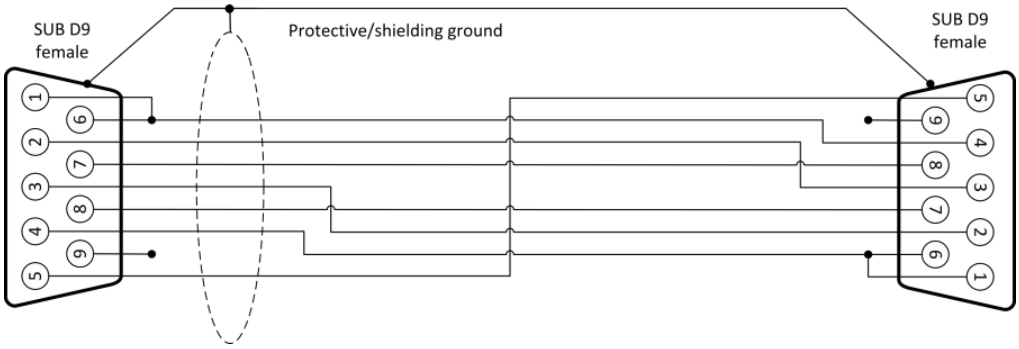
Anwendungsbereich: DTE-DTE-Kommunikation, bei der keine zusätzlichen Handshake-Signale verwendet werden.



SUB D9 Nullmodemkabel mit Full Handshaking

Anwendungsbereich: DTE-DTE Kommunikation mit DTE/DCE kompatibler Hardware-Flusskontrolle. Funktioniert auch, wenn keine Handshake-Signale verwendet werden.

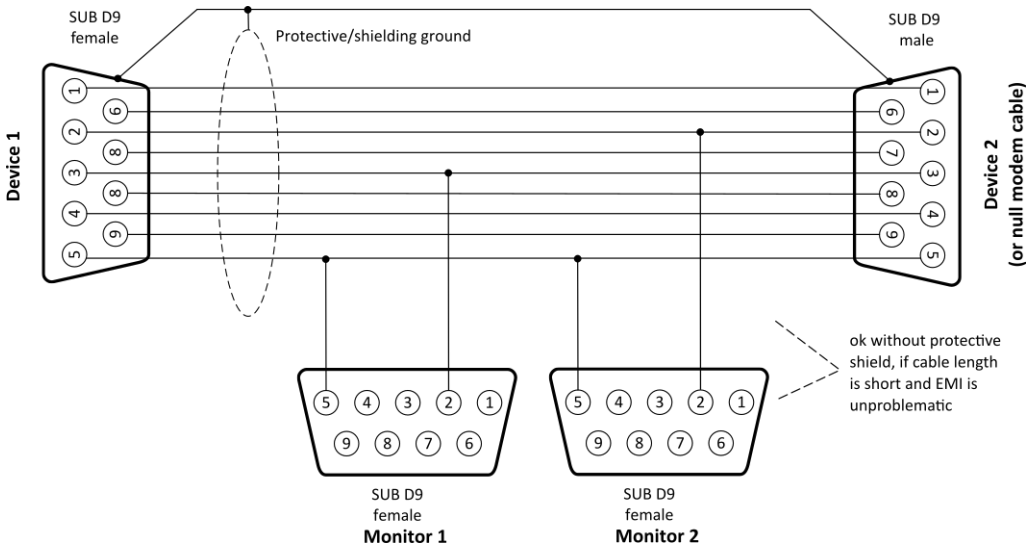
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de



Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

12.5 Docklight Überwachungskabel RS232 SUB D9

Docklight Monitoring Cable ist ein RS232-Vollduplex-Überwachungskabel, das für die Überwachung der seriellen Kommunikation zwischen zwei Geräten konzipiert ist.



Wir bieten ein robustes und vollständig geschirmtes RS232-Überwachungskabel als Zubehör an. Weitere Details finden Sie auf unseren Produktübersichtsseiten und im Datenblatt zum Docklight Monitoring Kabel.

HINWEIS: Unsere Docklight Tap oder Tap Pro / Tap RS485 Datenabgriffe bieten überlegene Überwachungseigenschaften und benötigen keine zwei freien RS232 COM-Ports an Ihrem PC. Nur in seltenen oder älteren Anwendungen ist das Docklight-Überwachungskabel auch heute noch die bevorzugte Wahl.

TIPP: Eine kostengünstige und schnelle Lösung für eine einfache Überwachung ist die Herstellung eines eigenen Überwachungskabels aus einem Flachbandkabel und SUB D9 Schneidklemmverbindern, die in jedem Elektronikfachgeschäft erhältlich sind.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

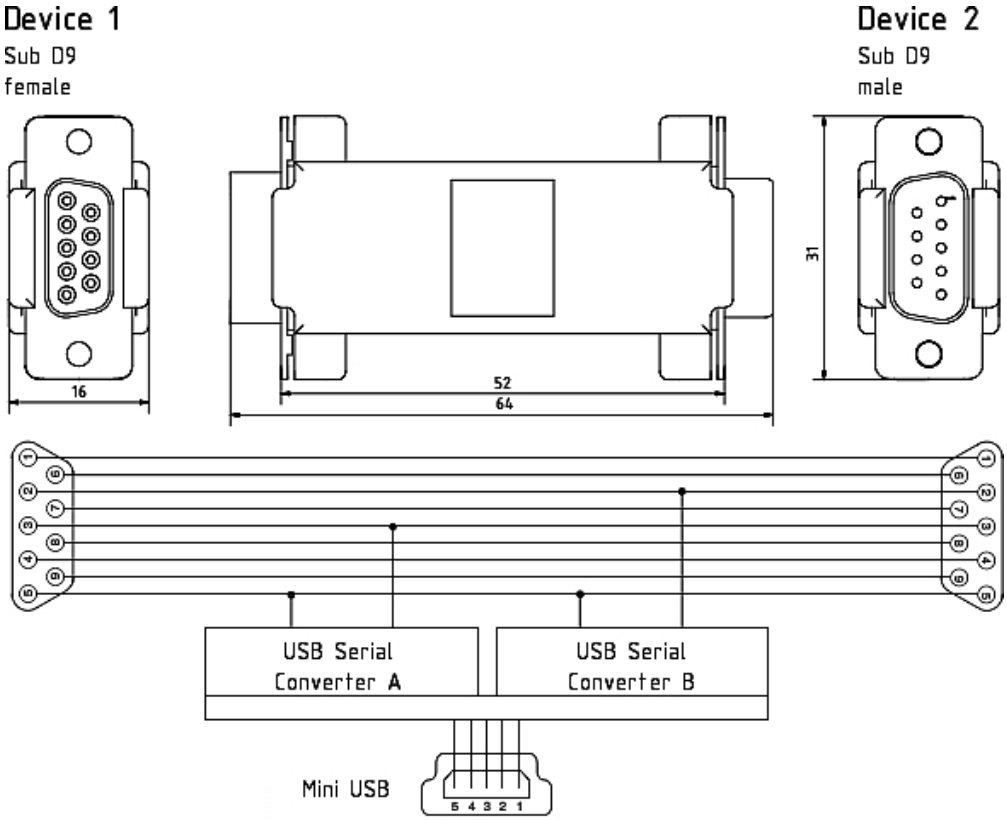
12.6 Docklight Wasserhahn

Docklight Tap ist eine Vollduplex-RS232-Kommunikationsüberwachungslösung für den USB-Anschluss.

Anwendungsbereich: Überwachung der seriellen Kommunikation zwischen zwei Geräten

Docklight hat eine integrierte Unterstützung für das Docklight Tap. Es erkennt den seriellen Dual-Port-USB-Konverter und bietet Hochgeschwindigkeitszugriff auf die Überwachungsdaten mit geringer Latenz. Verwenden Sie den **Docklight-Überwachungsmodus** und die Empfangskanaleinstellungen **TAP0** / **TAP1**. Einzelheiten finden Sie in den Docklight-Projekteinstellungen und in der Anleitung zur Erzielung der besten Timing-Genauigkeit.

Weitere Informationen zum Docklight Tap finden Sie auch auf unseren Produktübersichtsseiten.



Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

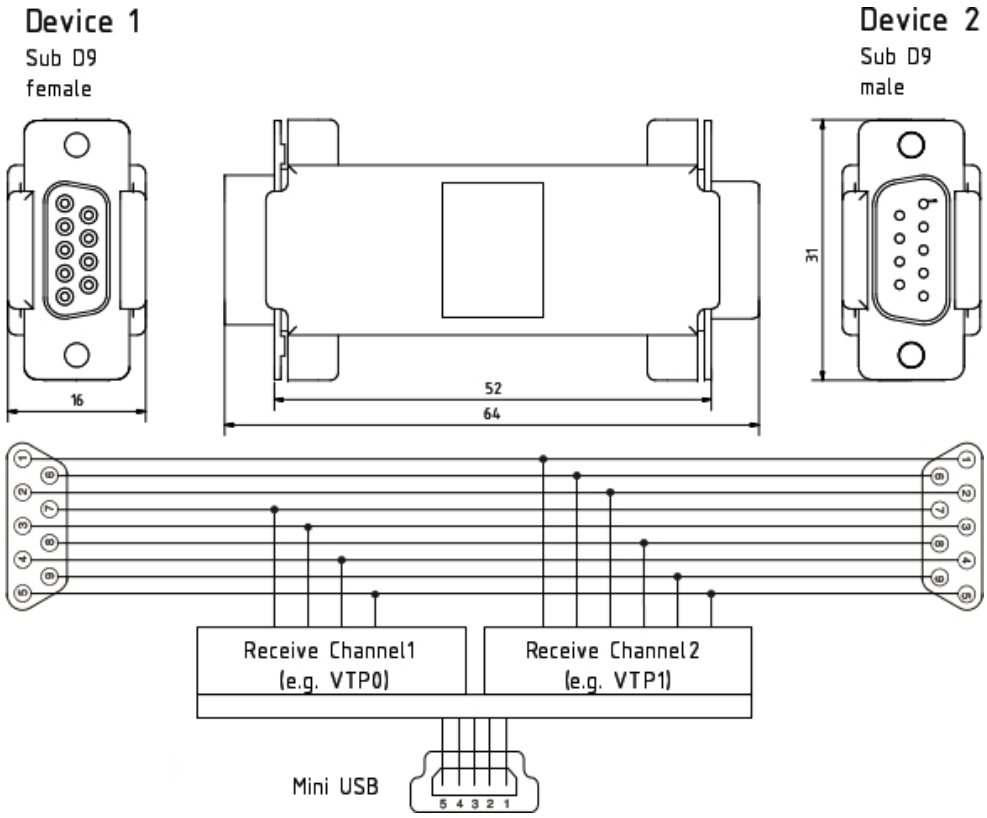
12.7 Docklight Wasserhahn Pro / Wasserhahn 485

Docklight Tap Pro und Docklight Tap 485 sind fortschrittliche, hochauflösende Überwachungslösungen für den USB-Anschluss. Sie ermöglichen echte Zeitmessungen im Millisekundenbereich und die Überwachung von Hochgeschwindigkeitsdatenverbindungen einschließlich RS232-Status/Handshake-Leitungen. Sie werden von Docklight in ähnlicher Weise unterstützt wie das Docklight Tap.

Für die Anwendungen Docklight Tap Pro und Tap 485 verwenden Sie die Einstellungen Docklight **Monitoring Mode** und Receive Channel **VTP0** / **VTP1**. Weitere Details finden Sie in den Docklight Projekteinstellungen.

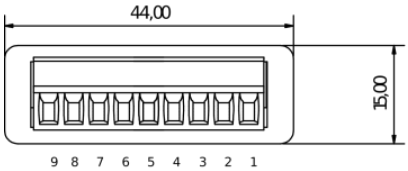
Weitere Informationen zum Docklight Tap Pro und Docklight Tap 485 finden Sie auch auf unseren Produktübersichtsseiten.

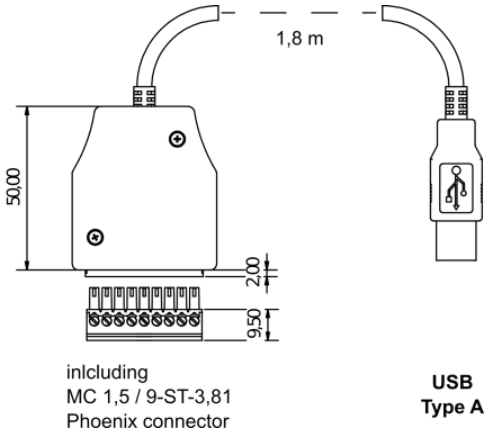
Docklight Tap Pro



Docklight Wasserhahn RS485

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de





Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Glossar / Verwendete Begriffe

13 Glossar / Verwendete Begriffe

13.1 Aktion

Für eine Empfangssequenz kann der Benutzer eine Aktion festlegen, die nach dem Empfang der angegebenen Sequenz ausgeführt wird. Mögliche Aktionen sind

- Senden einer Sendesequenz
  Nur Sendesequenzen ohne Platzhalter können verwendet
- werden Einfügen eines Kommentars
  Ein benutzerdefinierter Text oder ein zusätzlicher Datums-/Zeitstempel wird der Mitteilung hinzugefügt Datenfenster und Protokolldatei
- Auslösen eines Schnappschusses
- Unterbrechung der Kommunikation

13.2 Pause

Ein Unterbrechungszustand bei einer RS232-Verbindung ist dadurch gekennzeichnet, dass die Sendeleitung länger als die maximale Zeichenrahmenlänge einschließlich Start- und Stoppbits auf Space (logisch 0) geht. Einige Anwendungsprotokolle, z. B. LIN, verwenden dies für Synchronisationszwecke.

13.3 Zeichen

Ein Zeichen ist die Grundeinheit der von Docklight verarbeiteten Information. Docklight verwendet immer 8-Bit-Zeichen. Die Kommunikationseinstellungen erlauben jedoch auch eine Datenübertragung mit 7 Bit oder weniger. In diesem Fall wird nur eine Teilmenge der 256 möglichen 8-Bit-Zeichen verwendet, aber die Zeichen werden immer noch in einem 8-Bit-Format gespeichert und verarbeitet.

13.4 CRC

Zyklischer Redundanz-Code. Ein CRC ist eine Methode zur Erkennung, ob eine empfangene Sequenz/Nachricht z. B. durch Übertragungsfehler verfälscht wurde. Dazu wird ein zusätzlicher Prüfsummenwert konstruiert, der von den Nutzdaten der Nachricht abhängt, und dieser Wert dann an die ursprüngliche Nachricht angehängt. Der Empfänger berechnet die Prüfsumme aus den empfangenen Daten und vergleicht sie mit dem übertragenen CRC-Wert, um festzustellen, ob die Nachricht unverändert ist. CRCs werden häufig verwendet, da sie die Erkennung von typischen Übertragungsfehlern (Bitfehler, Burstfehler) mit sehr hoher Genauigkeit ermöglichen.

CRC-Algorithmen basieren auf Polynomarithmetik und sind in vielen verschiedenen Versionen erhältlich. Gängige Algorithmen sind CRC-CCITT, CRC-16 und CRC-32. Ein Beispiel für ein Anwendungsprotokoll, das einen CRC verwendet, ist Modbus over Serial Line.

Ein beliebter Artikel über CRCs ist "CRC Implementation Code in C" von Michael Barr, früher veröffentlicht als "Slow and Steady Never Lost the Race" und "Easier Said Than Done":
https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculation-C-Code

Die CRC-Funktionalität von Docklight Scripting (DL.CalcChecksum) wurde durch den obigen Artikel und die vorgeschlagene Boost-CRC-Bibliothek inspiriert:
http://www.boost.org/libs/crc/index.html

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Nicht zuletzt werden Sie, wenn Sie wirklich von der CRC-Alchemie fasziniert sind, früher oder später auf den folgenden klassischen Artikel aus dem Jahr 1993 stoßen: "A Painless Guide to CRC Error Detection Algorithms" von Ross N. Williams: http://ross.net/crc/crcpaper.html / http://ross.net/crc/download/crc_v3.txt

13.5 DCE

Datenkommunikationsgeräte. Die Begriffe DCE und DTE beziehen sich auf die seriellen Geräte auf beiden Seiten einer RS232-Verbindung. Ein Modem ist ein typisches Beispiel für ein DCE-Gerät. DCEs sind normalerweise mit einer **SUB** D9- oder SUB D25-Buchse ausgestattet. Siehe auch DTE.

13.6 DTE

Datenendgeräte. Die Begriffe DCE und DTE beziehen sich auf die seriellen Geräte auf beiden Seiten einer RS232-Verbindung. Ein PC oder ein Terminal sind Beispiele für ein typisches DTE-Gerät. DTE sind in der Regel mit einem **SUB** D9- oder SUB D25-Stecker ausgestattet. Alle Pinout-Spezifikationen sind aus der DTE-Perspektive geschrieben. Siehe auch DCE.

13.7 Durchflusskontrolle

Die Flusssteuerung bietet einen Mechanismus zur Unterbrechung der Übertragung, wenn ein Gerät beschäftigt ist oder aus irgendeinem Grund nicht weiter kommunizieren kann. Die DEE und die DÜE müssen sich auf den für eine Kommunikationssitzung verwendeten Flusskontrollmechanismus einigen. Es gibt zwei Arten der Flusskontrolle: Hardware und Software.

Hardware-Flusskontrolle
Verwendet Spannungssignale auf den RS232-Statusleitungen RTS / DTR (von der DEE gesetzt) und CTS / DSR (von der DÜE gesetzt), um das Senden und Empfangen von Daten zu steuern. Siehe auch RS232-Pinbelegung.

Software-Flusskontrolle
Verwendet spezielle ASCII-Steuerzeichen (XON / XOFF) zur Steuerung der Datenübertragung. Die Software-Flusskontrolle erfordert textbasierte Kommunikationsdaten oder andere Daten, die keine XON- oder XOFF-Zeichen enthalten.

13.8 HID

HID (Human Interface Device) ist eine Geräteklasse und API, die für USB- und Bluetooth-Geräte verwendet wird.

Docklight Scripting unterstützt den HID-Zugriff über VID / PID (Hersteller-ID / Produkt-ID) oder den vollständigen Windows-USB-Gerätepfad. Docklight Scripting ermöglicht binäre und textorientierte Datenübertragungen über HID Input und Output Reports.

Eine häufige Anwendung (neben der Standard-Windows-Tastatur/Maus-Integration) für die HID-Geräteklasse sind Embedded Devices mit einem eigenen Protokoll zur einfachen Datenübertragung, z.B. Messgeräte oder Sensoren

TIPP: Weitere Informationen zum Zugriff auf HID-Geräte über Docklight Scripting finden Sie auch in unseren Online-Support-Ressourcen unter www.docklight.de/support/.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

HINWEIS: Docklight Scripting unterstützt derzeit nur HID Input/Output Reports. Feature-Reports werden *nicht* unterstützt.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

13.9 LIN

Lokales Verbindungsnetz (Local Interconnect Network). Ein kostengünstiger serieller Kommunikationsbus für verteilte elektronische Systeme in Fahrzeugen, insbesondere für einfache Komponenten wie Türmotoren, Lenkradsteuerungen, Klimasensoren usw. Siehe auch den Wikipedia-Eintrag über LIN .

13.10 Modbus

Modbus ist ein Nachrichtenprotokoll der Anwendungsschicht, das eine Client/Server-Kommunikation zwischen Geräten ermöglicht, die an verschiedene Arten von Bussen oder Netzwerken angeschlossen sind. Es wird üblicherweise als "Modbus over Serial Line" in RS422/485-Netzwerken verwendet, kann aber auch mit TCP over Ethernet implementiert werden ("Modbus TCP").

Es sind zwei verschiedene serielle Übertragungsmodi für Modbus definiert: Der "RTU-Modus" für 8-Bit-Binärübertragungen und der "ASCII-Modus". Der "RTU-Modus" ist der Standardmodus, der von allen Geräten implementiert werden muss.

Eine vollständige Spezifikation des Modbus-Protokolls finden Sie unter www.modbus.org.

13.11 Multidrop-Bus (MDB)

Multidrop-Bus (MDB) ist eine exotischere RS232/RS485-Anwendung, die zum Beispiel in Automatensteuerungen verwendet wird und einen 9-Bit-kompatiblen UART erfordert. Das 9. Datenbit wird für die Auswahl zwischen einem ADDRESS- und einem DATA-Modus verwendet.

Eine Möglichkeit zur Überwachung und Simulation solcher Kommunikationsverbindungen unter Verwendung von Standard-8-Bit-UARTs,
d. h. Standard-RS232-zu-USB-Wandler, ist die Verwendung vorübergehender Paritätsänderungen.

Weitere Informationen und Details finden Sie in Wikipedia über MDB und in der ursprünglichen MDB 3.0-Spezifikation.

13.12 Benanntes Rohr

Eine Named Pipe ist ein Shared-Memory-Mechanismus, der für die Kommunikation zwischen zwei Prozessen auf einem Windows-PC verwendet werden kann.

Docklight Scripting kann eine Client-Verbindung zu einem Named Pipe-Server öffnen und 8-Bit-ASCII- oder Byte-Daten senden oder empfangen.

Einzelheiten zu Named Pipes finden Sie im Windows Development Center.

13.13 Empfangsreihenfolge

Eine Receive Sequence ist eine Sequenz, die von Docklight innerhalb der eingehenden seriellen Daten erkannt werden kann. Eine Receive Sequence wird spezifiziert durch
1. einen eindeutigen Namen (z. B. "Modem Answer OK"),
2. eine Zeichenfolge (z. B. "6F 6B 13 10" im HEX-Format),
3. eine Aktion, die ausgelöst wird, wenn Docklight die definierte Sequenz empfängt.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

13.14 RS232

Die RS232-Norm wird von der EIA/TIA (Electronic Industries Alliance / Telecommunications Industry Associations) definiert. Die Norm definiert einen asynchronen seriellen Datenübertragungsmechanismus sowie die physikalischen und elektrischen Eigenschaften der Schnittstelle.

RS232 verwendet serielle Bitströme, die mit einer vordefinierten Baudrate übertragen werden. Die Informationen werden in Zeichen von 5 bis 8 Bits Länge aufgeteilt. Zusätzliche Start- und Stoppbits werden zur Synchronisierung verwendet, und ein Paritätsbit kann als einfacher Fehlererkennungsmechanismus enthalten sein.

Die elektrische Schnittstelle umfasst unsymmetrische Leitungstreiber, d. h. alle Signale werden durch eine Spannung mit Bezug auf eine gemeinsame Signalmasse dargestellt. RS232 definiert zwei Zustände für die Datensignale: Mark State (oder logische 1) und Space State (oder logische 0). Der Spannungsbereich für die Darstellung dieser Zustände ist wie folgt festgelegt:

Signalzustand	Spannungsbereich des Senders	Spannungsbereich des Empfängers
Mark (logisch 1)	-15V bis -5V	-25V bis -3V
Leerzeichen (logisch 0)	+5V bis +15V	+3V bis +25V
Unbestimmt	-5V bis +5V	-3V bis +3V

Die physikalischen Eigenschaften des RS232-Standards sind im Abschnitt RS232-Anschlüsse / Pinout beschrieben

13.15 RS422

Eine RS422-Kommunikationsverbindung ist eine Vier-Draht-Verbindung mit symmetrischen Leitungstreibern. In einem symmetrischen Differenzialsystem wird ein Signal über zwei Drähte (A und B) übertragen. Der Signalzustand wird durch die Spannung an den beiden Drähten dargestellt. Obwohl eine gemeinsame Signalmasseverbindung erforderlich ist, wird sie nicht zur Bestimmung des Signalzustands am Empfänger verwendet. Dies führt zu einer hohen Immunität gegen EMI (elektromagnetische Störungen) und ermöglicht Kabellängen von über 1000 m, je nach Kabeltyp und Baudrate.

Die EIA-Norm RS422-A "Electrical characteristics of balanced voltage digital interface circuits" definiert die Eigenschaften einer RS422-Schnittstelle.

Die Eigenschaften von Sender und Empfänger gemäß RS422-A sind:

Signalzustand	Transmitter-Differenzspannung V_{AB}	Empfänger-Differenzialspannung V_{AB}
Markierung (oder logisch 1)	-6V bis -2V	-6V bis -200mV
Leerzeichen (oder logische 0)	+2V bis +6V	+200mV bis 6V
Unbestimmt	-2V bis +2V	-200mV bis +200mV

Zulässige Gleichtaktspannung V_{cm} (mittlere Spannung der Klemmen A und B in Bezug auf die Signalmasse): -7V bis +7V

13.16 RS485

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Die RS485-Norm definiert eine symmetrische Zweidraht-Übertragungsleitung, die von bis zu 32 Treiber-Empfänger-Paaren gemeinsam als Busleitung genutzt werden kann. Viele Merkmale der

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Sender und Empfänger sind die gleichen wie bei RS422. Die wichtigsten Unterschiede zwischen RS422 und RS485 sind

- Zwei-Draht-Übertragung (Halbduplex) anstelle von Vier-Draht-Übertragung
- Symmetrische Leitungstreiber mit Tristate-Fähigkeit. Der RS485-Leitungstreiber verfügt über ein zusätzliches "Freigabesignal", das zum Verbinden und Trennen des Treibers mit seinem Ausgangsanschluss verwendet wird. Der Begriff "tristate" bezieht sich auf die drei verschiedenen Zustände, die an der Ausgangsklemme möglich sind: Mark (logisch 1), Space (logisch 0) oder "disconnected".
- Erweiterter Gleichtaktspannungsbereich (V_{cm}) von -7V bis +12V.

Die EIA-Norm RS485 "Standard for electrical characteristics of generators and receivers for use in balanced digital multipoint systems" definiert die Eigenschaften eines RS485-Systems.

13.17 Sequenz senden

Eine Sendesequenz ist eine Sequenz, die von Docklight gesendet werden kann. Eine Sendesequenz wird angegeben durch

1. einen eindeutigen Namen (z. B. "Modemlautstärke einstellen"),
2. eine Zeichenfolge (z. B. "41 54 4C 0D 0A" im HEX-Format).

Es gibt zwei Möglichkeiten, Docklight eine Sequenz senden zu lassen:

- Das Senden einer Sequenz kann manuell durch Drücken der Schaltfläche "Senden" in der Liste "Sendesequenzen" ausgelöst werden (siehe Hauptfenster).
- Das Senden einer Sequenz kann eine mögliche Reaktion sein, wenn Docklight eine bestimmte Empfangssequenz in den eingehenden Daten erkennt (siehe Aktion).

13.18 Sequenz

Eine Sequenz besteht aus einem oder mehreren 8-Bit-Zeichen. Eine Sequenz kann ein beliebiger Teil der seriellen Kommunikation sein, die Sie analysieren. Sie kann aus druckbaren ASCII-Zeichen bestehen, kann aber auch jedes nicht druckbare Zeichen zwischen 0 und 255 dezimal enthalten.
Beispiel:
ATL2 (ASCII-Format)
41 54 4C 0D 0A (HEX-Format)
Diese Sequenz ist ein Modembefehl zum Einstellen der Lautsprecherlautstärke bei AT-kompatiblen Modems. Sie enthält ein Carriage Return (0D) und ein Line Feed (0A) Zeichen am Ende der Zeile.

Die maximale Sequenzgröße in Docklight beträgt 1024 Zeichen.

13.19 Sequenz-Index

Der Sequenzindex ist die Elementnummer einer Sendesequenz innerhalb der Sendesequenzliste bzw. einer Empfangssequenz innerhalb der Empfangssequenzliste. Der Sequenzindex wird in der oberen linken Ecke des Dialogs "Sendesequenz bearbeiten" oder "Empfangssequenz bearbeiten" angezeigt.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

13.20 Serieller Geräteserver

Ein Serial Device Server ist ein Netzwerkgerät, das eine oder mehrere serielle COM-Schnittstellen (RS232, RS422/485) bietet und die seriellen Daten über ein Ethernet-Netzwerk sendet/empfängt. Serial Device Server sind eine gängige Methode, um bestehende Geräte, die über eine serielle Schnittstelle gesteuert werden, aufzurüsten und "netzwerkfähig" zu machen.

13.21 Schnappschuss

Einen Snapshot in Docklight zu erstellen bedeutet, eine Darstellung der seriellen Kommunikation kurz vor und nach dem Erkennen einer Trigger-Sequenz zu erzeugen. Dies ist nützlich, wenn Sie auf einen seltenen Fehler testen, der durch eine bestimmte Sequenz gekennzeichnet ist. Weitere Informationen finden Sie unter Abfangen einer bestimmten Sequenz und Erstellen eines Schnappschusses...

13.22 TCP

Übertragungssteuerungsprotokoll. TCP ist neben UDP das wichtigste Protokoll der Transportschicht, das in IP-Netzen verwendet wird. TCP ist verbindungsorientiert - bevor zwei Netzwerk-Hosts mit TCP kommunizieren können, müssen sie zunächst eine Verbindung aufbauen. TCP ist ein Byte-Stream-Protokoll, das die Zustellung garantiert. TCP stellt sicher, dass die Datenpakete fehlerfrei und in der richtigen Reihenfolge übertragen werden, auch wenn das zugrunde liegende Netz unzuverlässig ist.

TCP verwendet die Portnummern 1-65535, um Anwendungsendpunkte zu identifizieren. Beispiele für bekannte TCP-Anwendungen und Portnummern sind FTP (21), TELNET (23), SMTP (25), HTTP (80) und POP3 (110).

13.23 Auslöser

Ein Trigger ist eine Empfangssequenz mit aktivierter Option "Trigger" (siehe Dialog: Empfangssequenz bearbeiten). Wenn die Schnappschuss-Funktion aktiviert ist, gibt Docklight erst dann eine Ausgabe aus, wenn eine Trigger-Sequenz in den seriellen Kommunikationsdaten erkannt wurde. Weitere Informationen finden Sie unter Abfangen einer bestimmten Sequenz und Erstellen eines Schnappschusses...

13.24 UART

Universal Asynchronous Receiver / Transmitter. Der UART ist die Hardwarekomponente, die die wichtigsten seriellen Kommunikationsaufgaben ausführt:
- Umwandlung von Zeichen in einen seriellen Bitstrom
- Hinzufügen von Start-/Stop-/Paritätsbits und Überprüfung auf Paritätsfehler auf der Empfängerseite
- alle Aufgaben im Zusammenhang mit Timing, Baudraten und Synchronisation

Herkömmliche UARTs sind mit dem 16550A UART kompatibel. Sie enthalten einen 16-Byte-Puffer für eingehende Daten (RX FiFo) und einen 16-Byte-Puffer für ausgehende Daten (TX FiFo). Normalerweise können diese Puffer über den Windows-Gerätemanager und das Öffnen der Eigenschaftsseite für den entsprechenden COM-Anschluss (z. B. COM1) deaktiviert/aktiviert werden.

13.25 UDP

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Benutzer-Datagramm-Protokoll. UDP ist ein Protokoll der Transportschicht, das in IP-Netzen verwendet wird. UDP ist ein verbindungsloses Protokoll - die Kommunikationspartner bauen vor der Datenübertragung keine Verbindung auf. UDP bietet keine zuverlässigen oder geordneten Übertragungen.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Datagramme können in falscher Reihenfolge ankommen, doppelt ankommen oder während der Übertragung verloren gehen. Anwendungen, die eine geordnete, zuverlässige Zustellung von Datenströmen erfordern, sollten stattdessen TCP verwenden.

UDP ist schneller als TCP und hat Vorteile für viele leichtgewichtige oder zeitkritische Netzanwendungen. UDP wird für das Domain Name System im Internet, für Streaming-Media-Anwendungen wie Voice Over IP und für den Rundfunk in IP-Netzen verwendet.

UDP verwendet die Anschlussnummern 1-65535, um Anwendungsendpunkte zu identifizieren. Beispiele für bekannte UDP-Dienste und Portnummern sind DNS (53), TIME (37) und SNMP (161 und 162).

13.26 Virtuelles Nullmodem

Ein virtuelles Nullmodem ist ein PC-Softwaretreiber, der zwei serielle COM-Ports emuliert, die über ein Nullmodemkabel verbunden sind. Wenn eine PC-Anwendung Daten über einen virtuellen COM-Anschluss sendet, kann eine zweite PC-Anwendung diese Daten über den zweiten virtuellen COM-Anschluss empfangen und umgekehrt.

Durch die Verwendung eines virtuellen Nullmodem-Treibers auf Ihrem PC können Sie serielle Datenverbindungen leicht debuggen und simulieren, ohne echte RS232-Anschlüsse und Kabel zu verwenden.

Virtuelle COM-Verbindungen bieten nicht dasselbe Timing wie echte RS232-Verbindungen und emulieren in der Regel nicht die tatsächliche Bit-für-Bit-Übertragung mit einer vordefinierten Baudrate. Jedes Datenpaket, das über den ersten COM-Anschluss gesendet wird, erscheint fast sofort im Empfangspuffer des zweiten COM-Anschlusses. Für die meisten Debugging- und Simulationszwecke kann diese Einschränkung leicht toleriert werden. Einige virtuelle Nullmodem-Treiber bieten einen zusätzlichen Baudraten-Emulationsmodus, bei dem die Datenübertragung verzögert wird, um eine echte RS232-Verbindung und deren begrenzte Übertragungsrate zu emulieren.

Eine Open-Source-Windows-Lösung, die erfolgreich mit Docklight getestet wurde, finden Sie unter
https://sourceforge.net/projects/com0com/
Wir empfehlen die com0com v2.2.2.0 signierte x64 Version, die wir erfolgreich *unter Windows 10* und *Windows 11* getestet haben:
https://sourceforge.net/projects/com0com/files/com0com/2.2.2.0/com0com-2.2.2.0-x64-fre-signed.zip/download

13.27 Platzhalter

Ein Platzhalter ist ein Sonderzeichen, das als Platzhalter innerhalb einer Sequenz dient. Er kann für Empfangssequenzen verwendet werden, wenn Teile der empfangenen Daten nicht spezifiziert sind, z. B. Messwerte, die von einem seriellen Gerät gemeldet werden. Wildcards können auch verwendet werden, um Parameter in einer Sendesequenz zu unterstützen.

Die folgenden Arten von Platzhaltern sind in Docklight verfügbar:
Platzhalter '?' (F7): Passt auf genau ein beliebiges Zeichen (beliebiger ASCII-Code zwischen 0 und 255)
Platzhalter '#' (F8): Entspricht null oder einem Zeichen. Dies ist nützlich für die Unterstützung von Befehlsargumenten variabler Länge (z. B. ein Statuswort) in Sende-/Empfangssequenzen. Beispiele und weitere Informationen finden Sie unter Prüfen auf Sequenzen mit zufälligen Zeichen oder Senden von Befehlen mit

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Parametern.
Andere Platzhalter, die zufällige Daten zulassen:
Funktion Zeichen '!' (F12): Bitweiser Vergleich. Dies ist nützlich, wenn es ein oder mehrere Bits innerhalb eines Zeichens gibt, die auf einen bestimmten Wert getestet werden sollen. Siehe Funktionszeichen '^' (F12) - bitweise Vergleiche für Details und ein Beispiel.

Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de