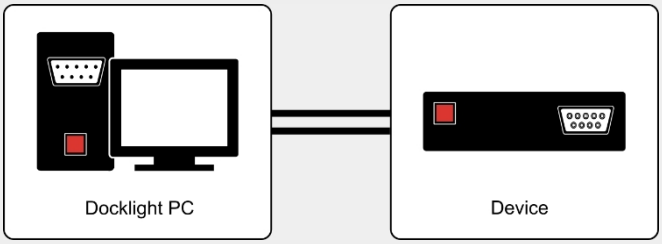
Manuel de l'utilisateur Docklight Scripting V2.4 02/2023



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

1. Droit d'auteur 7

2. Introduction 9

2.1 Docklight - Aperçu 10

2.2 Docklight Scripting - Vue d'ensemble 10

2.3 Applications typiques 11

2.4 Configuration requise 12

3. Interface utilisateur 14

3.1 Fenêtre principale (Scripting) 15

3.2 Presse-papiers - Couper, copier et coller 16

3.3 Domaine de la documentation 16

4. Caractéristiques et fonctions 18

4.1 Comment les données sérielles sont traitées et affichées 19

4.2 Édition et gestion de séquences 19

5. Travailler avec Docklight 21

5.1 Tester un périphérique série ou une implémentation de protocole 22

5.2 Simuler un périphérique série 23

5.3 Surveillance de la communication sérielle entre deux appareils 25

5.4 Interception d'une séquence spécifique et prise d'un instantané de la communication 27

5.5 Consigner et analyser un test 27

5.6 Vérification des séquences contenant des caractères aléatoires (séquence de réception Wildcards) 28

5.7 Enregistrement et chargement des données du projet, script et options 31

6. Travailler avec Docklight (niveau avancé) 33

6.1 Envoi d'instructions avec des paramètres (Wildcards de séquence d'envoi) 34

6.2 Augmentation de la vitesse de traitement et prévention des messages "Input Buffer Overflow" 35

6.3 Comment obtenir la meilleure précision de timing ? 36

6.4 Calcul et validation des sommes de contrôle 36

6.5 Contrôle et surveillance des signaux handshake RS232 38

6.6 Création et détection de retards entre les caractères 42

6.7 Définition et détection d'un état de "rupture". 44

6.8 Tester un périphérique serveur TCP (scripting) 45

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

6.9 Surveillance d'une connexion TCP client/serveur (scripting) 46

7. Exemples et tutoriels 49

7.1 Tester un modem - Projet exemple : ModemDiagnostics.ptp 50
7.2 Réagir à une séquence de réception - Projet exemple : PingPong.ptp 51
7.3 Modbus RTU avec somme de contrôle CRC - Projet exemple : ModbusRtuCrc.ptp 52

8. Exemples et tutoriels (scripting) 55

8.1 Tests automatisés de modems - exemple de script : ModemScript.pts 56
8.2 Démarrage à partir de la ligne de commande - exemple de script : LogStartupScript.pts 59
8.3 Manipulation d'un flux de données RS232 - exemple de script : Manipulation de caractères.pts....60
8.4 Communication TCP/IP - Exemples de projets PingPong_TCP_Server/Client.ptp 61

9. Référence 62

9.1 Menu et barre d'outils (scripting) 63
9.2 Dialogue de la page : Modifier la séquence d'envoi 65
9.3 Dialogue de la page : Modifier la séquence de réception 66
9.4 Dialogue de gestion : Démarrer la journalisation / Créer le(s) fichier(s) de journalisation 67
9.5 Dialogue de la page : Personnaliser la sortie HTML 68
9.6 Dialogue de la séquence : Rechercher une séquence 69
9.7 Dialogue de la page : Paramètres Séquence d'envoi 70
9.8 Dialogue de gestion : Paramètres du projet - Communication 70
9.9 Dialogue de gestion des flux : Paramètres du projet - Contrôle de flux 74
9.10 Dialogue de communication : Paramètres de projet - Filtres de communication 74
9.11 Le dialogue : Options....75
9.12 de dialogue : Options des experts 76
9.13 Console de clavier 78
9.14 Spécification des sommes de contrôle....78

10. Référence (scripting) 82

10.1 Les bases de VBScript 83
10.1.1 Avis de copyright 84
10.1.2 Structures de contrôle 84
10.1.2.1 Structures de la décision 84
10.1.2.2 Structures en boucle....85
10.1.3 Variables, tableaux, constantes et types de données....86
10.1.4 Opérateur 88
10.1.5 Fonctions date/heure....89
10.1.6 Autres 91
10.2 Commandes de script Docklight - L'objet DL 93

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

10.2.1 Méthodes 94
10.2.1.1 Ajouter un commentaire 94
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 Quitter 99
10.2.1.8 ResetCompteur de réception 99
10.2.1.9 Séquence d'émission 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 Méthodes (avancés) 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 (l'environnement). 114
10.2.2.7 GetHandshakeSignals 116
10.2.2.8 GetKeyState 117
10.2.2.9 GetReceiveComments 118
10.2.2.10 Champ de saisie2 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 Fichier de téléchargement 133
10.2.3 Caractéristiques 134
10.2.3.1 NoOfSendSequences 134
10.2.3.2 NoOfReceiveSequences 135
10.3 Procédures d'événements OnSend / OnReceive 135
10.3.1 Sub DL_OnSend() - Manipulation des données de la séquence d'envoi 136
10.3.2 Sub DL_OnReceive() - Évaluation des données de séquence reçues 138
10.3.3 OnSend / OnReceive - Déroulement temporel et flux de programmes 143
10.4 Objets FileInput / FileOutput pour la lecture et l'écriture de fichiers 145
10.4.1 FileInput - Lecture de fichiers 145
10.4.2 FileOutput - Écrire des fichiers 147
10.4.3 Fichiers d'entrée multiples / Fichiers de sortie multiples 148
10.5 Canaux latéraux - Utilisation de plusieurs connexions de données 148

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

10.5.1 OpenSideChannel / CloseSideChannel - Gestion de plusieurs canaux 148
10.5.2 DirectSend 150
10.6 Objet de débogage / débogage de script 151

10.7 directive #include 153
10.8 Syntaxe de la ligne de commande 153
10.9 Dialogue de la page : Personnaliser / Éditeur externe 154

11. Soutien 157

11.1 Assistance et dépannage sur le web 158
11.2 Assistance par e-mail.....158

12. Annexe 159

12.1 Tables de caractères ASCII 160
12.2 Raccourcis clavier 162
12.3 Connecteur RS232 / Pinout.....164
12.4 Câble RS232 standard 166
12.5 Docklight câble de surveillance RS232 SUB D9.....169
12.6 Robinet Docklight.....170
12.7 Robinet Docklight Pro / Robinet 485.....171

13. Glossaire / Termes utilisés 173

13.1 Action 174
13.2 Pause 174
13.3 Caractère.....174
13.4 CRC 174
13.5 DCE 175
13.6 DTE.....175
13.7 Contrôle du débit.....175
13.8 HID.....175
13.9 LIN.....176
13.10 Modbus.....176
13.11 Bus multipoint (MDB).....176
13.12 Tube nommé 176
13.13 Séquence de réception 176
13.14 RS232.....177
13.15 RS422.....177
13.16 RS485.....177
13.17 Séquence d'émission 178
13.18 Séquence.....178

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

13.19 Indice de séquence 178
13.20 Serveur pour périphériques série 179
13.21 Instantané 179

13.22 TCP 179
13.23 Déclencheur 179
13.24 UART 179
13.25 UDP 179
13.26 Modem zéro virtuel 180
13.27 Wildcard 180

Index 0

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

Auteurr vraiment

1 Droit d'auteur

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

Tous les droits sont réservés. Aucune partie de cet ouvrage ne peut être reproduite, sous quelque forme ou par quelque moyen que ce soit, sans l'autorisation écrite de l'éditeur.
- reproduites graphiquement, électroniquement ou mécaniquement, y compris par photocopie, enregistrement, enregistrement sur bande magnétique ou système de stockage et de récupération d'informations.

Marque de fabrique

Les produits auxquels il est fait référence dans ce document peuvent être soit des marques et/ou des marques déposées de leurs propriétaires respectifs. L'éditeur et l'auteur ne revendiquent pas ces marques.

Microsoft et *Windows* sont soit des marques déposées, soit des marques de Microsoft Corporation aux États-Unis et/ou dans d'autres pays.

Clause de non-responsabilité

Bien que toutes les précautions aient été prises lors de la préparation de ce document, l'éditeur et l'auteur n'assument aucune responsabilité pour les erreurs ou omissions ou pour les dommages résultant de l'utilisation des informations contenues dans ce document ou de l'utilisation des programmes et du code source qui pourraient être joints à ce document. En aucun cas, l'éditeur et l'auteur ne peuvent être tenus responsables d'un manque à gagner ou de tout autre préjudice commercial causé ou prétendument causé par ce document, directement ou indirectement.

Contact

Prise en charge du courrier électronique :
support@docklight.de www.docklight.de

Flachmann et Heggelbacher GmbH & Co. KG Arc de
l'église de la forêt 27
D-82061 Neuried
Allemagne
www.fuh-edv.de

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

Introduction

2 Introduction

2.1 Docklight - Aperçu

Docklight est un outil de test, d'analyse et de simulation pour les protocoles de communication série (RS232, RS485/422 et autres). Il vous permet de surveiller la communication entre deux appareils série ou de tester la communication série d'un seul appareil. Docklight est facile à utiliser et fonctionne sur presque tous les PC standard équipés de *Windows 11*, *Windows 10, Windows 8* ou *Windows 7*.

Les principales fonctions de Docklight sont les suivantes

Simulation de protocoles série - Docklight peut envoyer des séquences définies par l'utilisateur et réagir aux séquences entrantes, en fonction du protocole utilisé. Cela permet de simuler le comportement d'un dispositif de communication série, ce qui est particulièrement utile pour créer des conditions de test difficilement reproductibles avec le dispositif original (par exemple, des conditions problématiques).

Consignation des données RS232 - Toutes les données de communication série peuvent être consignées dans deux formats de fichier différents. Utilisez le format texte pur pour une journalisation et un stockage rapides de grandes quantités de données. Un format de fichier HTML avec du texte stylé vous permet de distinguer facilement les données entrantes des données sortantes ou des informations supplémentaires. Docklight peut également consigner tout flux de données binaires, y compris les octets ASCII 0 <NUL> et autres caractères de contrôle. **Détection de séquences de données spécifiques** - Dans de nombreux cas de test, vous devez rechercher une séquence spécifique dans les données RS232 qui indique un état problématique.

Docklight gère pour vous une liste de telles séquences de données et peut, après avoir détecté une séquence, effectuer des actions définies par l'utilisateur, par exemple prendre un instantané de toutes les données de communication avant et après la réception du message d'erreur.

Réponses aux données entrantes - Docklight vous permet de définir des réponses personnalisées aux différentes séquences de communication reçues. Vous pouvez ainsi créer un simulateur simple pour votre périphérique série en quelques minutes. Il peut également vous aider à trouver une erreur spécifique en envoyant une commande de diagnostic après la réception du message d'erreur.

Docklight fonctionne avec les interfaces COM fournies par votre système d'exploitation. Physiquement, ces interfaces sont souvent des interfaces RS232 SUB D9. Cependant, il est également possible d'utiliser Docklight pour d'autres normes de communication telles que RS485 et RS422, qui ont une structure électrique différente de RS232, mais qui suivent le mécanisme de communication RS232.

Docklight a également été testé avec succès avec de nombreux convertisseurs USB-série courants, des ports série Bluetooth, des récepteurs GPS, des null modems virtuels, Arduino, MicroPython/pyboard ou d'autres cartes embarquées/UART qui ajoutent un port COM dans Windows.

Pour les applications de surveillance RS232 full duplex, nous recommandons notre accessoire USB Docklight Tap ou notre câble de surveillance Docklight.

Dans ce manuel, seules les connexions série RS232 sont abordées, car elles constituent la base des autres connexions série mentionnées ci-dessus.

ASTUCE : pour commencer, consultez les exemples de projets Docklight, qui démontrent quelques-unes des fonctions de base de Docklight.

2.2 Docklight Scripting - Aperçu

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

Docklight Scripting est une version étendue de Docklight RS232 Terminal / RS232 Monitor. Elle offre un langage de script facile à utiliser et un éditeur intégré pour créer et exécuter des tâches de test automatisées. Un script Docklight vous permet d'effectuer toutes les opérations de base de Docklight

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

opérations (envoi de séquences de données prédéfinies, détection de certaines séquences dans le flux de données entrant, ...) et les intègre dans votre propre code de test.

Docklight Scripting est compatible avec le réseau. Au lieu d'un port COM série, Docklight Scripting peut établir des connexions TCP (mode client TCP), accepter une connexion TCP sur un port local (mode serveur TCP) ou agir comme un pair UDP. Il prend également en charge les connexions HID USB et les canaux nommés.

Docklight Scripting vous offre à la fois flexibilité et simplicité. En quelques minutes, vous pouvez développer et créer vos propres outils de test automatisés :

des ordres de contrôle programmés (par exemple, envoi d'une commande de diagnostic toutes les 5 minutes et signalement d'une erreur si la réponse de l'appareil n'est pas OK)
des cycles de test répétés (par exemple, un test d'endurance pour un système de contrôle des mouvements) /système d'entraînement) **scripts automatiques de configuration des appareils** (par exemple, réinitialisation des paramètres d'usine d'un appareil RS232 avant sa livraison) **outils d'analyse des erreurs pour les tâches de maintenance et d'entretien** (par exemple, exécution d'une série de commandes de diagnostic et exécution d'une analyse automatique des erreurs)
Testeur de protocole avec calcul automatique de la somme de contrôle (par ex. CRC - Cyclic Redundancy Codes)
Scripts de démarrage Docklight (par exemple, lancement automatique d'une tâche de journalisation du port COM au démarrage du PC)

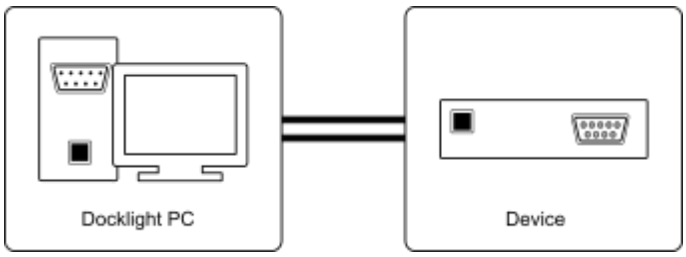
Docklight Scripting utilise le moteur VBScript, qui vous permet d'écrire vos tests dans un langage de script simple et connu. Les fonctions et caractéristiques de base de Docklight sont rendues disponibles par un petit ensemble pratique de commandes de script Docklight.

ASTUCE : pour commencer, consultez le script de test de modem Docklight, qui démontre l'utilisation des commandes de script Docklight pour un test de modem automatisé. Pour une démonstration simple des capacités TCP/IP, voir l'exemple TCP- Client/Serveur l .

2.3 Applications typiques

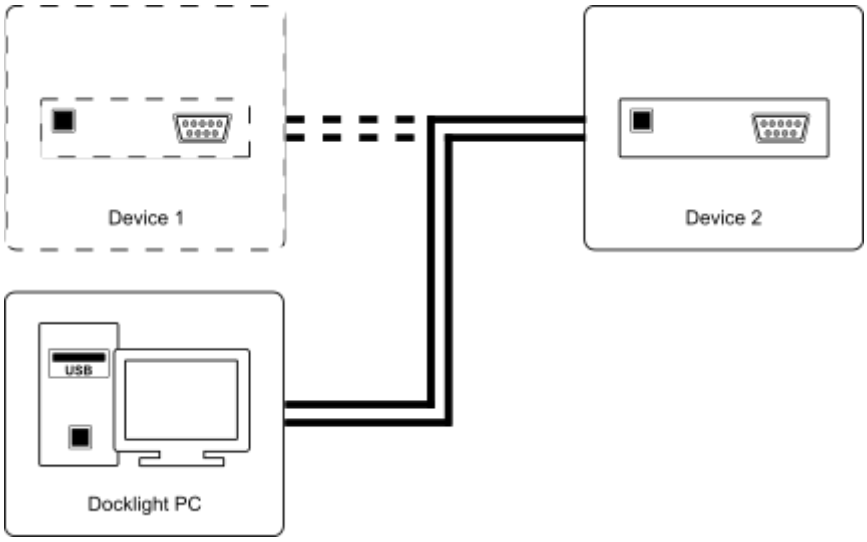
Docklight est l'outil idéal pour soutenir votre processus de développement et de test des appareils de communication série. Docklight peut être utilisé pour

Testez la fonctionnalité ou l'implémentation du protocole d'un appareil série. Vous pouvez définir des séquences de contrôle reconnues par votre périphérique, les envoyer, consigner et analyser les réponses et tester la réaction du périphérique.

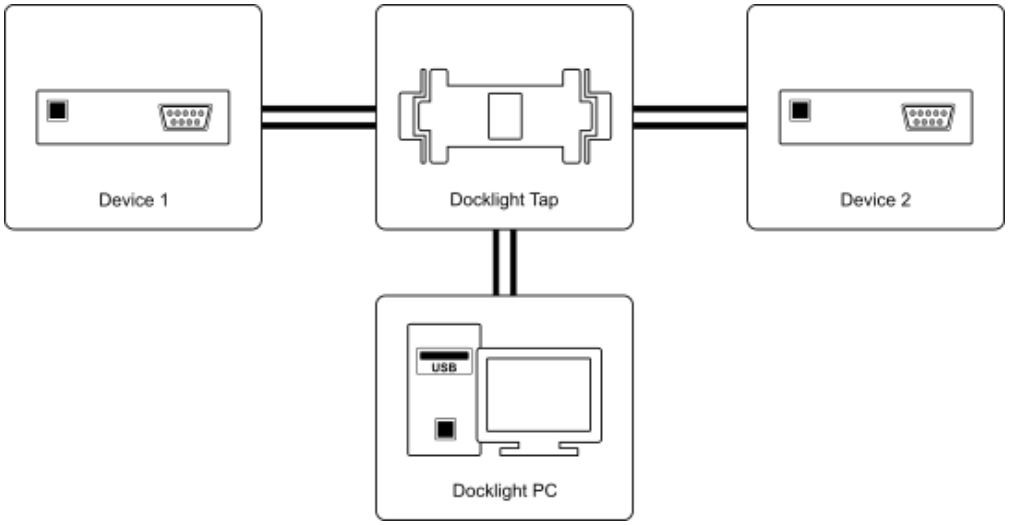


Simuler un périphérique série.
Bien que rare, la possibilité d'une erreur matérielle doit être prise en compte dans la plupart des systèmes. Imaginez que vous ayez un appareil qui envoie un message d'erreur en cas d'erreur matérielle. Un deuxième appareil doit recevoir ce message d'erreur et effectuer une réaction quelconque. Avec Docklight, vous pouvez facilement simuler le message d'erreur à envoyer et tester la réaction du deuxième appareil.

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



Surveiller la communication entre deux appareils
Branchez Docklight sur la liaison de communication entre deux appareils série. Surveillez et enregistrez la communication série dans les deux sens. Détectez les séquences de communication erronées ou les conditions d'erreur spécifiques au sein de la communication surveillée. Créez un instantané de la communication lorsqu'une telle condition d'erreur se produit.



2.4 Configuration requise

Le système d'exploitation
Windows 11, Windows 10, Windows 10 x64, Windows 8, Windows 8 x64, Windows 7, Windows 7 x64.

Exigences supplémentaires
Pour les tests ou les simulations RS232 : Au moins un port COM disponible. Deux ports COM pour surveiller la communication entre deux appareils série.
Pour la surveillance à faible latence avec Docklight Tap, Docklight Tap Pro ou Docklight Tap 485 : un port USB.
- Pour les applications Docklight Scripting TCP ou UDP : Réseau avec adressage IPv4.

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

Des câbles ou des pilotes logiciels supplémentaires peuvent être nécessaires pour connecter les appareils à tester. Voir les sections sur Docklight Tap, Docklight Monitoring Cable RS232 SUB D9, Standard RS232 Cable et Virtual Nullmodem Driver.

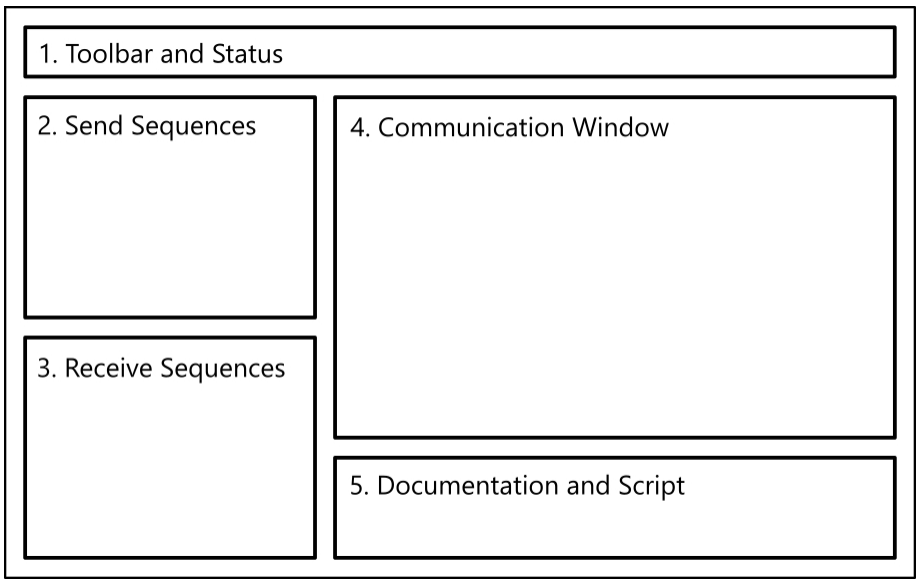
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Interface utilisateur

3 Interface utilisateur

3.1 Fenêtre principale (Scripting)

La fenêtre principale de Docklight Scripting est divisée en cinq parties :



1. Barre d'outils et état
Vous pouvez sélectionner toutes les fonctions importantes du Docklight à partir de la barre d'outils. La barre d'état en dessous affiche des informations supplémentaires sur l'état de la communication et les paramètres actuels.

2. Envoyer des séquences
Définissez, éditez et gérez ici vos séquences d'envoi. Utilisez le symbole de la flèche ou la barre d'espace pour envoyer la séquence sélectionnée.
Double-cliquez sur le champ vide à la fin d'une liste pour créer une nouvelle séquence. Un menu contextuel (bouton droit de la souris) permet de couper, de copier ou d'insérer dans le presse-papiers des séquences d'envoi entières. Voir Édition et gestion de séquences et Dialogue : Editer une séquence d'envoi pour plus d'informations.

3. Séquences reçues
Définissez, modifiez et gérez ici vos séquences de réception. Double-cliquez sur le champ vide à la fin d'une liste pour créer une nouvelle séquence. La liste des séquences de réception supporte les mêmes opérations de réorganisation et de presse-papier que la liste des séquences d'émission. Vous pouvez également copier une séquence d'envoi dans le presse-papiers et la coller dans la liste des séquences de réception.
Voir Édition et gestion des séquences et dialogue : Modifier la séquence de réception pour plus d'informations.

Vous pouvez réorganiser les listes de séquences par glisser-déposer : Autorisez d'abord le réarrangement de la liste en cliquant sur le petit symbole de cadenas dans le coin supérieur gauche. Si n'est pas verrouillé, la liste peut être modifiée en faisant glisser une séquence vers une nouvelle position tout en maintenant le bouton gauche de la souris enfoncé.

En cliquant sur le repère **|**<, vous pouvez réduire la zone des séquences d'envoi/réception.

4. Fenêtre Communication
Affiche la communication sortante et entrante de la liaison de données série.
Différents types de données de communication sont disponibles

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

ASCII / HEX / Décimal / binaire	Les options d'affichage sont disponibles, y compris Affichage, horodatage et surlignage (voir options). Lorsque la communication sérielle est arrêtée, toutes les données de la fenêtre de communication peuvent être copiées dans le presse-papier ou imprimées. Vous pouvez également rechercher des séquences spécifiques en utilisant la fonction de recherche

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

fonction de séquence. Pour plus d'informations, voir Comment les données sérielles sont-elles traitées et affichées ?

5.
5a. Documentation des projets et des processus
Saisissez des commentaires supplémentaires sur votre projet ou sur une séquence d'envoi / de réception spécifique. Docklight affiche la documentation spécifique à la séquence lorsque vous sélectionnez une séquence d'émission ou de réception dans la liste (2. et 3.). Docklight passe à la documentation principale du projet lorsque vous cliquez sur la ligne inférieure vide de la liste des séquences ou lorsque vous cliquez dans la fenêtre de communication (4.).

Pour éviter toute modification accidentelle, la zone de documentation est verrouillée par défaut et vous devez activer la modification en cliquant sur le petit symbole de cadenas au-dessus de la zone. Lorsque est déverrouillé, vous pouvez librement modifier/copier/coller/supprimer son contenu.

5b. Écriture
Modifiez ici votre code de script Docklight. Grâce à un menu contextuel (bouton droit de la souris), vous pouvez couper, copier, coller, supprimer ou rechercher/remplacer du code. Pour les fonctions d'édition avancées, il existe un support pour les éditeurs externes. Pour plus d'informations sur la création de scripts Docklight, consultez le guide Docklight Scripting Reference.

En cliquant sur le **v** à droite, vous pouvez réduire la zone Document ntation/Script.

3.2 Presse-papiers - Couper, copier et coller

Docklight prend en charge les opérations couper/copier/coller. Les opérations de presse-papier sont disponibles

- dans la fenêtre principale - Séquences d'envoi
- Fenêtre principale - Réception de
- séquences Fenêtre principale -
- Communication Fenêtre principale -
- Documentation
- Fenêtre principale - Éditeur de script (Docklight
- Scripting uniquement) Dialogue : Modifier la
- séquence d'envoi

- Dialogue de la séquence : Modifier la
- Rechercher une séquence
- séquence de réception Dialogue :
- Paramètres Séquence d'envoi
- Domaine Documentation
- Console de clavier

Vous pouvez couper une séquence de données série de la fenêtre de communication et créer une nouvelle séquence d'envoi ou de réception en la collant dans la liste correspondante. Ou bien vous pouvez éditer une séquence d'émission, copier une partie de cette séquence dans le presse-papier et créer une nouvelle séquence de réception à partir de celle-ci en la collant dans la fenêtre "Séquence de réception".

ASTUCE : Utilisez le menu contextuel du **bouton droit de la souris** pour les opérations couper/copier/coller ou la touche de raccourci clavier correspondante.

3.3 Domaine de la documentation

Docklight propose des zones de documentation dans la partie inférieure droite de la fenêtre principale et dans les boîtes de dialogue Editer la séquence d'envoi ou Editer la séquence de réception.

Vous pouvez utiliser ces zones pour noter des instructions supplémentaires

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

concernant votre application Docklight. Par exemple, comment utiliser les séquences d'émission/réception et les paramètres de séquence, ou des indications sur des appareils de test supplémentaires, etc.

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

Le contenu de la documentation est enregistré et chargé en même temps que tous les autres paramètres du projet Docklight (voir Enregistrement et chargement des données, scripts et options du projet).

ASTUCE : les zones de documentation sont de simples zones de texte sans menus de mise en forme ni outils. Pour une documentation formatée avec des images et des tableaux, vous pouvez préparer votre documentation dans *Windows WordPad* ou *Microsoft Word* et la copier-coller dans la zone de documentation Docklight.

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

Caractéristiques et fonctions

4 Caractéristiques et fonctions

4.1 Comment les données sérielles sont traitées et affichées

Docklight traite toutes les données sérielles d'une manière orientée sur 8 bits. Chaque séquence de données série se compose d'un ou de plusieurs caractères de 8 bits.
Avec Docklight, vous pouvez

afficher les données sérielles soit au format ASCII, HEX, décimal ou binaire

copier les données sérielles dans le presse-papiers et les coller dans un fichier texte standard ou un fichier formaté
document *Microsoft®* Word, ou créer une séquence d'envoi/réception en utilisant les données.

- Impression des données sérielles, des commentaires des utilisateurs et d'autres informations

La fenêtre de communication de Docklight affiche la communication en cours via le(s) port(s) série sélectionné(s). Docklight distingue deux canaux de communication (canal 1 et canal 2), qui représentent les données entrantes et sortantes en mode émission/réception ou les deux canaux de communication à surveiller en mode surveillance. Les données du canal 1 et du canal 2 sont affichées dans des couleurs ou des polices différentes. Les données de communication peuvent être imprimées ou enregistrées dans un fichier journal au format texte ou HTML.

Outre les données série, Docklight insère également des horodatages dans l'affichage de la communication. Par défaut, un horodateur est inséré chaque fois que le sens du flux de données change entre le canal 1 et le canal 2 ou avant qu'une nouvelle séquence d'envoi ne soit transmise. Plusieurs options sont disponibles pour insérer des horodatages supplémentaires. Ceci est particulièrement utile lors de la surveillance d'une ligne semi-duplex avec un seul canal de communication. Voir Options --> Date/ Horodatage

Docklight est capable de traiter des flux de données sérielles contenant n'importe quel code ASCII de 0 à 255 décimales. Comme il existe des caractères de contrôle non imprimables (code ASCII < 32) et différents codages pour le code ASCII > 127, tous ces caractères ne peuvent pas être affichés dans la fenêtre de texte ASCII. Néanmoins, tous les caractères sont correctement traités par Docklight et peuvent être affichés au format HEX, décimal ou binaire. Docklight traite les données sérielles de la même manière sur toutes les versions linguistiques du système d'exploitation Windows, même si l'affichage ASCII peut être différent. Pour les caractères de contrôle (code ASCII
< 32), une option d'affichage supplémentaire est disponible pour afficher leur équivalent textuel dans la fenêtre de communication. Voir la boîte de dialogue Options et l'annexe, Tables de caractères ASCII.

Docklight vous permet de supprimer toutes les données sérielles originales lorsque vous effectuez un test dans lequel vous n'avez pas besoin de voir les données réelles, mais seulement les évaluations supplémentaires générées à l'aide de séquences de réception. Voir à ce sujet les paramètres de projet pour les filtres de communication.

4.2 Édition et gestion de séquences

Un projet Docklight se compose principalement de séquences définies par l'utilisateur. Il peut s'agir soit de séquences d'émission envoyées par Docklight lui-même, soit de séquences de réception utilisées pour détecter un message spécifique dans les données sérielles entrantes.

Les séquences sont définies à l'aide de la boîte de dialogue Editer la séquence d'envoi ou Editer la séquence de réception. Cette boîte de dialogue s'ouvre
1. en **choisissant** l'option **Modifier** dans le menu contextuel du **bouton droit de la souris**.
2. en double-cliquant sur une séquence existante ou en appuyant sur
Ctrl + E lorsque la liste est sélectionnée "Envoyer la séquence" ou "Séquence

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

". 3. lors de la création d'une nouvelle séquence, en double-cliquant sur le champ vide à la fin d'une liste (ou en appuyant sur **Ctrl + E**).
4. lorsqu'une nouvelle séquence est insérée dans la liste des séquences.

Docklight prend en charge l'utilisation de jokers (par exemple le joker " ?" comme caractère de remplacement pour n'importe quel caractère) dans les séquences de réception et d'émission. Voir les sections

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

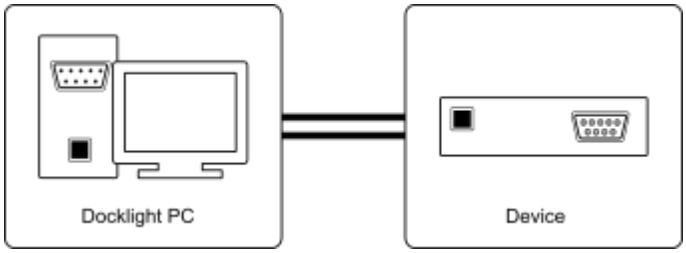
Envoi de commandes avec des paramètres et vérification des séquences avec des caractères aléatoires pour les détails et les exemples.

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

Travailler avec Docklight

5 Travailler avec Docklight

5.1 Tester un périphérique série ou une implémentation de protocole



Conditions préalables

Vous avez besoin de la spécification du protocole à tester, par exemple sous forme écrite. Le périphérique sériel à tester doit être connecté à l'un des ports COM du PC. Voir la section Câble RS232 standard pour les détails sur la connexion de deux appareils série.

- Le dispositif sériel doit être prêt à fonctionner.

Réalisation du test

A) Créer un nouveau projet

Créez un nouveau projet Docklight en sélectionnant le menu **Fichier > Nouveau projet**.

B) Régler les options de communication

1. Choisissez le menu **Outils > Paramètres du projet...**
2. Sélectionnez le mode de communication **Envoyer/Recevoir**
3. Sous **Envoyer/Recevoir sur le canal de communication**, définissez le port COM auquel votre appareil série est connecté.
4. **Réglez** le débit en bauds et tous les autres **paramètres de port COM** nécessaires.
5. Confirmez les paramètres et fermez la boîte de dialogue en cliquant sur le bouton **OK**. cliquer.

C) Définir les séquences d'émission à utiliser

Vous allez probablement tester votre périphérique série en envoyant certaines séquences selon le protocole utilisé par le périphérique et en observant la réaction du périphérique. Suivez les étapes ci-dessous pour créer votre liste de séquences :

1. Double-cliquez sur la dernière ligne du tableau Séquences d'envoi. La boîte de dialogue Modifier la séquence d'envoi s'affiche (voir aussi Modifier et gérer les séquences).
2. Saisissez un **nom** pour la séquence. Le nom de la séquence doit être unique pour chaque séquence d'envoi définie.
3. Saisissez vous-même la **séquence**. Vous pouvez saisir la séquence soit au format ASCII, soit au format HEX, soit au format décimal, soit au format binaire. Les boutons d'option du **mode d'édition** vous permettent de passer à tout moment d'un format à l'autre.
4. Après avoir cliqué sur le bouton **OK**, la nouvelle séquence est ajoutée à la liste des séquences d'envoi.

Répétez les étapes 1 à 4 pour définir les autres séquences d'émission dont vous avez besoin pour réaliser votre test.

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

D) Définir les séquences de réception utilisées
Si vous souhaitez que Docklight réagisse à la réception de certaines séquences, vous devez définir une liste de séquences de réception.

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

1. Double-cliquez sur la dernière ligne du tableau Séquences de réception. La boîte de dialogue Modifier la séquence de réception s'affiche. La boîte de dialogue se compose de trois parties : Champ de nom, Champ de séquence et Champ d'action.
2. Modifiez les champs **Nom** et **Séquence**.
3. Indiquez une action à effectuer après la réception de la séquence par Docklight. Quatre types d'actions sont possibles :
   Réponse - Après réception de la séquence, envoyez l'une des séquences d'envoi.
   Commentaire - Après réception de la séquence, insérez un commentaire personnalisé dans la fenêtre de communication (et le fichier journal, le cas échéant).
   Trigger - Il s'agit d'une fonction avancée qui, en capturant une séquence...
   Stop - Après la réception de la séquence, Docklight met fin à la communication.
4. Cliquez sur le bouton **OK** pour ajouter la nouvelle séquence à la liste.

Répétez les étapes 1 à 4 pour définir les autres séquences de réception dont vous avez besoin pour réaliser votre test.

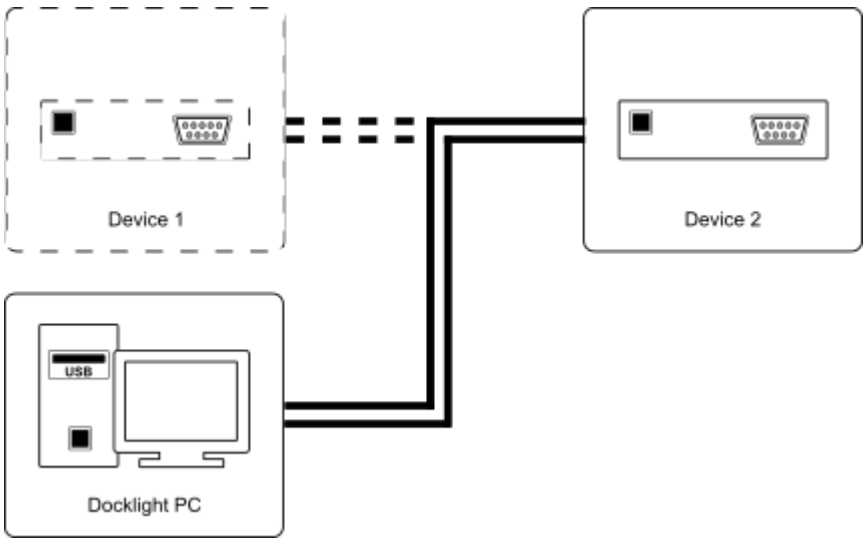
E) Enregistrer le projet
Avant d'effectuer le test proprement dit, il est recommandé de sauvegarder les paramètres de communication et les séquences définis. Cela se fait via le menu **Fichier > Enregistrer le projet**.

F) Réalisation du test
Démarrez Docklight en choisissant **Exécuter > Démarrer Communication.**

Docklight ouvre une connexion série selon les paramètres indiqués. Il affiche ensuite toutes les communications entrantes et sortantes dans la fenêtre de communication.
Utilisez le bouton **Envoyer** pour envoyer l'une des séquences définies au périphérique série. L'affichage de toutes les transmissions de données à l'écran vous permet de vérifier le comportement du périphérique. Toutes les informations du journal peuvent être consignées dans un fichier texte pour une analyse ultérieure. Voir la section Consigner et analyser un test.

ASTUCE : Grâce à la zone de documentation, vous pouvez facilement prendre des notes supplémentaires ou copier et coller des parties du protocole de communication pour les documenter davantage.

5.2 Simuler un appareil série



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

Conditions préalables

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

Vous avez besoin de la spécification du comportement du périphérique série que vous souhaitez simuler, z. Par exemple, quel type d'informations est renvoyé après la réception d'une certaine commande.
Un deuxième appareil est connecté à un port PC-COM qui est relié à votre simulateur communique.

Ce deuxième appareil et son comportement constituent le véritable objet d'intérêt. Un exemple pourrait être un appareil qui vérifie régulièrement l'état d'une ASI (alimentation sans interruption) via un protocole de communication série. Vous pourriez utiliser Docklight pour simuler le comportement de base de l'UPS et certains cas de problèmes d'UPS. Cela est très utile lorsque vous testez l'autre appareil, car il peut être assez difficile de reproduire une condition d'alarme (comme une mauvaise batterie) sur l'ASI réelle.

REMARQUE : le deuxième appareil peut également être une deuxième application logicielle. Il est possible de faire fonctionner à la fois Docklight et l'application logicielle sur le même PC. Il suffit d'utiliser un port COM différent pour chacune des deux applications et de relier les deux ports COM avec un câble null-modem RS232. Vous pouvez également utiliser un null-modem virtuel à cette fin.

Réalisation du test

A) Créer un nouveau projet
Créez un nouveau projet Docklight en sélectionnant le menu **Fichier > Nouveau projet**.

B) Régler les options de communication

1. Choisissez le menu **Outils > Paramètres du projet...**
2. Sélectionnez le mode de communication **Envoyer/Recevoir**
3. Sous **Envoyer/Recevoir sur le canal de communication**, définissez le port COM auquel votre appareil série est connecté.
4. **Réglez** le débit en bauds et tous les autres **paramètres de port COM** nécessaires.
5. Confirmez les paramètres et fermez la boîte de dialogue en cliquant sur le bouton **OK**. cliquer.

C) Définition des séquences d'émission utilisées
Définissez toutes les réactions de votre simulateur. Pensez aux réactions lorsque l'appareil simulé se trouve dans un état normal et aux réactions en cas de problème. Dans l'exemple de l'onduleur mentionné ci-dessus, une panne de batterie serait un tel cas de problème, difficile à reproduire avec l'appareil original. Pour tester comment d'autres appareils réagissent à une panne de batterie, définissez la séquence de réponse correspondante que votre UPS enverrait dans ce cas.

REMARQUE : Voir Tester un périphérique série... pour savoir comment définir des séquences d'envoi.

D) Définir les séquences de réception utilisées
Dans la plupart des cas, votre appareil simulé n'enverra pas de données non sollicitées, mais sera interrogé par l'autre appareil. L'autre appareil utilisera une série de séquences de commandes prédéfinies pour demander différents types d'informations. Définissez ici les séquences de commandes qui doivent être interprétées par votre simulateur.

Pour chaque séquence de commandes définie, indiquez **Réponse** comme action. Choisissez l'une des séquences définies sous C). Si vous souhaitez utiliser deux ou plusieurs séquences de réponse alternatives, créez plusieurs copies de la même séquence de réception et donnez-leur un nom différent (par exemple, "status cmd - answer ok", "status cmd - answer battery failure", "status cmd - Réponse panne de secteur") et attribuez différentes séquences d'envoi comme action.

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

Dans l'exemple, vous auriez trois éléments dans la liste Séquences de réception qui réagiraient à la même commande par trois réponses différentes. Pendant le test, vous pouvez décider quelle réponse doit être envoyée en activant ou en désactivant les éléments de la liste dans la colonne **Actif**.

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

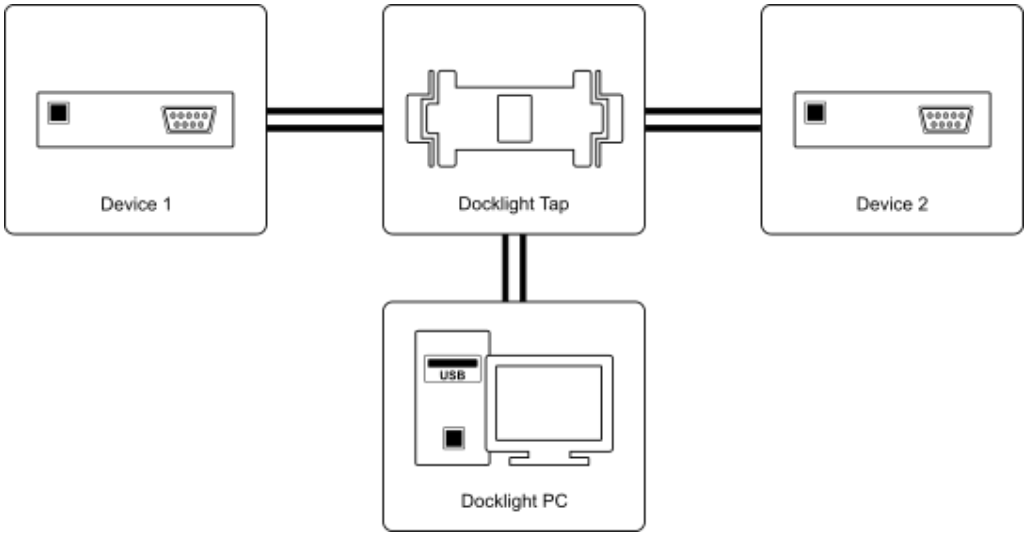
E) Enregistrer le projet
Avant d'effectuer le test proprement dit, il est recommandé de sauvegarder les paramètres de communication et les séquences définis. Cela se fait via le menu **Fichier > Enregistrer le projet**.

F) Réalisation du test
Démarrez Docklight en choisissant **Exécuter > Démarrer Communication.**

Le Docklight répond maintenant à toutes les commandes reçues du périphérique série connecté. Le témoin de transmission de données à l'écran vous permet de surveiller le flux de communication. Toutes les informations du journal peuvent être enregistrées dans un fichier texte pour une analyse plus approfondie. Voir la section Consigner et analyser un test.

ASTUCE : dans la zone de documentation, vous pouvez facilement ajouter des notes supplémentaires ou copier et coller des parties du protocole de communication pour une documentation plus approfondie.

5.3 Surveillance de la communication sérielle entre deux appareils



Conditions préalables

Un câble de surveillance Docklight, Docklight Tap ou Docklight Tap Pro/485 est nécessaire pour écouter les signaux d'émission RS232 des deux appareils sériels et les injecter dans Docklight sans perturber la communication entre les appareils. Pour installer un câble de surveillance Docklight, deux ports COM doivent être disponibles sur votre PC pour la surveillance. Chaque port reçoit les données de l'un des appareils sériels à surveiller.

- L'appareil 1 et l'appareil 2 doivent être prêts à fonctionner.

Réalisation du test

A) Créer un nouveau projet
Créez un nouveau projet Docklight en sélectionnant le menu **Fichier > Nouveau projet**.

B) Régler les options de communication

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

1. Choisissez le menu **Outils > Paramètres du projet...**

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

2. Sélectionner le mode de communication **Surveillance**

Alternative I - Utilisation du câble de surveillance du Docklight :

3. Sous **Receive Channel 1, réglez** le port COM sur lequel le signal de surveillance de l'appareil série 1 est reçu. Sous **Receive Channel 2**, réglez le port COM pour le deuxième appareil.

   REMARQUE : en mode monitoring Docklight, toutes les données reçues d'un port COM sont retransmises sur le canal d'émission du port COM opposé ("Data Forwarding"). Cela n'a pas d'effet sur les configurations de câbles de surveillance Docklight, car le signal TX n'est pas connecté. Elle peut toutefois être utile pour des applications spéciales où vous devez faire passer le trafic de données série par Docklight via un câblage RS232 standard. Si vous souhaitez un comportement de surveillance purement passif, où aucune donnée TX n'apparaît, vous pouvez désactiver le "transfert de données" via le menu **Outils >** Options expert....

Alternative II - Utilisation de Docklight Tap

Ouvrez pour le**canal de réception 1**, ouvrez la liste déroulante, faites défiler vers le bas jusqu'à la section **-- USB Taps --** et sélectionnez le premier port Tap,
z. PAR EXEMPLE, **TAP0**. Pour le **canal de réception 2**, le deuxième port de prise (par ex. **TAP1**) est automatiquement sélectionné.

Alternative III - Utilisation du Docklight Tap Pro / Docklight Tap 485 :

3. Pour le **canal de réception 1,** ouvrez la liste déroulante, faites défiler vers le bas jusqu'à la section **-- USB-Taps --** et sélectionnez le premier port VTP-Tap, par exemple **VTP0**. Pour le **canal de réception 2**, le deuxième port VTP-Tap (par ex. **VTP1**) est automatiquement sélectionné.

4. Réglez le débit en bauds et tous les autres paramètres de communication pour le protocole utilisé.

   REMARQUE : assurez-vous que le port série de votre PC fonctionne correctement avec le débit en bauds et les paramètres de communication utilisés par le périphérique 1 et le périphérique 2. Si les appareils 1 et 2 utilisent un protocole de transmission de données à grande vitesse, les ports série du PC et le logiciel Docklight lui-même peuvent être trop lents pour recevoir correctement toutes les données.

5. Confirmez les paramètres et fermez la boîte de dialogue en cliquant sur le bouton **OK**. cliquer.

C) Définir les séquences de réception utilisées
Définissez des séquences de réception qui doivent être marquées dans le protocole de contrôle ou qui déclenchent une action au sein de Docklight. Docklight vérifie les séquences de réception sur les deux canaux de surveillance, c'est-à-dire qu'il importe peu que les séquences proviennent du périphérique série 1 ou du périphérique série 2.

REMARQUE : Étant donné qu'un câble de surveillance spécial est utilisé pour ce test, toute la communication entre le périphérique série 1 et le périphérique série 2 n'est pas altérée et aucun retard supplémentaire n'est causé par Docklight lui-même. Ceci est particulièrement important lorsque Docklight est utilisé pour détecter des problèmes de temps. Cela signifie également qu'il n'y a aucune possibilité d'influencer la communication sérielle entre les deux appareils. Tant que le mode de communication **Surveillance** est sélectionné, il n'est pas possible d'utiliser des séquences d'envoi.

D) Enregistrer le projet

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

Avant d'effectuer le test proprement dit, il est recommandé d'examiner les paramètres définis.

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

enregistrer les paramètres de communication et les séquences. Cela se fait via le menu **Fichier > Enregistrer le projet**.

E) Réalisation du test

Démarrez Docklight en choisissant **Exécuter > Démarrer Communication**, puis activez les périphériques série 1 et 2 et effectuez un test.
Docklight affiche toute la communication entre le périphérique série 1 et le périphérique série 2. Docklight utilise différentes couleurs et polices de caractères

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

les polices de caractères pour faciliter la distinction entre les données transmises par l'appareil 1 et l'appareil 2. Les couleurs et les polices peuvent être sélectionnées dans l'onglet Affichage de la boîte de dialogue **Outils > Options...** peuvent être sélectionnés.

ASTUCE : La fonction Snapshot vous permet de localiser une séquence rare ou un état d'erreur dans un protocole de communication avec une grande quantité de données.

CONSEIL : Les sections Comment augmenter la vitesse de traitement... et Comment obtenir la meilleure précision de timing vous expliquent comment adapter Docklight aux applications qui traitent de grandes quantités de données ou qui exigent une plus grande précision de timing.

5.4 Capture d'une séquence spécifique et création d'un instantané de la communication

Lors de la surveillance de la communication série entre deux appareils, il peut arriver que vous souhaitiez tester une erreur rare, en ne vous intéressant qu'à la communication série avant et après cet événement. Vous pourriez rechercher cette situation en enregistrant le test et en recherchant la séquence d'erreur caractéristique dans les fichiers journaux. Cela pourrait signifier que vous devez enregistrer et analyser plusieurs Mo de données, alors que vous ne recherchez en fait que quelques octets, si tant est qu'ils apparaissent. Comme alternative, vous pouvez utiliser la fonction d'instantané, comme décrit ci-dessous.

Conditions préalables

Docklight est prêt à effectuer un test tel que décrit dans les cas d'utilisation précédents, par exemple la surveillance de la communication sérielle entre deux appareils.

Prendre un instantané

A) Définir un déclencheur pour l'instantané

1. Définissez la séquence qui se produit dans votre situation d'erreur comme séquence de réception.
2. Activez l'onglet "**Déclencheur"** dans la section "Action" de la boîte de dialogue "Recevoir une séquence" : L'option "Déclencheur" doit être activée s'il s'agit de la séquence que vous souhaitez tracer.

REMARQUE : N'oubliez pas de désactiver l'option de déclenchement pour toutes les autres séquences de réception qui doivent être ignorées dans votre test, afin qu'elles ne déclenchent pas le snapshot.

B) Créer un instantané

Cliquez sur le bouton **Snapshot dans** la barre d'outils. Docklight démarre la communication, mais n'affiche rien dans la fenêtre de communication. Lorsque la séquence de déclenchement est détectée, Docklight affiche les données de communication avant et après l'événement déclencheur. D'autres données sont traitées jusqu'à ce que la séquence de déclenchement se trouve à peu près au milieu de la fenêtre de communication. Docklight arrête alors la communication et positionne le curseur sur la séquence de déclenchement.

5.5 Consigner et analyser un test

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

Conditions préalables

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

Docklight est prêt à effectuer un test tel que décrit dans les cas d'utilisation précédents, par exemple le test d'un périphérique série ou d'une implémentation de protocole.

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

Enregistrement du test

Cliquez sur le bouton **Démarrer l'enregistrement** dans la barre d'outils principale. Une boîte de dialogue s'ouvre pour sélectionner les paramètres du fichier journal.

Pour chaque représentation (ASCII, HEX, ...), il est possible de créer un fichier de protocole séparé. Choisissez au moins une représentation. Les fichiers journaux ont une extension de fichier ".txt", ".htm" ou ".rtf", selon le format de fichier que vous avez choisi. Docklight ajoute également le type de présentation au nom du fichier afin de distinguer les différents fichiers journaux. Par exemple, si l'utilisateur indique "Test1" comme nom de fichier, le fichier texte ASCII sera appelé "Test1_asc.txt", tandis qu'un fichier journal RTF-HEX sera appelé "Test1_hex.rtf".

Confirmez vos paramètres pour le fichier journal et lancez la journalisation en cliquant sur le bouton **OK**.

Pour arrêter la journalisation et fermer le(s) fichier(s) de journalisation, cliquez sur le bouton **Stop Logging** dans la barre d'outils principale. Tant que le(s) fichier(s) de journalisation n'a/n'ont pas été fermé(s), il n'est pas possible de voir tout son/leur contenu.

ASTUCE : Si vous avez des exigences supplémentaires pour votre format de fichier journal, par exemple le lancement d'un nouveau fichier toutes les heures, vous pouvez utiliser un script Docklight et la méthode StartLogging à cet effet. Voir aussi l'exemple de script **LogFileNamesTimestamp.zip** (dossier **Extra\NLogFileNamesTimestamp** dans votre répertoire Script Samples).

5.6 Vérification des séquences contenant des caractères aléatoires (réception de séquences Wildcards)

De nombreux appareils sériels prennent en charge une série de commandes pour la transmission de données de mesure et d'autres informations associées. Dans les protocoles courants basés sur le texte, la réponse du périphérique série se compose d'une partie fixe (par exemple, "température=") et d'une partie variable représentant la valeur réelle (par exemple, "65F"). Pour reconnaître correctement toutes ces réponses dans le flux de données série, il est possible de définir des séquences de réception avec des caractères génériques.

Supposons par exemple la situation suivante : Un appareil série mesure la température et envoie périodiquement la valeur mesurée actuelle. Docklight affiche la sortie suivante : `07/30/2012 10:20:08.022 [RX] - temperature=82F<CR>`

```
30/07/2012 10:22:10.558 [RX] - température=85F<CR>
30/07/2012 10:24:12.087 [RX] - température=93F<CR>
30/07/2012 10:26:14.891 [RX] - température=102F<CR>
```

...

Définir une séquence de réception pour chaque valeur de température possible ne serait pas une option pratique. Au lieu de cela, vous définiriez une séquence de réception avec des caractères génériques. Par exemple
t | e | m | p | e | r | a | t | u | r | e | = | ? | # | # | F | r ("r"
est le caractère de retour chariot final <CR>)

Cette ReceiveSequence serait déclenchée pour chacune des chaînes de température énumérées ci-dessus. Elle autorise une valeur de 1 à 3 chiffres pour la

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

température (c'est-à-dire de "0" à "999"). L'exemple suivant décrit étape par étape comment définir la séquence ci-dessus. Voir également les remarques supplémentaires à la fin de ce

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

pour des informations supplémentaires sur les espaces réservés '#'.

REMARQUE : Voir Calcul et validation des sommes de contrôle sur la réception et la validation des données de somme de contrôle, par exemple les CRC. Il n'est pas nécessaire d'utiliser des caractères génériques pour les zones de somme de contrôle. Utilisez plutôt quelques valeurs de caractères standard, par exemple "00 00" en représentation HEX.

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

Conditions préalables

Docklight est prêt à effectuer un test tel que décrit dans les cas d'utilisation précédents, par exemple le test d'un périphérique série ou d'une implémentation de protocole.

- L'appareil sériel (dans notre exemple, l'appareil de mesure de la température) est en service.

Utilisation de séquences de réception avec des caractères génériques

A) Préparation du projet
Créez un nouveau projet Docklight et configurez tous les paramètres de communication.

B) Définir les séquences de réception utilisées

1. Créez une nouvelle séquence de réception. Saisissez un **nom** pour la séquence.
2. **Saisissez** la partie fixe de votre réponse attendue dans la section "**Séquence"**. Pour notre exemple, vous entreriez la séquence suivante en mode ASCII :
   t | e | m | p | e | r | a | t | u | r | e | =
3. Ouvrez le menu contextuel/populaire avec le **bouton droit de la souris** et sélectionnez **Caractère de remplacement ' ? (correspond à un caractère)** pour insérer le premier espace réservé à la position du curseur. Ajoutez deux espaces réservés '#' via le menu local Espace réservé **'#' (correspond à zéro ou à un caractère)**. La séquence se présente maintenant comme suit :
   t | e | m | p | e | r | a | t | u | r | e | = | ? | # | #
4. Saisissez l'extrémité fixe de notre chaîne de température, c'est-à-dire la lettre "F" et la terminaison Le caractère <CR>. Vous pouvez utiliser le raccourci standard **Ctrl+Enter** pour entrer le <CR> / code ASCII 13. La séquence est maintenant : t | e | m | p | e | r | a | t | u | r | e | = | ? | # | F | r
5. Indiquez une **action** à effectuer après la saisie d'une valeur de mesure de la température.
6. Cliquez sur **OK** pour ajouter la nouvelle séquence à la liste des séquences de réception.

REMARQUE : Pour distinguer les caractères de remplacement '?' et '#' des points d'interrogation réguliers ou des caractères numériques (code décimal 63 / 35), les caractères de remplacement sont représentés dans l'éditeur de séquence avec une couleur de fond différente.

C) Réalisation du test
Démarrez Docklight en choisissant **Exécuter > Démarrer la communication**.

Docklight détecte alors chaque valeur de température et exécute l'action indiquée.

REMARQUE : La procédure événementielle DL_OnReceive() permet de poursuivre l'évaluation et le traitement des données de mesure effectivement reçues.

Remarques supplémentaires sur les caractères génériques '#'.

1. Les caractères génériques '#' à la fin d'une Receive Sequence n'ont aucun effet. La Receive Sequence "HelloWorld###" se comporte comme une Receive Sequence "HelloWorld".

2. Une "correspondance au sein d'une correspondance" n'est pas renvoyée : Si une séquence de réception "Hello#######World" est définie et que les données entrantes sont "Hello1Hello2World", la séquence de réception reconnue est "Hello1Hello2World", et non "Hello2World".

Macros pour la réception de commentaires sur les séquences

Les macro-mots clés peuvent être utilisés dans la zone de texte Edit Receive Sequence **> 3 - Action > Comment pour créer** des textes de commentaires

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

Docklight avec des

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

données, par exemple les données effectivement reçues.

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

Macro	Est remplacé par
%_S	signal de cloche. Créez un "bip", en fonction de votre *Windows* système solide.
%_L	Retour à la ligne
%_T	Horodatage des données reçues
%_C	Docklight N° de canal / direction des données (1 ou 2) pour les données reçues
%_X	Le nom du canal ou l'alias du canal correspondant à la direction des données %_C. Par exemple "RX", "TX" ou "COM5".
%_I	Index de la liste des séquences de réception, voir le dialogue : Modifier la séquence de réception
%_N	Nom de la séquence de réception
%_A	Les données réelles qui ont déclenché cette séquence de réception. Utiliser la représentation ASCII
%_H	La même chose que %_A, mais en représentation HEX
%_D	Comme %_A, mais en représentation décimale
%_B	Comme %_A, mais en représentation binaire
%_A(1,4)	Syntaxe avancée : N'insérer que les 4 premiers caractères de cette séquence de réception (en commençant par le caractère n° 1, longueur de la séquence = 4).
%_H(3,-1)	Syntaxe avancée : Insérez tout à partir du troisième caractère jusqu'à la fin de la séquence (longueur = -1). Utilisez la représentation HEX.

exemple :
Pour une séquence de réception telle que décrite ci-dessus (t | e | m | p | e | r | a | t | u | r | e | = | ? | # | # | F | r_), vous pourriez définir le texte de commentaire suivant :

Nouvelle température = %_L %_A(13, -3) °F

La sortie de Docklight pourrait alors ressembler à ceci :

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


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

5.7 Enregistrement et chargement des données de votre projet, des scripts et des options

Vous pouvez définir le comportement de Docklight Scripting à l'aide de trois types de données de configuration utilisateur :

- Pdonnées du projet
- Scripts
- Options du programme

Enregistrement et chargement des données du projet

Les données du projet comprennent

- Envoi de séquences
- Réception de séquences
- Paramètres supplémentaires du projet : Mode de communication, ports COM utilisés, paramètres des ports COM (vitesse de transmission, parité, ...)
- Contenu de la documentation

Le projet est enregistré avec le menu **Fichier > Enregistrer le projet** ou **Fichier > Enregistrer le projet sous...** dans un fichier de projet Docklight (fichier .ptp).

Il est généralement recommandé de sauvegarder votre projet avant de lancer un

test. REMARQUE : Lorsque vous enregistrez votre projet,seules les séquences, les

réglages et les
Documentation section Données. Si vous souhaitez enregistrer un protocole de communication pendant un test voir la section Enregistrement et analyse d'un test.

Le chargement d'un projet se fait via le menu **Fichier > Ouvrir un projet....**

Enregistrement et chargement de scripts

Le code des scripts Docklight pour les tâches de test automatisées est enregistré dans un fichier séparé (fichier .pts). Utilisez le menu **Scripting > Enregistrer le script** ou **Scripting > Enregistrer le script sous...**

Enregistrement et chargement des options du programme

Les options de Docklight (formatage du texte, comportement des caractères de contrôle, etc.) peuvent être modifiées à l'aide de la boîte de dialogue Options de Docklight (menu **Outils > Options...**).

ASTUCE : lorsque vous exécutez votre script, vous voudrez peut-être utiliser un ensemble d'options spécifique pour vous assurer que Docklight génère la communication et la sortie du journal dans un format bien défini. Utilisez les méthodes SaveProgramOptions et LoadProgramOptions pour créer un fichier d'options et charger les options au démarrage de votre script.

Utiliser les paires de projets et de scripts : _auto.pts

Dans la plupart des applications de script Docklight, un script Docklight (fichier .pts) nécessite un projet associé (fichier .ptp). Vous pouvez utiliser le schéma de nommage

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

suivant

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

pour permettre le chargement et le lancement automatiques du script :

monprojet**.ptp**
monprojet_auto**.pts**

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

Si les deux fichiers se trouvent dans le même dossier, Docklight Scripting effectue les opérations supplémentaires suivantes :

Lorsque **myproject.ptp** est ouvert (soit par un double-clic dans l'Explorateur Windows, soit via le menu **Fichier > Ouvrir un projet...**), Docklight Scripting ouvre également **myproject_auto.pts** au passage, s'il n'est pas déjà ouvert.
Lorsque **myproject_auto.pts** est ouvert, Docklight Scripting ouvre également **myproject.ptp** à côté, si elle n'est pas déjà ouverte.

Lorsque **Démarrer la communication** est exécuté, le port de communication est ouvert et le script démarre.

REMARQUE : si **myproject.ptp** est ouvert dans Docklight (sans script), un avertissement apparaît pour indiquer qu'il semble s'agir d'un projet avec support de script et que son utilisation dans Docklight (sans script) est limitée.

REMARQUE : Les méthodes de script OpenProject et StartCommunication sont protégées par le **_auto.**pts comportement n'est pas affecté.

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

Travailler avec Docklight (niveau avancé)

6 Travailler avec Docklight (niveau avancé)

6.1 Envoi d'instructions avec des paramètres (séquences d'envoi - jokers)

Lorsque vous testez un périphérique série, il est fort probable que le périphérique prenne en charge une série de commandes contenant un paramètre.

Exemple : un appareil photo numérique prend en charge une commande de réglage du temps d'exposition. Pour régler le temps d'exposition à 25 millisecondes, vous devez envoyer la séquence suivante :
e | x | p | 0 | 2 | 5 | r ("r" est un caractère de retour chariot <CR> final)

Pour éviter de définir une nouvelle séquence d'envoi pour chaque enregistrement, vous pouvez utiliser à la place une séquence d'envoi avec des caractères génériques :
e | x | p | | ? | ? | ? | r

L'exemple suivant, étape par étape, décrit comment définir une commande de temps d'exposition avec un paramètre et utiliser une valeur d'exposition différente à chaque fois que la séquence est envoyée.

Conditions préalables

Docklight est prêt à effectuer un test tel que décrit dans la section Tester un périphérique série ou une implémentation de protocole.

Exécution du test via des commandes avec des paramètres

A) Préparation du projet
Créez un nouveau projet Docklight et configurez tous les paramètres de communication.

B) Définir les commandes utilisées

1. Créez une nouvelle séquence d'envoi. Saisissez un **nom** pour la séquence.
2. **Saisissez** la partie fixe de votre commande dans la section "**Séquence"**. Pour notre exemple, vous entreriez la séquence suivante en mode ASCII : e | x | p | |
3. Ouvrez maintenant le menu contextuel avec **le bouton droit de la souris** et sélectionnez **Espace réservé ' ? (correspond à un caractère) F7** pour insérer un espace réservé à la position du curseur. Dans notre exemple, nous devrions répéter cette opération jusqu'à ce qu'il y ait trois espaces réservés '?' pour notre temps d'exposition à trois chiffres. La séquence se présente alors comme suit :
   e | x | p | | ? | ? | ?
4. Ajoutez maintenant le caractère final <CR> en utilisant le raccourci standard pour les caractères de contrôle **Ctrl+Enter.** L'exemple de séquence est maintenant
   e | x | p | | ? | ? | ? | r
5. Cliquez sur **OK** pour ajouter la nouvelle séquence à la liste des séquences d'envoi.

Répétez les étapes 1 à 5 pour définir d'autres commandes nécessaires à la réalisation de votre test.

REMARQUE : Pour distinguer un caractère de remplacement '?' d'un caractère ASCII en forme de point d'interrogation (code décimal 63), le caractère de remplacement est représenté dans l'éditeur de séquence par une couleur de fond différente.

C) Envoi d'une commande au périphérique série

1. Utilisez le bouton ----> **Envoyer** pour envoyer le message sériel.

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

Fig. 35 Communication : port et d'envoyer une commande au périphérique série.
envoyer.

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

2. La communication s'arrête et la boîte de dialogue "Paramètres de la séquence d'envoi" s'affiche, dans laquelle vous pouvez saisir la valeur du paramètre. Dans notre exemple, il s'agit d'un temps d'exposition, par exemple "025".
3. Confirmez en appuyant sur la **touche "Entrée"**. La séquence est maintenant envoyée au périphérique série.

Il est possible de définir des commandes avec plusieurs paramètres en utilisant plusieurs plages de caractères génériques au sein d'une séquence. La boîte de dialogue Envoyer les paramètres de la séquence apparaît alors plusieurs fois avant qu'une séquence ne soit envoyée.

REMARQUE : Si vous utilisez le **joker '?'**, vous devez indiquer exactement un caractère pour chaque '?' lorsque vous envoyez la séquence. Pour les paramètres de longueur variable, utilisez **Wildcard**
'#' (correspond à zéro ou à un caractère) F8.

REMARQUE : Vous ne pouvez pas utiliser une séquence d'envoi avec des caractères génériques comme réponse automatique à une séquence de réception (voir Action).

REMARQUE : Si votre séquence d'envoi nécessite une somme de contrôle, vous pouvez la définir comme décrit dans Calcul et validation des sommes de contrôle. La somme de contrôle est calculée après que la zone des caractères de remplacement/paramètres a été remplie avec les données réelles, puis les données de séquence résultantes en sont transmises à la file d'attente d'envoi.

6.2 Augmentation de la vitesse de traitement et prévention des messages "Input Buffer Overflow".

Lorsqu'il surveille la communication sérielle entre deux appareils, Docklight ne peut pas contrôler la quantité de données entrantes. Étant donné que Docklight applique une série de règles de formatage et de conversion aux données sérielles, seul un nombre limité d'octets par seconde peut être traité. Il existe de nombreux facteurs qui déterminent la vitesse de traitement, tels que les périphériques PC et COM utilisés, les paramètres d'affichage et les actions de séquence de réception définies. Il n'est donc pas possible de donner un débit de données typique.

La tâche qui prend le plus de temps à Docklight est le formatage des couleurs et des polices, qui est appliqué par défaut (voir Options d'affichage de Docklight). Si Docklight ne peut pas suivre le formatage des données entrantes, il passe automatiquement en mode texte clair, plus simple.

Si cela n'est toujours pas assez rapide pour traiter les données entrantes, Docklight ajoute le message suivant dans la sortie de la fenêtre de communication et dans les fichiers journaux.

```
DOCKLIGHT signale : Dépassement de la mémoire tampon d'entrée sur COM1
```

ASTUCE : Cherchez ce message avec la fonction **Séquence dans la fenêtre de communication...** (Ctrl + F).

Si vous observez le comportement décrit ci-dessus, Docklight vous offre plusieurs possibilités d'augmenter le débit de données.

1. Simplifier la sortie d'écran :
   - Désactivez tous les modes d'affichage inutiles dans la boîte de dialogue **Options...**
   - Utilisez dès le départ le mode Texte clair (voir le comportement de commutation automatique décrit ci-dessus).
   - Si vous utilisez le mode ASCII, désactivez l'option Description des caractères de contrôle

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

. Sæ l e s données de communication dans un fichier texte pur, au lieu de d'utiliser la ou les fenêtres de communication
- Utilisez le format de fichier journal "texte brut".
- Créez uniquement un fichier de protocole pour la représentation (ASCII / HEX / décimale / binaire) dont vous avez réellement besoin.

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

- Désactiver les fenêtres de communication pendant l'enregistrement en utilisant l'option High Speed Logging

3. Utilisez le filtre de communication de la boîte de dialogue **ramètres du projet...** et désactivez les données série originales pour l'une ou les deux directions de communication. Ceci est particulièrement utile lorsque vous savez exactement ce que vous recherchez et que vous pouvez définir une ou plusieurs séquences de réception pour ces données. Ces séquences de réception peuvent émettre un commentaire à chaque fois que la séquence apparaît dans le flux de données, de sorte que vous savez toujours ce qui s'est passé, même si les données sérielles d'origine ne sont pas affichées par Docklight.

6.3 Comment obtenir la meilleure précision de timing

De nombreux programmes de surveillance RS232 - dont Docklight - ne peuvent saisir les données sérielles que de manière limitée dans le temps. Cela peut avoir pour conséquence que les données de deux directions de communication différentes soient affichées dans un ordre temporel incorrect ou que plusieurs télégrammes d'une direction de communication apparaissent comme un seul paquet de données.

Ce comportement n'est pas dû à une mauvaise programmation, mais est plutôt caractéristique d'un système PC/Windows et des différentes couches matérielles et logicielles impliquées.
Les retards non spécifiés et les imprécisions temporelles peuvent être dus à :

- Le chipset du périphérique COM, par exemple le tampon de données FIFO interne (premier entré, premier sorti). La transmission par bus USB (pour les convertisseurs USB-série).
- Le pilote de périphérique série pour Windows.
- La planification des tâches/processus dans un système d'exploitation multitâche comme Windows.
- La précision du fournisseur de la date et de l'heure.

Docklight dispose d'une horloge et d'une date très précises avec une granularité de l'ordre de la milliseconde, mais doit néanmoins accepter les contraintes de l'environnement matériel et logiciel environnant.

Vous pouvez faire ce qui suit pour minimiser les retards et les imprécisions supplémentaires et atteindre une précision typique de 5 millisecondes ou mieux pour le chronométrage :

1. Procurez-vous notre Docklight Tap pour les temps de latence USB les plus bas. Ou utilisez les ports RS232 intégrés s'ils sont encore disponibles sur votre PC.

2. Sélectionnez sous **Outils > Options expert...** le mode de processus Externe / Haute priorité. Dialogue.

3. Lorsque vous surveillez de grandes quantités de données, suivez les recommandations de la section précédente Comment augmenter la vitesse de traitement... afin d'éviter que le tampon d'entrée ne déborde et que l'ordinateur ne réagisse plus en raison de la charge élevée de l'unité centrale.

REMARQUE : Les **options expertes** recommandées ci-dessus... modifient l'équilibre général du système et doivent être utilisées avec précaution. Les meilleurs résultats ne peuvent être obtenus que si Docklight est **exécuté en tant qu'administrateur**. Veuillez vous assurer que vous avez bien compris les remarques et les avertissements figurant dans la documentation.

4. Comme alternative aux 1-3 ci-dessus, utilisez nos accessoires Docklight Tap Pro ou Docklight Tap 485, qui utilisent leur propre fournisseur de temps embarqué et éliminent complètement les imprécisions basées sur le PC.

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

6.4 Système Validation des sommes de contrôle

De nombreux protocoles de communication contiennent des champs de somme de contrôle supplémentaires afin de garantir l'intégrité des données et de détecter les erreurs de transmission. Un algorithme courant est le CRC (Cyclic

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

Redundancy Code), qui est utilisé dans différentes variantes pour différents protocoles. L'exemple suivant décrit étape par étape comment configurer le calcul à la volée des sommes de contrôle pour une séquence d'émission et comment activer la validation automatique d'une plage de sommes de contrôle dans une séquence de réception.

ASTUCE : Vous trouverez un exemple d'adressage d'un appareil Modbus esclave dans le tutoriel Modbus RTU avec somme de contrôle CRC.

ASTUCE : Voir aussi la méthode DL.CalcChecksum pour calculer les sommes de contrôle en utilisant du code de script comme alternative.

Conditions préalables

Vous connaissez la spécification des sommes de contrôle pour les messages du protocole : Quelle partie des données de la séquence est protégée par une somme de contrôle ?
Où se trouve la somme de contrôle ? (Normalement à la fin de la séquence.) Quel algorithme de somme de contrôle doit être utilisé ? (Très probablement l'un des types CRC standard ou une simple somme MOD256).

Utilisation de séquences d'envoi avec calcul automatique de la somme de contrôle

A) Définition d'une séquence d'envoi contenant une somme de contrôle

1. Créez une nouvelle séquence d'envoi. Saisissez un **nom** pour la séquence.

2. **Saisissez** la partie "séquence" de votre message dans la section "**Séquence"**.
   Ici, nous utilisons par exemple un message HEX très simple comme séquence :
   01 | 02 | 03 | 04 | **? ?**

   Utilisez le menu contextuel via le **bouton droit de la souris** ou **F7** pour créer le caractère de

   remplacement **? ?** REMARQUE : Pour plus d'informations sur les caractères de remplacement et les

   paramètres, voir le
   Section Paramètres de la séquence d'envoi.

3. Ajoutezmaintenant à une autre valeur 00 comme caractère de remplacement pour la somme de contrôle. 01 | 02 | 03 | 04 | ? ? | **00**

   REMARQUE : Dans une séquence d'envoi, vous pouvez utiliser n'importe quel code de caractère de 00 à FF comme caractère de remplacement aux endroits où la somme de contrôle calculée doit être insérée ultérieurement. Cela diffère du fonctionnement dans une séquence de réception, où vous pouvez utiliser ? ? Caractère de remplacement. Voir l'exemple suivant pour une séquence de réception.

4. Allez dans l'onglet **Paramètres supplémentaires | Somme de contrôle** et définissez la somme de contrôle. Ici, par exemple, nous avons choisi **MOD256** dans la liste déroulante.

   REMARQUE : Le champ de texte pour la somme de contrôle autorise les commentaires. Tout ce qui suit le signe # n'est qu'un commentaire. Vous pouvez ajouter vos propres commentaires pour décrire ce dont il s'agit avec cette somme de contrôle.

5. Cliquez sur **OK** pour ajouter la nouvelle séquence à la liste des séquences d'envoi.

B) Réalisation du test

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

6. utilisez le bouton **Envoyer** ----> pour envoyer l'une des commandes prédéfinies. Saisissez une valeur de paramètre, par exemple 05.

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

FortgBeevsocr dhierDiattteen gnesee ndet, Docklight calcule le MOD256-actuel.
somme de contrôle. Le résultat se trouve à la position de somme de contrôle indiquée. Pour MOD256, il s'agit par défaut de la dernière position de caractères, ce qui signifie que le caractère de remplacement 00 est écrasé par le résultat de la somme de contrôle.

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

Si nous utilisons 05 comme paramètre lors de l'envoi de la séquence, les données envoyées par Docklight se présentent comme suit :

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

Le caractère de remplacement a été remplacé par la somme des octets de message :
1 + 2 + 3 + 4 + 5 = 15 ou Hex 0F.

Utilisation de séquences de réception avec vérification automatique de la somme de contrôle

A) Définition d'une séquence de réception avec somme de contrôle

1. Créez une nouvelle séquence de réception. Saisissez un **nom** pour la séquence.
2. Saisissez les données de la séquence, y compris une plage de caractères de remplacement pour un octet de données utiles aléatoire et un caractère de remplacement pour la somme de contrôle. Nous utilisons le même télégramme que dans l'exemple de séquence d'envoi ci-dessus : 01 | 02 | 03 | 04 | ? | **00**.
3. Allez sur l'onglet **Action | Commentaire** et saisissez le texte suivant : **Somme de contrôle correcte**
4. Allez dans l'onglet **Somme de contrôle** et sélectionnez **MOD256** dans la liste déroulante de gauche.
5. **Conservez** l'option **Détecter la somme de contrôle OK**. Elle signifie que la séquence de réception ne sera déclenchée que si l'octet de somme de contrôle MOD256 est correct dans les données reçues.
5. Cliquez sur **OK** pour confirmer les modifications.

B) Réalisation du test

6. Démarrez la communication et envoyez quelques télégrammes de données à votre application Docklight / port COM.

La sortie dans la fenêtre de communication pourrait ressembler à ceci :

```
15.02.2016 17:43:28.072 [RX] - 01 02 03 04 05 0F Correct Somme de contrôle
15.02.2016 17:43:31.870 [RX] - 01 02 03 04 0F 19 Correct Somme de contrôle
15.02.2016 17:43:35.833 [RX] - 01 02 03 04 10 1A Correct Somme de contrôle
```

REMARQUE : Dans cet exemple, nous avons montré comment définir une séquence de réception qui ne se déclenche que pour les télégrammes de données dont la somme de contrôle est correcte. Il est également possible de faire l'inverse : détecter une erreur de somme de contrôle.
Pour ce faire, passez à l'onglet **Somme de contrôle** et modifiez l'option **Reconnaître la somme de contrôle OK** en **somme de contrôle Faux**.

6.5 Contrôle et surveillance des signaux handshake RS232

Les paramètres du projet Docklight pour la prise en charge du contrôle des flux offrent

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

un mode manuel qui vous permet de définir ou de réinitialiser manuellement les signaux RTS et DTR en cliquant sur l'indicateur correspondant. La section suivante décrit comment utiliser le **signe de fonction ' ! (F11**

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

touche) pour modifier temporairement les signaux RTS et DTR au sein d'une séquence d'émission ou pour détecter les changements pour les lignes CTS, DSR, DCD ou RI à l'aide d'une séquence de réception.

Conditions préalables

Docklight est prêt à exécuter un test tel que décrit dans Tester un périphérique série ou une implémentation de protocole.
L'assistance au contrôle de flux est réglée sur "Manuel" dans les paramètres du projet.
Le projet Docklight contient déjà une ou plusieurs séquences d'émission, mais il y a une exigence supplémentaire pour l'alternance des signaux RTS/DTR lors de l'émission.

Mise en œuvre des changements de signaux RTS/DTR

Pour notre exemple, nous supposons que nous utilisons un convertisseur RS485 qui nécessite le contrôle de l'émetteur-récepteur RS485, mais qui utilise le signal DTR au lieu du RTS pour commuter entre l'"envoi" et la "réception". Nous supposons également qu'il existe déjà une séquence d'envoi "test" qui se présente comme suit en mode ASCII :
T | e | s | t

A) Modifier la séquence d'envoi existante

1. Ouvrez la boîte de dialogue Modifier la séquence d'envoi.
2. **Passez** le **mode d'édition** en mode **décimal**. Notre exemple de "test" se présente comme suit en mode décimal :
   084 | 101 | 115 | 116
3. Insérez un caractère de fonction RTS/DTR au début : Appuyez sur **F11** ou ouvrez le menu contextuel avec le **bouton droit de la souris** et sélectionnez **Caractère de fonction ' ! (signaux RTS et DTR)**. La séquence d'exemple est maintenant la suivante :
   ! | 084 | 101 | 115 | 116
4. Ajoutez le nouvel état RTS/DTR comme valeur de paramètre décimale (voir ci-dessous). Dans notre exemple, le signal DTR doit être défini sur High. Nous choisissons "002" comme valeur de paramètre, de sorte que la séquence soit maintenant
   ! | 002 | 084 | 101 | 115 | 116
5. Insérez un caractère de fonction RTS/DTR à la fin de la séquence et utilisez "000" comme valeur de paramètre pour mettre le signal DTR à zéro. Les données de la séquence sont maintenant les suivantes :
   ! | 002 | 084 | 101 | 115 | 116 | ! | 000
6. Cliquez sur **OK** pour confirmer les modifications.

REMARQUE : Pour distinguer un ' ! RTS/DTR d'un point d'exclamation ASCII (code décimal 33), le point de fonction RTS/DTR est affiché dans l'éditeur de séquence avec une couleur de fond différente.

REMARQUE : le caractère qui suit un caractère de fonction RTS/DTR est utilisé pour définir les signaux RTS/DTR et n'est pas envoyé à l'appareil série (voir les valeurs des paramètres ci-dessous).

B) Envoi des données avec commande DTR supplémentaire

1. Envoyez la séquence de test en cliquant sur le bouton ----> **Envoyer**.

Docklight met alors le signal DTR à l'état haut, envoie la séquence ASCII "Test" et réinitialise ensuite DTR.

REMARQUE : les indicateurs RTS/DTR indiquent tout changement d'état RTS ou

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

DTR. Toutefois, dans l'exemple ci-dessus, DTR est défini et réinitialisé très rapidement, de sorte que

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

ne donnera probablement pas de retour visuel.
Si vous voulez vraiment "voir" le comportement de DTR, essayez d'introduire un petit délai entre les caractères.

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

Signe de fonction ' ! (F11) - Réglage de RTS et DTR

Valeur du caractère (mode décimal)	RTS	DTR
000	Faible	Faible
001	Haute	Faible
002	Faible	Haute
003	Haute	Haute

Modifications temporaires de la parité / applications 9 bits

Certains protocoles et applications nécessitent un 9e bit de données, par exemple pour l'adressage des appareils sur un bus. La seule façon d'adresser de tels appareils avec un UART standard de 8 bits de données maximum est d'utiliser des réglages sériels contenant un bit de parité et de modifier temporairement ce bit de parité dans une séquence d'envoi. Le caractère de fonction '!' supporte à cet effet des valeurs de paramètres supplémentaires :

Valeur du caractère (mode décimal)	Parité
016	Pas de parité
032	Parité impaire
048	Parité droite
064	Marquer le point. Mettre le bit de parité à '1' logique
080	espacement des caractères. Mettre le bit de parité à '0' logique

Les nouveaux paramètres de parité sont appliqués à partir du caractère régulier suivant, tant du côté de l'émission que de la réception. La parité revient aux paramètres de communication d'origine une fois que la séquence d'envoi est entièrement transmise.

REMARQUE : Les paramètres les plus utiles pour ce caractère de fonction sont les paramètres "Mark" et "Space", car ils vous permettent de définir le bit de parité à une valeur définie qui sert effectivement de neuvième bit de données.

REMARQUE : Il est recommandé de définir le signe d'erreur de parité sur "(ignorer)" afin de pouvoir lire les données entrantes dans les deux cas, 9.

ASTUCE : Voir aussi le projet exemple **SwitchParityDemo.ptp** (dossier **Extras\ParitySwitch_9BitProtocols** dans votre répertoire \Samples).

Signe de fonction ' ! (F11) - Détection des changements de signaux de Handshake (CTS, DSR, DCD ou RI)

Docklight Scripting reconnaît certes les modifications des signaux de poignée de main CTS, DSR, DCD ou RI, mais en fonctionnement normal, ces modifications ne sont pas visibles dans la fenêtre de communication de Docklight (un peu comme un état de rupture).

Le caractère de fonction '!' permet de rendre ces modifications visibles et/ou de définir une action après la détection de telles modifications. Le caractère de fonction '!' supporte à cet effet les valeurs de paramètres suivantes :

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

Valeur du caractère (mode décimal)	**Signal Handshake**

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

	CTS = élevé
002	DSR = élevé
004	DCD = élevé

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

008	RI (Ring Indicator) = élevé

REMARQUE : Voir aussi DL.GetHandshakeSignals() pour l'ensemble étendu d'états de signaux pris en charge dans les applications Tap Pro / Tap RS485.

Exemple de définition de séquence de réception en mode d'édition décimal :

Ordre de réception (mode décimal)	Description
! | 001	est déclenché si CTS=high, tous les autres signaux low
! | 006	est déclenché si CTS=low, DSR=high, DCD=high, RI=low
! | ? ??	est déclenché à chaque modification des lignes d'état

Pour l'exemple suivant, nous supposerons que Docklight est prêt à exécuter un test, comme décrit dans la section Tester un périphérique série ou une implémentation de protocole, et que la prise en charge du contrôle de flux est réglée sur "Manuel" dans les paramètres du projet.

A) Créer une nouvelle séquence de réception pour détecter les changements de signal de poignée de main.

1. Ouvrez la boîte de dialogue Modifier la séquence de réception.
2. **Passez** le **mode d'édition** en mode **décimal**.
3. Insérez un caractère de fonction "Signaux" au début : Appuyez sur **F11** ou ouvrez le menu contextuel avec **le bouton droit de la souris** et sélectionnez **Signe de fonction ' ! (modifications CTS/DSR/DCD/RI)**.
4. Ajoutez l'état du handshake comme valeur de paramètre décimale (voir ci-dessus). Dans notre exemple, nous voulons détecter si CTS est à l'état haut alors que tous les autres signaux sont à l'état bas. Cela signifie que nous devons entrer "001" comme valeur de paramètre, de sorte que la séquence est maintenant la suivante :
   ! | 001
5. Indiquez un **commentaire** pour cette séquence, par exemple "[CTS = high, DSR/DCD/RI = low]".
6. Cliquez sur **OK** pour confirmer le nouvel ordre.

B) Lancez le test et confirmez que Docklight détecte désormais lorsque la ligne CTS passe de basse à haute.

REMARQUE : Cet exemple ne fonctionne que si CTS est la seule ligne de poignée de main avec un niveau "élevé". Pour une approche plus flexible, vous pouvez définir le caractère après le caractère de fonction '!' comme un caractère générique et utiliser la procédure d'événement DL_OnReceive() pour évaluer l'état des lignes de poignée de main.

Caractère de fonction '^'. (F12) - comparaison bit à bit

Le caractère de fonction '^' peut être ajouté en appuyant sur **F12** dans la boîte de dialogue "Edit Receive Sequence". Après le caractère '^', deux valeurs de caractères supplémentaires indiquent quels bits doivent être vérifiés (*"mask"*) et quelles valeurs sont attendues pour ces bits ("*value"*).

Séquence de réception (mode HEX)	Description

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

^ | *masque* | *valeur*	Est une correspondance pour le prochain caractère reçu si : ((*nextCharacterReceived* XOR *value*) AND *mask*) = 0 En d'autres termes - le caractère '^' ne sélectionne que les bits marqués dans le *masque* et les compare avec les bits correspondants dans la *valeur*. Voir ci-dessous pour des exemples.

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

\c\eOcF	Est une correspondance si pour le prochain caractère ce qui suit s'applique :

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 = (n'a pas d'importance)
! | ^ | 04 | 04	Cette séquence de réception est déclenchée lorsque le nouvel état du signal Handshake est DCD = haut. Tous les autres signaux de Handshake peuvent prendre n'importe quel état. REMARQUE : cette séquence de réception est déclenchée à chaque modification d'un signal de poignée de main si DCD reste toujours à l'état haut.

ASTUCE : Cette extension est également utilisée dans l'exemple de projet Docklight Scripting Docklight_TapPro_Demo**.ptp** (voir le dossier **Extras\TapPro** dans votre répertoire). \Samples répertoire)

6.6 Création et détection de retards entre les caractères

Certaines applications, en particulier les applications de microcontrôleur ne disposant pas de leur propre tampon de données série, nécessitent un délai supplémentaire entre les différents caractères afin d'éviter les dépassements de tampon et de permettre au microcontrôleur d'exécuter d'autres codes.

Dans Docklight, vous pouvez réaliser des retards entre les caractères en insérant un ou plusieurs **caractères de fonction '&' (touche F9)** dans vos données de séquence d'envoi, suivis d'un caractère indiquant le temps de retard souhaité, de 0,01 seconde à 2,55 secondes.

Vous pouvez également utiliser le caractère de retard '&' dans une séquence de réception pour indiquer un temps de pause minimum pendant lequel aucun autre caractère ne doit être reçu. Cela est utile pour détecter les pauses dans le flux de données qui indiquent le début/la fin d'un télégramme, en particulier dans les protocoles où il n'y a pas de caractère de début ou de fin propre.

Conditions préalables

Docklight est prêt à exécuter un test tel que décrit dans Tester un périphérique série ou une implémentation de protocole.
Le projet Docklight contient déjà une ou plusieurs séquences d'émission, mais un délai supplémentaire est nécessaire à certaines positions de caractères.

Envoi de données avec des retards entre les caractères

Nous utilisons comme exemple une application de microcontrôleur qui comprend une commande "get". En mode ASCII, la séquence d'envoi serait la suivante :
g | e | t | r_ ("r" est un caractère de retour de chariot <CR> final)

Les étapes suivantes décrivent comment ajouter un délai supplémentaire de 20 millisecondes entre chaque caractère afin d'éviter les dépassements de mémoire tampon du côté du microcontrôleur.

A) Modifier la séquence d'envoi existante

1. Ouvrez la boîte de dialogue Modifier la séquence d'envoi.
2. **Passez** le **mode d'édition** en mode **décimal**. Notre exemple "get" se présente comme suit en mode décimal : 103 | 101 | 116 | 013
3. Insérez un caractère de temporisation entre le premier et le deuxième caractère : Appuyez sur **F9**, ou ouvrez avec le **bouton droit de la souris** le

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

Comme vous le voyez, sélectionnez le signe de fonction '&' (retard). Le site Kickdrive vous montre un message d'erreur. La séquence d'exemple est maintenant : 103 | & | 101 | 116 | 013

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

4. Ajoutez le temps de retard : dans cet exemple, une valeur décimale de 002 (20 millisecondes) est ajoutée après le caractère de fonction "&". La séquence est maintenant
   103 | & | 002 | 101 | 116 | 013
5. Insérez un délai entre toutes les autres positions de caractère intermédiaire : Vous pouvez copier le caractère de retard et le temps de retard avec **Ctrl+C** et les coller aux endroits souhaités avec **Ctrl+V**. Notre exemple de séquence est finalement le suivant
   103 | & | 002 | 101 | & | 002 | 116 | & | 002 | 013
   Ou retour en mode ASCII :
   g | & o|| e | & o|| t | & o|| r
6. Cliquez sur **OK** pour confirmer les modifications.

REMARQUE : Pour distinguer un caractère de retard '&' d'un caractère ASCII commercial normal (code décimal 38), le caractère de fonction de retard est affiché dans l'éditeur de séquence avec une couleur de fond différente.

REMARQUE : le caractère qui suit un caractère de la fonction de temporisation est interprété comme un temps de temporisation et n'est pas envoyé au périphérique série.

B) Envoi de la commande à l'application du microcontrôleur

1. Envoyez les séquences d'envoi modifiées en cliquant sur le bouton ----> **Envoyer**.

Docklight envoie les mêmes données qu'auparavant, mais laisse des intervalles de temps supplémentaires définis par les caractères de temporisation. L'écran de communication affiche les mêmes données de communication que sans les caractères de temporisation.

REMARQUE : la précision de Docklight pour le temps de retard est limitée, car il n'a aucun contrôle sur le tampon TX-FIFO interne de l'UART. Les temps de retard indiqués pour le caractère '&' de la fonction de retard sont des valeurs minimales. Les valeurs de retard mesurées sont beaucoup plus élevées, car Docklight attend toujours un temps minimum pour s'assurer que la mémoire tampon TX-FIFO de l'UART est vide. Le format d'affichage et les réglages de puissance supplémentaires influencent également le timing. Si vous avez des exigences plus spécifiques concernant le timing de la séquence d'envoi et que vous devez contrôler le temps d'attente de Docklight ainsi que vos paramètres UART-FIFO, veuillez contacter notre support par e-mail.

ASTUCE : Si vous avez besoin du même délai entre les différents caractères des données envoyées, consultez l'exemple de script **SendByteTiming.pts** (voir le dossier **Extras\SendByteTiming** dans votre répertoire de scripts). Ce script décompose automatiquement vos séquences d'envoi en caractères individuels et envoie les données "octet par octet" avec un délai prédéfini entre les caractères.

Détection des pauses avec une séquence de réception

Docklight propose déjà l'option d'affichage Détection des pauses... pour insérer des horodatages ou des sauts de ligne supplémentaires après les pauses de communication.

Si vous n'avez pas seulement besoin d'un formatage visuel, mais que vous souhaitez également définir des actions après une pause minimale ou simplement vous assurer que l'algorithme de reconnaissance de la séquence de réception redémarre après une pause, vous pouvez ajouter le caractère de la fonction de temporisation à votre définition de la séquence de réception.

Dans la plupart des applications, le meilleur emplacement pour le caractère de temporisation est au début de la séquence de réception, avant les caractères de

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

données de réception proprement dits. Vous pouvez également créer une séquence de réception qui ne contient qu'un caractère de retard.
/pause et ne contient pas de données sérielles à proprement parler. Cela peut être très utile

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

, cumhzreiittlitcheenBeesch restrictions àmettreenœuvre,parexemple
Réinitialiser la reconnaissance des télégrammes après une pause.
pour

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

ASTUCE : Voir le projet **LineParser.ptp / LineParser.pts** et le fichier script (dossier **Extras\LineParser** dans votre répertoire Script Samples) pour un exemple d'application.

6.7 Définition et reconnaissance d'un état de "break

Certains protocoles d'application série (par ex. LIN) utilisent l'état "break" à des fins de synchronisation. Docklight Scripting prend en charge l'envoi d'un "break" dans une séquence d'envoi et la détection d'un état "break" à l'aide d'une définition de séquence de réception. Les signaux "Break" sont insérés dans votre définition de séquence en insérant un **signe de fonction '%' (touche F10)**. Un signal "Break" du Docklight a une longueur minimale de 15 *.
<longueur de bit nominale>.

Conditions préalables

Docklight est prêt à effectuer un test tel que décrit dans la section Tester un périphérique série ou une implémentation de protocole.
Le projet Docklight contient déjà une ou plusieurs séquences d'émission, mais la signalisation ou la détection d'un état de "break" est également nécessaire.

Envoi d'un état de "break

Nous supposons qu'il existe déjà une séquence d'envoi "test" qui se présente comme suit en mode ASCII :
T | e | s | t

1. Ouvrez la boîte de dialogue Modifier la séquence d'envoi.
2. Insérez un signe de fonction "Break" au début : Appuyez sur **F10** ou ouvrez le menu contextuel avec **le bouton droit de la souris** et sélectionnez **Signe de fonction '%' (signal Break)** . L'exemple de séquence est maintenant
   % | T | e | s | t
3. Cliquez sur **OK** pour confirmer les modifications.
4. Envoyez la séquence de test en cliquant sur le bouton ----> **Envoyer**.

La ligne d'émission passe en blanc (logiquement 0) pendant au moins 15 bits, puis la séquence ASCII "Test" est transmise. Le caractère "Break" n'apparaît pas sur l'affichage de la fenêtre de communication.

Détection d'un état de "break

Les signaux "Break" reçus ne sont pas affichés dans la fenêtre de communication, car ils ne font pas partie de la séquence de données proprement dite. Il est néanmoins possible de définir une séquence de réception qui contient un signal de fonction "Break".

1. Créez une nouvelle séquence de réception. Saisissez un **nom** pour la séquence.
2. Ajout d'un **caractère de fonction '%' (signal d'interruption)** pour les données de séquence.
3. Saisissez une **action pour la** séquence de réception, par exemple le commentaire "BREAK detected".
4. Cliquez sur **OK** pour confirmer les modifications.
5. Démarrer la communication.

Docklight ajoute désormais `BREAK detected` à l'affichage de la fenêtre de communication chaque fois qu'un signal d'interruption `est détecté`.

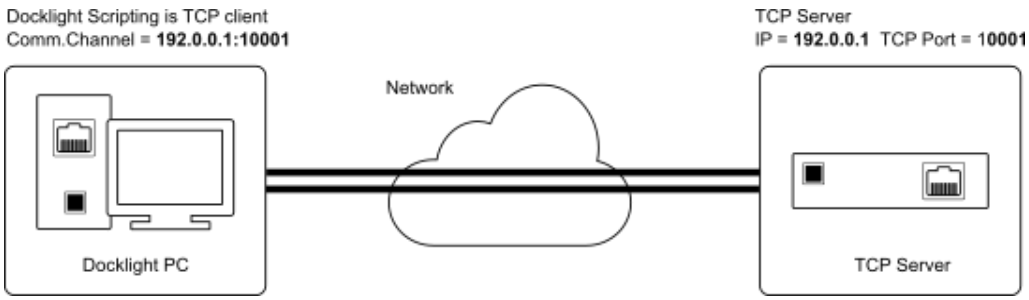
REMARQUE : Après la détection d'un signal d'interruption, un signal supplémentaire peut être émis. le caractère <NUL> (code décimal 0) apparaît dans le flux de données reçu. Ce comportement ne peut pas être contrôlé par

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

Docklight, cela dépend de la manière dont l'UART série du port COM de votre PC interprète l'état de break.

REMARQUE : si vous devez mettre en œuvre une séquence de réception qui vérifie la présence d'un signal d'interruption suivi de données supplémentaires, sachez que Docklight ne peut pas déterminer la position exacte du signal d'interruption dans le flux de données. Le signal d'interruption apparaîtra parfois plus tôt dans le flux de données, mais jamais plus tard qu'à sa position réelle. Pour définir une séquence de réception qui se déclenche de manière sûre à Break + certaines données, vous pouvez utiliser la solution de contournement suivante : Insérez quelques caractères de remplacement '#' (zéro ou un caractère) entre le caractère Break et les données supplémentaires. La séquence de réception qui en résulte pourrait ressembler à ce qui suit :
% | # | # | # | # | # | # | # | T | e | s | t

6.8 Tester un appareil serveur TCP (scripting)



Conditions préalables

L'adresse IP de l'appareil est connue et l'appareil est accessible via le réseau à partir de l'ordinateur sur lequel Docklight Scripting fonctionne - c'est-à-dire qu'un "ping" vers l'adresse IP de l'appareil fonctionne.
Vous savez quel port TCP utiliser pour vous connecter à votre appareil.
Vous connaissez la spécification du protocole pour le dispositif, par exemple Modbus TCP, et l'ensemble des commandes que le dispositif comprend.

Tester les fonctions de protocole du serveur TCP

A) Régler les options de communication

1. Choisissez le menu **Outils > Paramètres du projet...**
2. Sélectionnez le mode de communication **Envoyer/Recevoir**
3. Sous **Envoyer/Recevoir sur le canal de communication**, saisissez l'adresse IP et le numéro de port TCP pour la connexion avec l'appareil, par exemple 192.0.0.1:10001.
4. Confirmez les paramètres et fermez la boîte de dialogue en cliquant sur le bouton **OK**. cliquer.

ASTUCE : si vous voulez vous connecter à un serveur qui fonctionne sur le même ordinateur que Docklight, vous pouvez utiliser le mot-clé **LOCALHOST** au lieu de l'adresse IP réelle de votre ordinateur (par exemple, LOCALHOST:10001 pour une connexion à un serveur sur le port 10001 du même ordinateur). L'utilisation de l'adresse de bouclage 127.0.0.1 a le même effet.

B) Définition des séquences d'émission et de réception utilisées :
Définissez toutes les commandes et réponses nécessaires à votre test, comme décrit dans Tester un périphérique série ou une implémentation de protocole.

C) Réalisation du test

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

"

- Sélectionner Run > Start Communication.

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

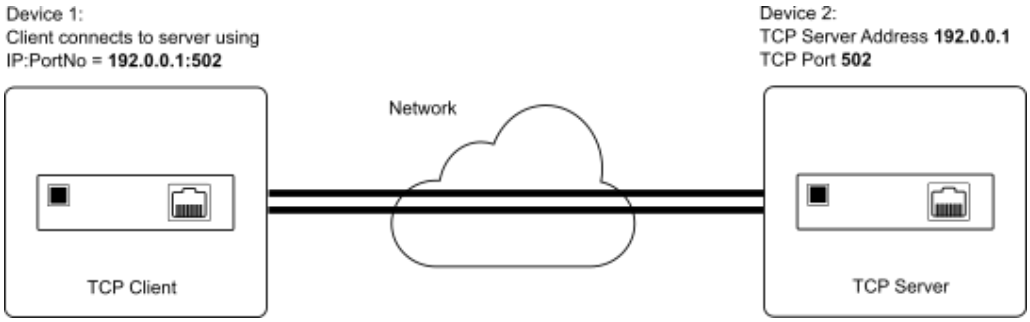
Docklight Scripting tente alors d'établir une connexion avec l'appareil serveur TCP. Une fois la connexion établie, vous pouvez envoyer l'une de vos séquences d'envoi prédéfinies en cliquant sur le bouton Envoyer ----> . Tant que le serveur TCP n'accepte pas la demande de connexion, aucune donnée d'envoi n'est affichée dans la fenêtre de communication.

REMARQUE : si le serveur met fin à la session TCP avant que vous n'ayez sélectionné Run **>** Stop Communication dans Docklight ■ , vous recevrez le message d'erreur "TCP/IP connection closed by the remote computer" et la communication sera interrompue.

REMARQUE : si vous recevez le message d'erreur "IP Address / TCP port in use" lorsque vous démarrez la communication, vérifiez si un autre serveur ou même une autre instance de Docklight Scripting bloque le port. Essayez également de fermer et de redémarrer Docklight Scripting - parfois, la couche de pilotes TCP utilisée par Docklight Scripting ne libère un port TCP que lorsque l'application qui l'utilise est fermée.

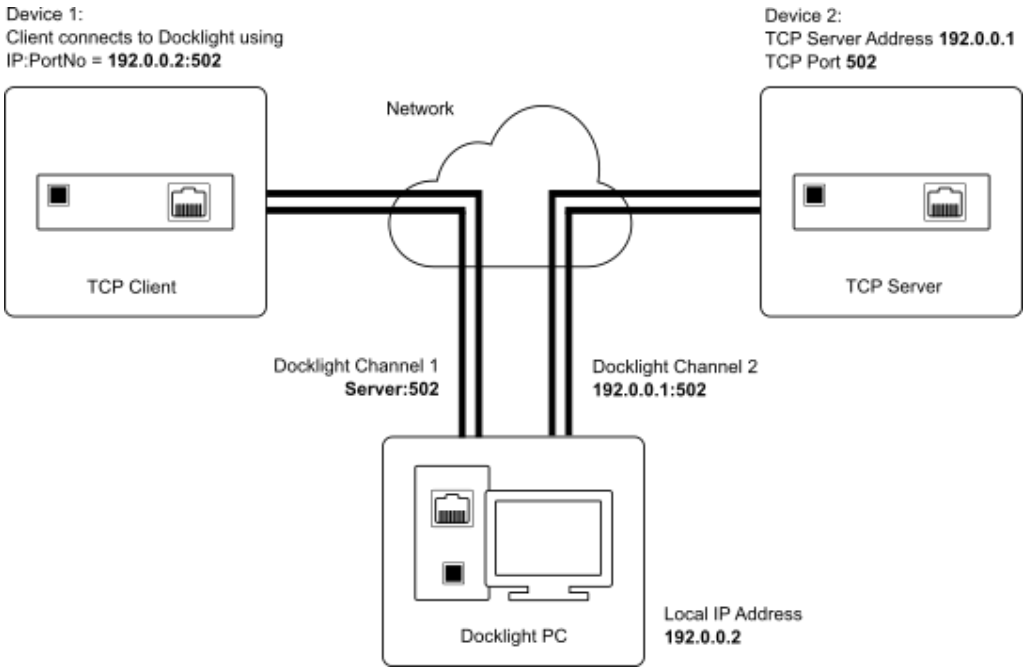
REMARQUE : Même si aucun autre serveur ou client ne bloque un port, il peut s'écouler jusqu'à 4 minutes avant qu'un port soit effectivement libéré et à nouveau disponible. Il s'agit d'une restriction dans la couche de pilote TCP utilisée dans Docklight Scripting et malheureusement, Docklight Scripting ne peut pas contrôler cela.

6.9 Surveillance d'une connexion TCP client/serveur (scripting)



Docklight Scripting vous permet de surveiller et de déboguer une application basée sur TCP aussi facilement que si vous utilisiez des ports et des câbles RS232. Au lieu d'utiliser un câble de surveillance Docklight entre les deux appareils à tester, vous pouvez exécuter Docklight Scripting au sein du réseau et laisser simplement le client (appareil 1) se connecter à Docklight Scripting au lieu du produit compatible réseau (appareil 2).

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



Conditions préalables

L'appareil 1, l'appareil 2 et le PC avec Docklight sont connectés sur un réseau commun (LAN).
Toutes les adresses IP et le numéro de port TCP sont connus.
L'appareil 1 est actuellement configuré pour se connecter à l'appareil 2 et la communication entre les deux appareils fonctionne.

Routage et débogage du trafic TCP

A) Dirigez le trafic grâce à Docklight Scripting
Modifier les paramètres de communication dans l'appareil 1 : L'appareil 1 doit se connecter au Docklight-PC (dans notre exemple, IP 192.0.0.2).

B) Configurer Docklight Scripting comme pont pour la communication entre Device 1 et Device 2

1. Choisissez le menu **Outils > Paramètres du projet...**
2. Sélectionnez le mode de communication **Surveillance (réception uniquement)**
3. Pour le **canal de réception 1**, entrez le mot-clé **SERVER** et le port TCP à écouter (par exemple SERVER:502).
4. Pour le **canal de réception 2,** saisissez l'**adresse IP** et le **numéro de port TCP** pour la connexion avec l'appareil 2 (par ex. 192.0.0.1:502).
5. Confirmez les paramètres et fermez la boîte de dialogue en cliquant sur le bouton **OK**. cliquer.

C) Réalisation du test

Démarrez Docklight Scripting avec **Run > Start Communication**. Laissez l'appareil 1 se connecter et effectuez un test. Docklight Scripting fait office de pont entre les appareils et vous montre toutes les données TCP qui sont transmises entre les appareils dans la fenêtre de communication.

REMARQUE : Docklight Scripting n'autorise pas les connexions multiples sur un port SERVER. Une seule connexion peut être utilisée à la fois. Cela correspond au fonctionnement standard de nombreux Serial Device Server.

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

ASTUCE : Un exemple qui peut être essayé sur n'importe quel ordinateur avec un navigateur web et un accès à Internet est le projet **TCP_Monitoring_HTTP.ptp**, qui se trouve dans le dossier **\Network** du répertoire \ScriptSamples.

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

Exemples et tutoriels

7 Exemples et tutoriels

Ce chapitre décrit deux exemples de projets qui démontrent certaines des fonctions de base de Docklight. Les fichiers de projet Docklight correspondants (fichiers **.**ptp) se trouvent dans le répertoire
\Samples Dossier dans le répertoire d'installation de Docklight (par ex. **C:\Programme\FuH\Docklight V2.3\Samples**).

ASTUCE : le dossier **\Samples** est également accessible via l'écran de bienvenue du Docklight (menu **Aide > Ecran de bienvenue et exemples...**).

REMARQUE : Si vous êtes connecté avec un compte d'utilisateur limité, vous n'avez pas le droit de faire des modifications dans le répertoire des fichiers du programme. Dans ce cas, l'enregistrement d'un fichier de projet ou d'autres données dans le dossier **\Samples provoque** une erreur.

REMARQUE : vous trouverez également d'autres exemples de projets et des conseils d'utilisation dans nos ressources en ligne à l'adresse https://docklight.de/examples/.

7.1 Tester un modem - Projet exemple : ModemDiagnostics.ptp

Le projet Docklight **ModemDiagnostics.ptp** permet d'effectuer un contrôle du modem. Un certain nombre de commandes de diagnostics de modems sont définies dans la liste Séquences d'envoi.

Il s'agit d'un exemple simple de test d'un périphérique série ou d'une implémentation de protocole. L'exemple de projet utilise les paramètres de communication indiqués ci-dessous. Cela devrait fonctionner pour la plupart des modems standard.

Mode de communication	Envoyer/recevoir
Paramètres du port COM	9600 Baud, pas de parité, 8 bits de données, 1 bit d'arrêt

Premiers pas

Utilisez le **Gestionnaire de périphériques Windows** pour déterminer quel **port COM** est un périphérique modem. Ce projet de démonstration peut être utilisé avec n'importe quel modem compatible AT présent sur votre PC,
par exemple un modem intégré pour ordinateur portable ou un pilote de modem GSM ou Bluetooth auquel on peut accéder via un port COM virtuel.

ASTUCE : pour un test simple sans matériel spécifique, ajoutez votre téléphone portable en tant que **périphérique Bluetooth** sur votre PC Windows. Ensuite, trouvez votre téléphone dans la liste **Périphériques et imprimantes** de Windows. Cliquez dessus avec le bouton droit de la souris, sélectionnez **Propriétés** et allez dans l'onglet **Matériel**. Dans la liste **Fonctions du périphérique**, les ports COM liés au modem doivent être répertoriés.

Accédez à la boîte de dialogue **Paramètres du projet...** et assurez-vous d'avoir sélectionné le même port COM pour l'**envoi/la réception sur le canal de communication**.

- Appuyez sur le bouton **Démarrer la communication** dans la barre d'outils.

Essayez d'envoyer l'une des commandes de modem prédéfinies en appuyant sur la touche **Envoyer** Bouton

Vous devriez maintenant recevoir une réponse de votre modem, par exemple "OK" si votre commande a été acceptée, un numéro d'identification du modèle, etc. La réponse varie selon le modèle de modem.

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

Après l'envoi de plusieurs séquences, la fenêtre de communication du Docklight pourrait se présenter comme suit :

```
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>
...
```

Plus d'informations

La liste des séquences d'envoi contient les commandes de modem AT standard suivantes :

Envoyer une séquence	Description / Réponse par modem
ATQ0V1E0	Initialise la requête.
AT+GMM	Identification du modèle (la recommandation ITU V.250 n'est pas prise en charge par tous les modems).
AT+FCLASS= ?	Classes de fax prises en charge par le modem, le cas échéant.
AT#CLS= ?	Indique si le modem prend en charge le jeu de commandes vocales Rockwell.
ATI<n>	Affiche les informations du fabricant pour <n> = 1 à 7. Vous trouverez ici vous trouverez des informations comme la vitesse de connexion, le résultat d'un test de somme de contrôle et les informations sur le modèle. Vérifier les la documentation du fabricant pour les prévisions de Résultats.

Le dossier \Samples contient également un fichier journal **ModemDiagnostics_Logfile_asc.txt**. Si e montre une exécution test au cours de laquelle les séquences d'émission susmentionnées ont été envoyées à un modem réel.

7.2 Réagir à une séquence de réception - Projet exemple : PingPong.ptp

Le projet Docklight **PingPong.ptp** est un exemple simple de la manière de définir et d'utiliser des séquences de réception.

Premiers pas

Affichez la boîte de dialogue **Paramètres du projet...** et sélectionnez un port COM. Placez une boucle de rétroaction simple sur ce port COM :
Reliez la broche 2 (RX) à la broche 3 (TX). Voir RS232 SUB D9 Pinout.

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

Appuyez maintenant sur la touche ----> **Envoyer** pour l'une des deux séquences d'envoi. Le site

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

La communication est lancée et la séquence d'envoi est transmise. Elle est bien sûr immédiatement reçue sur la ligne RX du port COM.

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

Docklight reconnaît que les données entrantes sont l'une des séquences de réception définies. Il exécute alors l'action prédéfinie pour cet événement, à savoir l'envoi d'une autre séquence. En conséquence, Docklight envoie alternativement les séquences d'envoi "Ping" et "Pong".

- Utilisez le bouton **Stop communication** pour arrêter la démo. L'écran de communication du Docklight devrait ressembler à cela :

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

Voir aussi les fichiers journaux correspondants dans le dossier \Samples (**PingPong_Logfile_asc.htm** et **PingPong_Logfile_hex.htm**).

Plus d'informations

Ce projet de démonstration peut également être exécuté dans trois configurations alternatives :

1. Exécuter deux applications Docklight sur le même PC via des ports COM différents. Les deux ports COM sont reliés par un simple câble null-modem.
2. Au lieu d'utiliser deux ports COM RS232 et un câble null-modem, vous pouvez utiliser un null-modem virtuel.
3. Utilisez deux PC et faites fonctionner Docklight sur chaque PC. Connectez les deux PC avec un simple câble null-modem.

ASTUCE : Pour Docklight Scripting, il existe également un exemple de projet apparenté qui utilise une connexion UDP en boucle et ne nécessite pas de ports de données série. Voir le projet **PingPong_UDP_Loopback.ptp** dans le dossier **\Network** du répertoire \ScriptSamples.

7.3 Modbus RTU avec somme de contrôle CRC - Projet d'exemple : ModbusRtuCrc.ptp

Le fichier de projet Docklight **ModbusRtuCrc.ptp** démontre comment la valeur CRC nécessaire pour envoyer une trame Modbus-RTU valide est calculée automatiquement.

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

Le fichier de projet utilise les paramètres de communication indiqués ci-dessous, conformément à la classe d'implémentation Modbus "Basic".

Mode de communication	Envoyer/recevoir
Envoi/réception sur le canal de communication	COM1
Paramètres du port COM	19200 bauds, parité paire, 8 bits de données, 1 Bit d'arrêt

Premiers pas

Ouvrir le fichier de projet **ModbusRtuCrc.ptp** (menu Ouvrir le projet ...). Le fichier se trouve dans le dossier \Samples.
Connectez le port COM du PC à votre réseau Modbus. Ouvrez la boîte de dialogue **Paramètres du projet...** et assurez-vous que vous avez sélectionné le bon port COM pour l'**envoi/la réception sur le canal de communication**.
Cliquez sur le bouton ----> **Envoyer** dans la ligne **Read Input Register Slave= ?...** de la liste des séquences d'envoi
Dans la boîte de dialogue **Paramètres de la séquence d'envoi**, saisissez un numéro d'esclave, par ex. "01" pour l'adressage de l'esclave n° 1.

Après l'envoi des commandes "Read Input Register" aux esclaves 1 - 4, la fenêtre de communication pourrait se présenter comme suit :

```
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
Trame Modbus détectée = 01 04 02 FF FF B8 80
SlaveID=01
Code de fonction=04
Adr/Données=02 FF FF
CRC=B8 80

  Registre d'entrée Réponse : 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
Trame Modbus détectée = 02 04 02 7F 58 DC FA
SlaveID=02
Code de fonction=04
Adr/Données=02 7F 58
CRC=DC FA

  Registre d'entrée Réponse : 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
Trame Modbus détectée = 03 04 02 01 0A 41 67
SlaveID=03
Code de fonction=04
Adr/Données=02 01 0A
CRC=41 67

  Registre d'entrée Réponse : Slave=003

ValueHex=010A 23.09.2019 07:05:51.677 [TX] - 04 04

00 03 00 01 C1 9F
```


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

```
23.09.2019 07:05:51.789 [RX] - 04 04 02 40 00 44 F0
Trame Modbus détectée = 04 04 02 40 00 44 F0
SlaveID=04
Code de fonction=04
```


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

Addr/Data=02 40 00
CRC=44 F0

Registre d'entrée Réponse : Esclave=004 WertHex=4000

Le canal [RX] indique les réponses des esclaves Modbus :
L'esclave 1 a répondu avec la valeur "-1",
L'esclave 2 a répondu "32600",
L'esclave 3 a répondu "266" et
L'esclave 4 a répondu "16384".

REMARQUE : si vous utilisez l'exemple Docklight Modbus sur un bus RS485 et que vous ne voyez pas de réponse de l'appareil, vérifiez si votre interface matérielle RS485 commute automatiquement entre les modes émission et réception ou si vous devez utiliser l'option RS485 Transceiver Control.

Plus d'informations

Le calcul du CRC est effectué selon les spécifications pour la transmission de ligne série Modbus (mode RTU). La fonction de somme de contrôle de Docklight prend en charge un modèle "CRC-MODBUS" à cette fin. Pour plus d'informations générales sur la mise en œuvre du calcul des sommes de contrôle, voir Calcul et validation des sommes de contrôle.
Si vous ne disposez pas de dispositifs Modbus esclaves, vous pouvez utiliser un simulateur de logiciel. Voir www.plcsimulator.org/ comme mentionné à l'origine sur www.modbus.org, "Modbus Technical Resources", "Modbus Serial RTU Simulator". Ce simulateur a été utilisé pour générer les données d'exemple présentées ci-dessus.

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

Exemples et tutoriels (scripting)

8 Exemples et tutoriels (scripting)

Ce chapitre décrit des exemples de scripts qui démontrent certaines des possibilités de Docklight Scripting. Les fichiers de script Docklight correspondants (fichiers .pts) et d'autres fichiers associés se trouvent dans le dossier **\ScriptSamples** du répertoire d'installation de Docklight Scripting (par ex. **C:\Programme\FuH\Docklight Scripting V2.3\ScriptSamples**).

ASTUCE : Le dossier **\ScriptSamples** est également accessible via l'écran de bienvenue de Docklight Scripting (menu **Aide > Ecran de bienvenue et exemples...**).

REMARQUE : Si vous travaillez avec un compte d'utilisateur ayant un accès limité au système, il se peut que vous n'ayez pas le droit d'enregistrer dans le répertoire des fichiers du programme. Dans ce cas, l'enregistrement d'un fichier de projet ou d'autres données dans le dossier **\ScriptSamples** provoque une erreur.

REMARQUE : vous trouverez d'autres exemples de scripts, de projets et de notes d'application dans nos ressources en ligne à l'adresse https://docklight.de/examples/.

8.1 Tests automatisés de modems - exemple de script :

ModemScript.pts

Le script Docklight **ModemScript.pts** et le fichier de projet associé **ModemATCommands.ptp** montrent comment utiliser un script Docklight pour un test automatisé ou une tâche de configuration avec interaction de l'utilisateur.

Le fichier de projet utilise les paramètres de communication indiqués ci-dessous. Cela devrait fonctionner pour la plupart des modems standard.

Mode de communication	Envoyer/recevoir
Envoi/réception sur le canal de communication	COM3
Paramètres du port COM	9600 Baud, pas de parité, 8 bits de données, 1 bit d'arrêt

Premiers pas

Connectez le modem à un port COM libre, par exemple COM1, et mettez-le sous tension. La démo peut également fonctionner sur un ordinateur portable avec un modem intégré. Dans de nombreux cas, vous trouverez le modem intégré de votre ordinateur portable sur COM3, de sorte que vous pouvez essayer d'exécuter le script d'exemple sans modifier les paramètres du projet.

Allez dans la boîte de dialogue **Paramètres du projet...** et assurez-vous que vous avez sélectionné le même port COM pour l'**envoi/la réception sur le canal de communication**.

Appuyez sur le bouton **Exécuter le script** de la barre d'outils.

Saisissez la plage de commandes AT à tester ou acceptez simplement la valeur par défaut en appuyant sur la **touche "Entrée"**.

Le script établit alors une connexion avec le modem et exécute le jeu d'instructions AT. Une fois le test terminé avec succès, la fenêtre de communication du Docklight pourrait ressembler à ceci :

```
Attendre le modem ...
```


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

```
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

Vérifier le jeu d'instructions AT...

```
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>
Construit le 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>
Allemagne<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] - Description de
  l'état<CR><LF
-------------------------------------------------------------------------------<CR><LF>
Dernier
        ConnexionInconnu<CR><LF> Taux de
        porteuse d'émission initial0<CR><LF>
        Taux de porteuse d'émission initial
Taux de porteuse en réception0<CR><LF> Taux de
porteuse en émission finale9600<CR><LF> Taux
de porteuse en réception finale9600<CR><LF>
Résultat de la négociation du protocoleAucun<CR><LF>
Résultat de la compression des données
        AUCUNE<CR><LF>Rapport signal/bruit
estimé                          (dB)
00<CR><LF> Niveau de puissance du signal reçu
(-dBm) 00<CR><LF> Niveau de puissance du
signal émis ( - d B m ) 10<CR><LF>Délai de
circulation (msec) 1000<CR><LF>
Niveau de l'écho proche       (-dBm) 00<CR><LF>
FarLevel                      (-dBm)00<CR><LF>
Transmit Frame
                             Compte0<CR><LF>
Transmit Frame Error Count 0<CR><LF> Nombre
de trames reçues 0<CR><LF> Nombre d'erreurs
de trames reçues 0<CR><LF> Réapprentissage
via modem local 0<CR><LF> Renégociation par
modem distant
                                   Modem0<CR
><LF> Renégociation du taux par modem local
0<CR><LF> Renégociation du taux par modem
distant 0<CR><LF> Cause de l'interruption de
l'appel                            0<CR><LF>
Signalisation du bit Robbed0<CR><LF>Perte
numérique                     (dB)00<CR><LF>
ID du serveur distant              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>
ERREUR<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>
ERREUR<CR><LF

 Résultats :
 13 commandes AT valides trouvées.
 2 La ou les commandes AT n'ont pas fonctionné.
```

8.2 Démarrage à partir de la ligne de commande - exemple de script :

LogStartupScript.pts

Le script Docklight **LogStartupScript.pts**, le fichier de projet correspondant **LogStartupSettings.ptp** et le fichier batch **LogStartup.bat** montrent comment lancer Docklight à partir de la ligne de commande, créer un fichier journal selon des paramètres prédéfinis et lancer automatiquement la communication.

Le fichier de projet utilise les paramètres de communication indiqués ci-dessous.

Mode de communication	Surveillance (réception uniquement)
Canal de réception 1	COM1
Canal de réception 2	COM3
Paramètres du port COM	9600 Baud, pas de parité, 8 bits de données, 1 bit d'arrêt

Premiers pas

Lancez le fichier batch **LogStartup.bat** à partir d'une ligne de commande ou allez dans le répertoire \ScriptSamples et exécutez **LogStartup.bat** en double-cliquant sur le fichier.

Docklight Scripting est lancé, un fichier journal ASCII **C:\DocklightScripting_Logfile_asc.txt** est créé et la communication est immédiatement établie.

Utilisez **Maj+F6** pour arrêter l'exécution du script et fermer les ports de communication et le fichier journal.

REMARQUE : cet exemple nécessite une licence logicielle pour la version standard de Docklight, car il utilise la fonction de journalisation de Docklight. Une licence Docklight Scripting n'est pas nécessaire pour exécuter l'exemple.

Plus d'informations

Le fichier batch **LogStartup.bat** contient la ligne suivante :

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

Cela lance Docklight Scripting, ouvre le fichier de script **LogStartupScript.pts** et l'exécute immédiatement (option **-r**). Le script **LogStartupScript.pts** contient les commandes suivantes :

```
' LogStartupScript.pts
' Démarrer la journalisation et la
communication DL.OpenProject
"LogStartupSettings
' Créer (ou joindre) un fichier journal ASCII DL.StartLogging
"C:\DocklightScripting_Logfile", True, "A
DL.StartCommunication
```


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

```
' Maintenir la communication et la journalisation
jusqu'à ce que l'utilisateur arrête Do
```


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

```
    DL.Pause 1 ' (la pause réduit la charge du CPU au
ralenti) Boucle
```

Les paramètres de communication utilisés dans **LogStartupSettings.ptp ne** sont qu'un exemple. Si vous avez besoin d'autres paramètres, vous devez ouvrir le fichier de projet, modifier les paramètres du projet et enregistrer les modifications. Il est recommandé de copier tous les fichiers associés (**LogStartupScript.pts**, **LogStartupSettings.ptp** et **LogStartup.bat**) à un autre endroit avant d'effectuer toute modification. Dans ce cas, vous devez indiquer le chemin d'accès complet à l'application **Docklight_Scripting.exe** à l'intérieur du fichier de configuration. Spécifier le fichier **.bat.**

8.3 Manipulation d'un flux de données RS232 - exemple de script :

CharacterManipulation.pts

Le script Docklight **CharacterManipulation.pts** démontre comment un flux de données RS232 peut être manipulé à l'aide de la procédure événementielle DL_OnReceive(). Toutes les données reçues sur la ligne RX sont retransmises sur la ligne TX, mais certains caractères sont remplacés.

Premiers pas

Ouvrir le fichier de projet **CharacterManipulationPrj.ptp** (via le menu **Open Project ...**) et le fichier de script **CharacterManipulation.pts** (via le menu **Open Script ...**). Les fichiers se trouvent dans le dossier \ScriptSamples.
Affichez la boîte de dialogue **Paramètres du projet...** et sélectionnez un port COM. Appuyez sur la touche BoutonExécuter le script dansl a barre d'outils.
Lancez une deuxième instance de Docklight et ouvrez le fichier de projet **CharacterManipulationTest.ptp**.
Choisissez un autre port COM pour cette deuxième instance de Docklight (ou utilisez un autre ordinateur).
Connectez les deux ports COM avec un simple câble null-modem. Ou utilisez plutôt un null-modem virtuel.
Utilisez le bouton ----> **Envoyer** sur la deuxième instance de Docklight pour envoyer le jeu de test.

L'écran de communication de la deuxième instance de Docklight devrait ressembler à quelque chose comme ça :

```
2/21/2009 11:56:57.343 [TX] - Ceci est un test pour l'exemple
de script de manipulation de caractères
2/21/2009 11:56:57.502 [RX] - Ceci est un Desd pour le personnage
manibuladion samble scribd
```

Chaque "T" est remplacé par un "D" et chaque "P" par un "B". (Les visiteurs de la région de Nuremberg, où notre entreprise est implantée, remarqueront peut-être que les dialectophones y font quelque chose de similaire...)

Plus d'informations

L'exemple utilise la procédure événementielle **DL_OnReceive()** pour effectuer des opérations supplémentaires chaque fois qu'un nouveau caractère est reçu. Pour plus de détails, voir Évaluation des données de séquence reçues.

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

Les performances d'un processeur caractère par caractère dans Docklight Scripting sont assez limitées. Vous pouvez facilement le surcharger en envoyant un flux de données constant. Dans ce cas, Docklight affiche un commentaire dans le

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

Fenêtre de communication, par exemple
DOCKLI GHT se manifeste : Tampon d'entrée via la connexion à COM1
Pour des raisons de performance, l'affichage de toutes les données TX et RX en
Manipulation de caractèresPrj.ptp

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

Si vous envisagez d'écrire un manipulateur pour votre propre protocole, vous devriez envisager une approche basée sur les paquets, dans laquelle une séquence de réception peut reconnaître un paquet entier ou une commande de votre protocole. Cela permet des débits de données plus élevés que l'approche basée sur présentée ici.

8.4 Communication TCP/IP - Exemples de projets PingPong_TCP_Server/Client.ptp

Les fichiers de projet **PingPong_TCP_Server.ptp** et **PingPong_TCP_Client.ptp** dans le répertoire Les dossiers \ScriptSamples\NNetwork montrent comment utiliser Docklight Scripting comme serveur TCP ou client TCP et comment échanger des données.

Les exemples montrent comment un serveur et un client peuvent fonctionner sur le même ordinateur. Le nom de réseau **utilisé est LOCALHOST, qui se réfère** toujours à l'ordinateur sur lequel Docklight fonctionne.

Premiers pas

Ouvrez le projet **PingPong_TCP_Server.ptp** dans Docklight Scripting. Appuyez sur le bouton **Démarrer la communication dans** la barre d'outils.
Si vous utilisez un pare-feu personnel sur votre PC, celui-ci vous indiquera probablement que Docklight Scripting souhaite agir en tant que serveur. Confirmez cela et autorisez-le si nécessaire.
Démarrez une deuxième instance de Docklight Scripting et ouvrez la Projet **PingPong_TCP_Client.ptp**

Dans cette instance "client", appuyez sur le bouton **Envoyer** pour la séquence "Ping". Si vous utilisez un pare-feu personnel sur votre PC, autorisez Docklight Scripting, se connecter à Internet.

Le docklight "client" se connecte alors au docklight "serveur" et les données sont échangées comme si les deux instances du docklight étaient reliées par un câble sériel null-modem.

La fenêtre de communication du côté client affiche maintenant les messages suivants :

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

Côté serveur, ils ressemblent alors à quelque chose comme ça :

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


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

Référence

9 Référence

9.1 Menu et barre d'outils (scripting)

REMARQUE : des touches de raccourci sont disponibles pour les fonctions courantes des menus et des barres d'outils.

Menu "Fichier

Nouveau projet
Fermez le projet Docklight actuel et créez-en un nouveau.

Projet ouvert ...
Fermez le projet Docklight en cours et ouvrez un autre projet.

Importer un listing de séquences ...
Importez toutes les séquences d'émission et de réception d'un deuxième projet Docklight.

Enregistrer le projet / Enregistrer le projet sous ...
Enregistrez le projet Docklight actuel.

Imprimer le projet ...
Imprime les données du projet, c'est-à-dire la liste des séquences d'émission et de réception définies. Les séquences sont imprimées dans la même représentation (ASCII, HEX, décimale ou binaire) que celle utilisée dans la fenêtre principale du Docklight. La représentation peut être sélectionnée via la boîte de dialogue Options.

Imprimer la communication ...
Imprime le contenu de la fenêtre de communication. Les données de communication sont imprimées dans la même représentation que celle qui est actuellement visible dans la fenêtre de communication.

Sortie
Quittez Docklight.

Menu Édition

Modifier la liste des séquences d'envoi ...
Modifiez la liste des séquences d'envoi, c'est-à-dire ajoutez de nouvelles séquences ou supprimez des séquences existantes.

Modifier la liste des séquences de réception ...
Modifier la liste des séquences de réception, c'est-à-dire ajouter de nouvelles séquences ou supprimer des séquences existantes.

Permutation des listes de suivi d'envoi et de réception
Convertir toutes les séquences d'émission en séquences de réception et inversement.

Rechercher une séquence dans la fenêtre de communication...
Rechercher une séquence spécifique parmi les données affichées dans la fenêtre de communication. Voir la fonction Rechercher une séquence.

Une fenêtre de communication claire
Effacer le contenu de la fenêtre de communication. Ceci est valable pour les quatre représentations (ASCII, HEX, décimale, binaire) de la fenêtre de communication.

Exécuter le menu

Début de la communication

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

Ouvrez les ports de communication et activez le transfert de données en série.

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

Mettre fin à la communication
Arrêter la transmission de données en série et fermer les ports de communication.

Menu "Outils

Démarrer l'enregistrement des communications ...
Créez un ou plusieurs nouveaux fichiers journaux et commencez à enregistrer les données sérielles entrantes/sortantes. Voir Consignation et analyse d'un test.

Arrêter la journalisation des communications
Arrête la journalisation et ferme le(s) fichier(s) de journalisation actuellement ouvert(s).

Démarrer le mode Instantané
Attendre une séquence de déclenchement et prendre un instantané. Voir Interception d'une séquence spécifique...

Quitter le mode Instantané
Annulation d'un instantané et réactivation de l'affichage de la fenêtre de communication.

Clavier Console On
Activez la console de clavier pour envoyer directement les saisies au clavier.

Clavier Console Désactivé
Désactiver la console de clavier.

Minimiser/Restaurer Documentation et script
Réduisez la taille de la zone de documentation et de script ou ramenez-la à sa taille normale.

Réduction/restauration des listings de séquences
Réduisez les listes de l'ordre d'envoi/réception ou ramenez-les à leur taille normale.

Paramètres du projet...
Sélectionnez les paramètres actuels du projet (ports COM, vitesse de transmission, ...).

Les options...
Sélectionnez les options générales du programme (par ex. mode d'affichage, date / horodatage).

Options expert...
Sélectionnez les options du programme expert pour les utilisateurs avancés et les applications spéciales (par exemple, haute précision de surveillance).

Menu "Scripting

Exécuter / continuer le script
Exécutez le code dans l'éditeur de script.

Arrêter le script
Arrêter un script en cours d'exécution.

Pause Script
Interrompt un script en cours.

Nouvelle police
Fermez le script Docklight actuel et créez-en un nouveau.

Ouvrir le script ...
Fermez le script Docklight actuel et ouvrez un autre script.

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

Enregistrer le script / Enregistrer le script sous ...
Enregistrez le script Docklight actuel.

Personnaliser / Editeur externe...
Utiliser un éditeur externe au lieu de l'éditeur intégré de Docklight.

9.2 Dialogue de la page : Modifier la séquence d'envoi

Dans cette boîte de dialogue, vous pouvez définir de nouvelles séquences d'envoi et modifier des séquences existantes (voir aussi Modifier et gérer des séquences).

Index
L'index de la séquence affichée ci-dessous. La première séquence a un index de 0 (zéro).

1 - Nom
Nom unique pour cette séquence (par exemple "Régler le volume du modem"). Ce nom sert à référencer la séquence. Il ne s'agit pas des données envoyées par le port série. Voir "2 - Séquence" ci-dessous.

2 - Séquence
Chaîne de caractères transmise par l'interface série.

ASTUCE : Pour transférer des blocs de données plus importants, qui dépassent la taille maximale de la séquence, utilisez la commande de script DL.UploadFile.

ASTUCE : des caractères de fonction spéciaux sont disponibles pour créer des retards entre les caractères, définir des signaux de handshake et des bits de parité ou définir un état d'interruption.

3 - Paramètres supplémentaires
Répéter - Activez l'option "Envoyer périodiquement..." pour définir une séquence qui sera envoyée périodiquement. Il est possible de définir un intervalle de temps compris entre 0,01 seconde et 9999 secondes.

REMARQUE : Le temps de référence Windows utilisé à cette fin a une précision limitée. Les intervalles de temps < 0,03 seconde ne sont généralement pas précis.

Somme de contrôle - Calcul automatique de tout type de somme de contrôle, y compris tout type de norme CRC comme Modbus, CCITT, CRC32.

ASTUCE : pour une vue d'ensemble générale, voir Calcul et validation des sommes de contrôle et Spécification des sommes de contrôle pour le format de texte utilisé pour définir une somme de contrôle.

Cartes sauvages
Les caractères de remplacement peuvent être utilisés pour insérer dans une séquence d'envoi des paramètres que vous souhaitez insérer manuellement à chaque fois que la séquence est envoyée. Vous trouverez des détails et des exemples dans la section Envoyer des commandes avec des paramètres.

Combinaisons de touches pour les caractères de contrôle
L'utilisation de raccourcis clavier est d'une grande aide lors de l'édition d'une séquence contenant à la fois des caractères imprimables (lettres A-z, chiffres 0-9, ...) et des caractères de contrôle non imprimables (code ASCII 0 à 31). Les raccourcis clavier prédéfinis sont
Ctrl+entrée pour retour chariot / <CR> / code décimal 13
Ctrl+Shift+Enter pour le saut de ligne / <LF> / code décimal 10

Utilisez Options... --> Raccourcis de caractères de contrôle pour définir d'autres raccourcis que vous jugez utiles.

Documentation de la séquence

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

Ajoutez ici une documentation sur cette séquence. Cette documentation s'affiche également dans la fenêtre principale r lorsque vous sélectionnez la séquence dans la liste Séquences d'envoi.

9.3 Dialogue de la page : Modifier la séquence de réception

Dans cette boîte de dialogue, vous pouvez définir de nouvelles séquences de réception et modifier les séquences existantes (voir aussi Modifier et gérer les séquences).

Index
L'index de la séquence affichée ci-dessous. La première séquence a un index de 0 (zéro).

1 - Nom
Nom unique pour cette séquence (par exemple "Recevoir un ping"). Ce nom sert à référencer la séquence. Il ne s'agit pas de la séquence reçue via le port série. Voir "2 - Séquence" ci-dessous.

2 - Séquence
La chaîne de caractères qui doit être reconnue par Docklight dans les données série entrantes.

ASTUCE : des caractères de fonction spéciaux sont disponibles pour détecter les retards entre les caractères, évaluer les changements de signal de poignée de main ou détecter un état d'interruption.

3 - Action
La ou les actions à effectuer lorsque Docklight détecte la séquence définie ci-dessus.

Vous avez le choix entre les actions suivantes :

Réponse - Envoyez l'une des séquences d'envoi après réception de la séquence. Seules les séquences d'envoi qui ne contiennent pas de caractères génériques peuvent être utilisées comme réponse automatique. **Commentaire** - Après réception de la séquence, insérez un commentaire défini par l'utilisateur dans la fenêtre de communication (et le fichier journal, le cas échéant). Différentes macros de commentaires sont disponibles pour la création de textes de commentaires dynamiques.
Déclencheur - déclenche un instantané lorsque la séquence est détectée. Il s'agit d'une fonction avancée décrite dans la section Capture d'une séquence spécifique...
Stop - Arrête la communication et met fin au test.
Somme de contrôle - Effectuez une validation automatique d'une somme de contrôle, y compris tous les types de normes CRC comme Modbus, CCITT, CRC32. Définissez la spécification de la somme de contrôle, ainsi que ce qui doit être fait avec le résultat : **Detect Checksum OK** - les données reçues doivent avoir la même somme de contrôle que la valeur calculée par Docklight.
Somme de contrôle Faux - le contraire. Une somme de contrôle non concordante représente une "concordance de séquence".
Les deux OK/faux - la séquence est toujours reconnue. La zone de somme de contrôle contient tous les "1" ASCII (HEX 31) pour une somme de contrôle correspondante, ou les "0" ASCII (HEX 30) pour une somme de contrôle erronée.

ASTUCE : pour une vue d'ensemble générale, voir Calcul et validation des sommes de contrôle et Spécification des sommes de contrôle pour le format texte utilisé pour définir une somme de contrôle.

Cartes sauvages
Les caractères jokers peuvent être utilisés pour vérifier les séquences qui ont une partie variable avec des valeurs changeantes (par exemple, les valeurs de mesure ou d'état). Voir la section Vérification des séquences contenant des caractères aléatoires pour plus de détails et d'exemples.

Abréviations pour les caractères de contrôle

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

L'utilisation de raccourcis clavier est d'une grande aide lors de l'édition d'une

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

Séquence contenant à la fois des caractères imprimables (lettres A-z, chiffres 0-9, ...) et des caractères de contrôle non imprimables (code ASCII 0 à 31). Les combinaisons de touches prédéfinies sont

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

Ctrl+entrée pour retour chariot / <CR> / code décimal 13
Ctrl+Shift+Enter pour le saut de ligne / <LF> / code décimal 10

Utilisez Options... --> Raccourcis de caractères de contrôle pour définir d'autres raccourcis que vous jugez utiles.

Documentation de la séquence
Ajoutez ici une documentation sur cette séquence. Cette documentation s'affiche également dans la fenêtre principale r lorsque vous sélectionnez la séquence dans la liste Séquences d'envoi.

9.4 Dialogue de gestion : Démarrer la journalisation / Créer un ou plusieurs fichiers journaux

Menu **Outils > Démarrer le protocole de**

communication ... Format du fichier journal
Les formats de protocole disponibles sont le texte simple (.txt), le HTML pour les navigateurs web (.htm) ou le format RTF Rich Text pour *Microsoft* Word ou Wordpad (.rtf).

Un simple fichier texte (.txt) est un bon choix si vous vous attendez à ce que vos fichiers journaux soient très volumineux.

ASTUCE : l'éditeur Notepad intégré dans *Windows* peut être très lent lors de l'ouverture et de l'édition de gros fichiers. Nous vous recommandons le populaire éditeur open source Notepad++, disponible sur http://notepad-plus.sourceforge.net, qui est une alternative beaucoup plus rapide et plus puissante.

REMARQUE : il n'y a pas de limite de taille pour les fichiers journaux de Docklight, hormis les limites de votre PC Windows. Nous avons testé avec succès Docklight dans des applications de surveillance à long terme / à haut volume et avons créé des fichiers journaux de plusieurs Go sans aucun problème de stabilité.

Les fichiers HTML (.htm) sont plus pratiques à analyser, car ils contiennent tout le formatage visuel des fenêtres de communication Docklight (couleurs, caractères gras et italiques). Cependant, la taille du disque dur est plus importante pour un tel fichier que pour un format texte pur, et les fichiers HTML volumineux ralentissent les navigateurs web les plus courants.

ASTUCE : Si vous avez des exigences particulières concernant le format de sortie, vous pouvez adapter la sortie HTML.

RTF Rich Text Format (.rtf) est un bon choix aussi bien pour les petits que pour les grands fichiers journaux contenant du texte formaté - *Microsoft* Word et Wordpad peuvent tous deux naviguer rapidement dans des fichiers plus grands sans devenir insensibles.

REMARQUE : en raison des spécificités du format de document RTF, Docklight ne peut pas joindre efficacement de nouvelles données à un fichier journal existant, mais doit d'abord créer une copie temporaire du journal existant, ce qui peut entraîner des retards supplémentaires. Il n'est pas non plus possible de joindre de nouvelles données de journal avec des couleurs et des paramètres de police différents de ceux utilisés au début du fichier.

Répertoire et nom de base du fichier journal
Sélectionnez ici le répertoire et le nom de fichier de base pour le(s) fichier(s) journal. Le chemin d'accès réel au fichier, utilisé pour chaque représentation de fichier journal, est affiché dans les champs de texte du cadre "Représentation de fichier journal".

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

Mode d'écrasement/d'ajout
Sélectionnez "Joindre de nouvelles données" si vous ne voulez pas que Docklight utilise les données existantes.

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

écrase les fichiers journaux. Docklight insère alors un message "Start Logging / Stop Logging" à l'ouverture/fermeture des fichiers journaux. De cette manière, il est possible de savoir quand un fichier journal individuel a commencé ou s'est terminé, même en mode "ajout".

Représentation

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

Un fichier de protocole distinct peut être créé pour chaque représentation de données (ASCII, HEX, ...). Choisissez au moins une représentation. Les fichiers journaux ont une extension de fichier ".txt" ou ".htm". Docklight ajoute en outre le type de représentation au nom de fichier afin de distinguer les différents fichiers journaux. Par exemple, si l'utilisateur indique "Test1" comme nom de fichier, le fichier journal ASCII s'appelle "Test1_asc.txt", tandis que le fichier journal HEX s'appelle "Test1_hex.txt".

Désactiver la fenêtre de communication pendant la journalisation
Si vous surveillez une connexion de communication à haut débit ou si vous faites fonctionner Docklight sur un ordinateur lent, il se peut que Docklight ne puisse pas traiter toutes les données transmises, voire qu'il se bloque (aucune réponse aux entrées de l'utilisateur).
Cette option permet de désactiver la sortie de la fenêtre de communication pendant que les données sont enregistrées dans un fichier. Docklight fonctionne alors beaucoup plus rapidement, car le formatage et la mise à jour continus de l'affichage nécessitent un temps de CPU considérable.

REMARQUE : Pour plus d'informations sur les applications à grande vitesse, consultez également la section Comment augmenter la vitesse de traitement...

9.5 Dialogue de la page : Personnaliser la sortie HTML

(via le menu **Outils > Démarrer le protocole de communication ...** puis sélectionnez **Fichier HTML pour navigateur web (.htm)** et cliquez sur **Personnaliser la sortie HTML**)

Dans cette boîte de dialogue, vous pouvez modifier l'apparence des fichiers journaux HTML en modifiant le code de modèle HTML que Docklight utilise pour générer les données du journal HTML.

Vous devez avoir une compréhension de base des documents HTML et des attributs de style CSS. Nous vous recommandons www.htmldog.com (anglais) ou www.selfhtml.org (allemand) pour un aperçu rapide de ces sujets.

Modèle d'en-tête HTML
L'en-tête du document HTML. Ici, vous pouvez modifier la police de caractères utilisée pour les données du fichier journal en utilisant les propriétés de style CSS suivantes :

Attribut de style CSS	Description et exemple

font-family	Définit une ou plusieurs polices (ou : catégories de polices) que le navigateur HTML doit utiliser pour imprimer un texte. Si le navigateur ne supporte pas la première police, il essaie la deuxième, etc. La dernière police définit généralement une catégorie de police générique que chaque navigateur supporte. Exemples :
	`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

Taille de la police	Indique la taille de la police. Les tailles absolues et relatives sont toutes deux possibles. Exemples de tailles de police absolues :
	`font-size:12pt`
	`font-size:xx-small`
	`font-size:x-small`
	`font-size:medium`
	`font-size:large`
	`font-size:x-large`
	`font-size:xx-large`
	Exemples de tailles de police relatives (par rapport au HTML parent)
	Élément)
	`font-size:plus petit`
	`font-size:plus grand`

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

	Taille `de la police:90`

REMARQUE : Utilisez le point-virgule (" ;") comme séparateur entre deux attributs de style CSS différents, par exemple
`font-family:sans-serif ; font-size:small`

REMARQUE : Docklight insère des balises HTML supplémentaires <u> (souligné), <i> (italique) et <b> (gras) lorsque ces options de formatage sont activées dans les paramètres d'affichage. Il n'est pas nécessaire d'utiliser l'attribut **font-style** ou **font-weight** pour créer ces effets.

Modèle de pied de page HTML
Ajoute du texte supplémentaire en bas de page et ferme le document HTML.

Modèle d'élément de données
Pour chaque nouvelle information dans le fichier journal (données du canal 1, données du canal 2 ou un texte de commentaire), un nouveau fichier journal HTML est créé.
Ajout d'une balise avec une autre couleur de texte.

Le code du modèle pour l'en-tête, le pied de page et les données contient des espaces réservés spécifiques à Docklight, qui ne doivent pas être supprimés :

Espace réservé	Description
%BACKCOLOR% (COULEUR DE FOND)	La couleur de fond telle que sélectionnée dans les paramètres d'affichage
%HEADERMSG%	Texte d'en-tête au début du fichier journal
%FOOTERMSG%	Texte de bas de page à la fin du fichier journal
%DATA%	une partie des données du fichier journal : données du canal 1, données du canal 2 ou un texte de commentaire
%TEXTCOLOR% (COULEUR DU TEXTE)	La couleur de texte à appliquer à %DATA%, telle que sélectionnée dans les paramètres d'affichage

Lors de la création d'un fichier journal, Docklight remplace les caractères génériques par les paramètres d'affichage actuels et les données de communication réelles.

9.6 Dialogue de la séquence : Rechercher une séquence

Menu **Édition > Rechercher une séquence dans la fenêtre de communication...**

La fonction **Rechercher une séquence** recherche le contenu de la fenêtre de communication. La recherche est effectuée dans l'onglet actuellement sélectionné de la fenêtre de communication (ASCII, HEX, décimal ou binaire). Mais vous pouvez aussi définir votre séquence de recherche dans n'importe quelle autre représentation.

La recherche dans les fenêtres de communication n'est possible que si la communication est arrêtée.

Vous pouvez rechercher tout ce qui est déjà défini comme séquence d'envoi ou de réception, ou vous pouvez définir votre propre séquence de recherche.

REMARQUE : Si vous recherchez une séquence dans la fenêtre de communication ASCII, veuillez tenir compte des restrictions suivantes :

La fonction **Rechercher une séquence n'**est pas en mesure de trouver des séquences contenant des caractères de contrôle non imprimables (code décimal ASCII < 32) ou d'autres caractères spéciaux (code décimal > 127). Cela est dû au type d'affichage ASCII. Utilisez plutôt l'onglet de la fenêtre de communication HEX ou décimale pour la recherche.

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

En mode ASCII, la fonction **"Find Sequence"** traite les horodatages et tous les autres commentaires comme des données de communication régulières. En mode HEX / décimal /

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

mode binaire, toutes les informations supplémentaires sont ignorées tant qu'elles ne ressemblent pas à une valeur d'octet de caractères.

9.7 Dialogue de la page : Paramètres séquence d'envoi

Saisissez ici une ou plusieurs valeurs pour une séquence d'envoi avec des caractères génériques. Comme dans la boîte de dialogue "Modifier la séquence d'envoi/réception", vous pouvez utiliser des raccourcis clavier ou des fonctions du presse-papiers.

Paramètre n
Une séquence d'envoi peut contenir un nombre quelconque de caractères génériques. Chaque ensemble de caractères génériques successifs est considéré comme un paramètre distinct. La valeur de chaque paramètre est saisie séparément.

Caractères minimaux requis
Pour chaque caractère générique ' ?, un seul caractère est nécessaire. Par conséquent, le nombre minimal de caractères requis est égal au nombre de caractères génériques '?' dans un paramètre.

REMARQUE : pendant que la boîte de dialogue Paramètres de la séquence d'envoi est affichée, toute la communication sérielle est interrompue. Le Docklight ne reçoit pas de données et n'envoie pas de séquences d'envoi (périodiques).

9.8 Dialogue de gestion : Paramètres du projet - Communication

Menu **Outils > Paramètres du projet... | Communication**

Mode de communication

Envoyer/recevoir
Docklight fonctionne à la fois comme émetteur et comme récepteur de données série. Ce mode est utilisé pour tester la fonctionnalité ou l'implémentation du protocole d'un périphérique série ou pour simuler un périphérique série.
Conventions de dénomination : Les données reçues (RX) sont affichées et traitées comme "canal 1", les données envoyées (TX) sont affichées comme "canal 2".

Surveillance
Docklight reçoit des données sérielles sur deux canaux de communication différents. Ce mode est utilisé par exemple lors de la surveillance de la communication entre deux appareils.
Conventions d'appellation : Les données série de l'appareil 1 sont "canal 1", les données de l'appareil 2 sont "canal 2".

Canaux de communication - ports COM série, Docklight TAP/VTP, réseau TCP/UDP, HID, Named Pipes
Dans Docklight Scripting, un canal de communication peut être configuré en tant que Port COM série (RS232, RS422 ou RS485), port TAP

- pour la surveillance de la lampe d'accueil Tap
- Connexion VTP pour Docklight Tap Pro ou Tap 485 Surveillance
- Socket de communication réseau pour TCP ou UDP
- Connexions HID pour les appareils USB ou Bluetooth

- Tubes désignés Client

Les paramètres suivants peuvent être utilisés :

Réglage / Exemples	Description

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

COMxxx	Le canal est connecté à un port COM série.

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

COM1 COM256	Utilisez la liste déroulante pour afficher tous les ports COM du système d'exploitation Windows disponibles sur votre PC.
TAPx TAP0 TAP1	Le canal est connecté à l'une des directions de données de surveillance de Docklight Tap. Les connexions TAP ne sont disponibles que si le mode de communication est réglé sur "Surveillance", si le Docklight Tap est branché et si les pilotes du périphérique USB Docklight Tap sont correctement installés.
VTPx VTP0 VTP1	Le canal est relié à l'une des directions de données de surveillance Tap Pro / Tap 485, de manière similaire à l'application Docklight Tap avec les paramètres TAPx.
RemoteHost:*Port distant* 192.168.1.100:10001 NIC.COM:80 LOCALHOST:504	Le canal fait office de client TCP. Lorsque la communication démarre, une connexion est établie avec l'hôte et le port TCP indiqués. Pour *RemoteHost*, vous pouvez entrer - une adresse IP4, par exemple 192.168.1.100 ▪ un nom d'hôte, par exemple NIC.COM (pour accéder à un serveur sur l'Internet) ou le nom *Windows* NetBIOS pour un autre ordinateur de votre réseau local. ▪ le mot-clé LOCALHOST, qui se réfère toujours à l'ordinateur sur lequel Docklight fonctionne. Cela équivaut à en utilisant l'adresse IP de bouclage 127.0.0.1.
SERVER:*LocalPort* SERVEUR:10001 SERVEUR:80 SERVEUR:504	Le canal fait office de serveur TCP. Lorsque la communication est lancée, Docklight accepte une connexion d'un client TCP. Lorsqu'un client est connecté, les autres tentatives de connexion sont rejetées.
PROXY:*Connexion locale* PROXY:10001	Comme le SERVER, mais en mode surveillance, il contrôle le deuxième canal en fonction de la connexion acceptée par le serveur. Si le deuxième canal ferme de force une connexion, le serveur PROXY déconnecte également la connexion acceptée.
UDP:*Hôte distant*:*Connexion* UDP:10.0.0.1:8001 UDP:LOCALHOST:10001	Le canal se comporte comme une contrepartie UDP. Les données d'émission sont envoyées à la destination *RemoteHost*:*Port*, et Docklight écoute les données UDP au numéro de port UDP local *Port.* Si vous utilisez un réglage de canal tel que UDP:LOCALHOST:10001, vous créez effectivement une rétroaction, similaire à celle d'une ligne série. Interface dans laquelle toutes les données sortantes sont reçues immédiatement.
UDP:*Hôte distant* *RemotePort* : *LocalPort* UDP:10.0.0.1:8001:8002	Le canal se comporte comme un pair UDP, mais utilise des numéros de port différents pour les données sortantes et entrantes. Les données sont transmises au *RemotePort* et Docklight écoute au *LocalPort*.
UDP:*LocalPort* UDP:10001	Le canal fait office de serveur UDP. Docklight écoute les données UDP sur le *LocalPort*. Les données d'envoi sont transmises à l'IP source et au numéro de port du dernier paquet UDP. reçues.

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

PIPE:*myNamedPipe* PIPEREAD:*myNamedPipe* PIPEWRITE:*myNamedPipe*	Connexion du client à un tuyau nommé avec lecture /Connexion client en lecture seule Accès en écriture seulement
USBHID:*vendorId*:*productId* USBHID:4D8:F708	Rapport d'entrée/sortie USB-HID Docklight ouvre une connexion au périphérique HID USB spécifié (ou à un périphérique HID Bluetooth) et permet d'envoyer et de recevoir des données de rapport d'entrée et de sortie HID.

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

(ou utiliser le chemin d'accès au périphérique USB) USBHID:\\ ? \hid#vid_04d8&pid_...	L'indicateur de communication Docklight est basé sur des rapports : Chaque rapport d'entrée génère un nouvel horodatage Docklight et imprime les données de rapport HID d'origine, y compris l'ID du rapport d'entrée comme premier octet si > 0.
USBHID:*vendorId* :*productId*:P USBHID:4D8:F708:P	Accès HID USB, basé sur le protocole. Seules les données utiles proprement dites sont affichées, sans l'octet d'ID de rapport d'entrée qui précède et/ou les octets zéro qui suivent. Les horodatages sont générés selon les règles d'horodatage habituelles, pas avant chaque rapport.
USBHID:*vendorId* :*productId*:I USBHID:4D8:F708:I	Accès HID USB, mais avec un ID de rapport de sortie variable faisant partie des données de la séquence d'envoi. Dans ce mode, toutes les définitions de séquence d'envoi nécessitent un octet supplémentaire au début de la séquence, qui détermine l'ID de rapport de sortie à utiliser réellement.
USBHID:*vendorId* :*productId* : *[P],outputID,outputPayloadSize* USBHID:4D8:F708:,1,63 USBHID:4D8:F708:P,2	Syntaxe étendue USB HID : *outputID* : si spécifié, cet identifiant d'état de sortie sera utilisé à la place de la valeur par défaut de zéro. *outputPayloadSize* : si spécifié, la longueur de l'état sera remplacée par la valeur que vous avez spécifiée. Windows signale via HID_CAPS.OutputReportByteLength, ignoré
PIPE:*myNamedPipe* PIPEREAD:*myNamedPipe* PIPEWRITE:*myNamedPipe*	Connexion du client à un tuyau nommé avec lecture /Connexion client en lecture seule Accès en écriture seulement

Mode de surveillance - Combinaisons de canaux et leurs applications
En mode de surveillance, deux canaux de communication sont disponibles et peuvent être configurés individuellement. Ainsi, Docklight Scripting peut être utilisé dans une multitude d'applications et d'environnements de test différents. Vous trouverez ci-dessous une liste de combinaisons de canaux typiques :

Réglages du canal de communication	**Exemple - Réglages**	**Inscription**
Canal 1 : Port COM Ch2 : COM-Raccordement	COM1 COM2	Surveillance de la communication sérielle entre deux appareils avec un câble de surveillance Docklight
Le Ch1 : Taper sur la station d'accueil Le Ch2 est le plus important : Taper sur la station d'accueil	TAP0 TAP1	Surveillance de la communication sérielle entre deux appareils avec un Docklight Tap
Ch1 : Tap Pro ou 485 Ch2 : Taraud Pro ou 485	VTP0 VTP1	Surveillance de la communication sérielle entre deux appareils avec un Docklight Tap Pro ou un Docklight Tap 485

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

Canal 1 : port COM Ch2 : Serveur TCP	COM1 SERVEUR:100 0 1	Émulation d'un périphérique série sur le serveur. Un client peut se connecter au serveur Docklight sur le port 10001 et parler au périphérique série connecté à COM1.
Ch1 : Client TCP Ch2 : Serveur TCP	10.0.0.1:502 SERVER:502	Surveillance d'une connexion TCP client/serveur avec Docklight comme passerelle.
		entre les deux parties.
Le Ch1 : UDP Peer Ch2 : Pair UDP	UDP:10.0.0.1:8 001 UDP:10.0.0.2:8 002	Surveillance et transmission d'une requête UDP Transmission, similaire à l'exemple TCP ci-dessus. Notez que pour chaque canal, vous devez utiliser un un port UDP différent, puisque chaque canal doit écouter son propre numéro de port.

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

Paramètres de port COM (canaux COM, TAP et VTP uniquement)

Vitesse de transmission
Sélectionnez une vitesse de transmission standard dans la liste déroulante ou utilisez une vitesse de transmission non standard en saisissant n'importe quel nombre entier compris entre 110 et 9999999.

REMARQUE : Les vitesses de transmission non standard peuvent ne pas fonctionner sur tous les ports COM, en fonction des capacités de la puce UART matérielle de votre port COM. Vous ne recevrez pas d'avertissement si un débit en bauds non standard ne peut pas être appliqué.

REMARQUE : bien que vous puissiez définir des vitesses de transmission allant jusqu'à 9 Mbauds dans les paramètres de projet de Docklight, cela ne signifie pas que Docklight peut gérer ce débit de données de manière permanente. Le débit de données moyen dépend fortement des performances de votre PC et des paramètres d'affichage de Docklight. Voir aussi Comment augmenter la vitesse de traitement.

REMARQUE : Il existe de nombreux pilotes et applications de port COM qui n'utilisent pas la transmission RS232/422 ou 485 et qui ne nécessitent aucun des paramètres de communication RS232. Dans certains cas, ces pilotes de port COM renvoient même une erreur lorsqu'ils tentent de définir les paramètres RS232, de sorte que Docklight ne peut pas ouvrir le canal COM. Pour ces applications, utilisez le paramètre **Pas** de débit en bauds.

Bits de données et bits d'arrêt
Indiquez ici le nombre de bits de données et de bits d'arrêt. Comme pour la vitesse de transmission, il se peut que certains des réglages disponibles ne soient pas pris en charge par les périphériques de port COM de votre PC.

Coq 485 caractères. Niveau.
La Docklight Tap Pro / Tap 485 prend en charge des niveaux de tension supplémentaires en plus des tensions RS232 standard :

RS485/422 - les niveaux de tension différentiels pour les applications de bus RS485 et RS422. **Inversé** - mode RS232/TTL inversé, dans lequel l'état de marquage (ou 1 logique) est la tension positive et l'état d'espace (0 logique) est la tension négative ou zéro volt.

Parité
Toutes les options courantes de contrôle de parité sont disponibles ici. (Les paramètres 'Mark' et 'Space' ne sont probablement pas utilisés dans la pratique. 'Mark' indique que le bit de parité est toujours à 1, 'Space' que le bit de parité est toujours à 0, indépendamment du caractère transmis).

Signe d'erreur de parité
Il s'agit du caractère qui remplace un caractère non valide dans le flux de données lorsqu'une erreur de parité se produit. Vous devez indiquer un caractère ASCII (imprimant ou non) qui n'apparaît normalement pas dans votre flux de données série. Les caractères peuvent être définis en saisissant le caractère lui-même ou en saisissant le code décimal ASCII (veuillez saisir au moins deux chiffres).

REMARQUE : sélectionnez "(ignorer)" pour le caractère d'erreur de parité si vous devez envoyer/recevoir le bit de parité, mais que vous voulez que Docklight conserve tous les caractères entrants, même si le bit de parité est incorrect. Ceci est utile pour les applications où un 9e bit est utilisé pour l'adressage et non pour la vérification des erreurs.

Utiliser le balayage de la vitesse de transmission - canaux VTP uniquement

Les appareils Docklight Tap Pro / Tap 485 offrent un mode de balayage de la vitesse de transmission / autodétection. Pour activer le balayage de la vitesse de

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

transmission, sélectionnez dans le réglage **Baudrate**

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

option **Aucune** et fermez la boîte de dialogue Paramètres du projet. Démarrez maintenant la communication via le menu **Exécuter > Démarrer la communication (F5)**. Le Docklight Tap Pro / Tap 485 scanne maintenant la communication indépendamment dans les deux directions. Si des données sérielles ont pu être détectées dans les deux sens de données,

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

les paramètres les plus probables sont affichés sous forme de commentaires dans la fenêtre de communication. Ils sont également affichés dans la barre d'état de la communication sous la barre d'outils principale.

REMARQUE : La précision de cette fonction de détection automatique dépend du flux de données réel présent pendant l'analyse. Un flux continu de données très aléatoires donne une grande précision de reconnaissance, tandis que de petites transmissions d'octets individuels ou des motifs répétitifs peuvent donner lieu à des estimations erronées des vitesses de transmission, des bits de données ou des parités.

9.9 Dialogue de gestion des flux : Paramètres du projet - Contrôle du flux

Menu **Outils > Paramètres du projet... | Contrôle de flux**

Permet de spécifier des paramètres de contrôle de flux matériel ou logiciel supplémentaires pour la communication série en mode d'envoi/réception de la Docklight.

Soutien au contrôle des flux

De
Aucun mécanisme de contrôle de flux matériel ou logiciel n'est utilisé. RTS et DTR sont activés lorsque le port COM est ouvert.

Manuel
Dans ce mode, vous pouvez contrôler manuellement les signaux RTS et DTR et afficher l'état actuel des lignes CTS, DSR, DCD et RI. Lorsque le contrôle de flux est réglé sur "Manuel", un élément d'état supplémentaire s'affiche dans la fenêtre principale du Docklight. Vous pouvez commuter les lignes RTS et DTR en double-cliquant sur l'indicateur correspondant.

REMARQUE : Les signaux de contrôle de flux ne sont pas traités comme des données de communication et ne sont pas affichés dans la fenêtre de communication ni consignés dans un fichier.

handshaking matériel, handshaking logiciel
Prise en charge du contrôle de flux matériel RTS/CTS et du contrôle de flux logiciel XON/XOFF. Il s'agit de paramètres experts qui sont rarement nécessaires pour les applications de communication actuelles.

Contrôle de l'émetteur-récepteur RS485
Certains convertisseurs RS232 vers RS485 nécessitent un contrôle RTS manuel, c'est-à-dire que le périphérique RS232 (PC) indique au convertisseur quand il doit activer son pilote RS485 pour la transmission. Si vous choisissez "RS485 Transceiver Control", le port COM met RTS à l'état haut avant que le premier caractère d'une séquence d'envoi ne soit transmis et le remet à l'état bas après que le dernier caractère ait été transmis.

REMARQUE : de nombreux convertisseurs USB vers série ou pilotes de port COM virtuel n'implémentent pas correctement le mode Windows RTS_CONTROL_TOGGLE. Si vous avez des problèmes avec le contrôle de l'émetteur-récepteur RS485, essayez d'utiliser un PC avec port COM intégré ou une carte PCI-COM standard.

9.10 Dialogue de communication : Paramètres du projet - Filtres de communication

Menu **Outils > Paramètres du projet... | Filtre de**

communication Filtre de contenu

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

Utilisez cette option si vous n'avez pas besoin de voir les données de communication originales sur la ligne série et si vous avez uniquement besoin des commentaires supplémentaires insérés par une séquence de réception. Cette option est utile pour les applications à haut débit où la plupart des données ne sont pas pertinentes pour les tests et où vous ne devez prêter attention qu'à des événements très spécifiques. Ces

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

Les événements (et la sortie d'affichage correspondante) peuvent être définis avec Receive Sequences.

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

Alias du canal
Cela permet de réétiqueter les deux directions de données Docklight en fonction de l'application. Par exemple [Docklight] / [Appareil] au lieu de [TX] / [RX]. Ou [Maître] / [Esclave] au lieu de [TAP0] / [TAP1].

9.11 Le dialogue : Options

Menu **Outils > Options...**

Annonce

Sortie de texte formaté (Rich Text Format)
sert à régler l'apparence de la fenêtre de communication du Docklight. Les deux flux de données sériels différents, "canal 1" et "canal 2", peuvent être représentés avec des couleurs et des polices différentes. Par défaut, des couleurs différentes sont utilisées pour les deux canaux, mais il est également possible d'utiliser des styles de police différents (par exemple, l'italique pour le "canal 2"). Vous pouvez également choisir ici la taille totale de la police.

REMARQUE : si vous modifiez la taille de la police, le contenu de la fenêtre de communication sera effacé. Pour toutes les autres modifications, Docklight essaie de conserver le contenu de l'affichage.

Plain Text Output (affichage plus rapide, mais pas de couleurs ni de polices) La sortie de texte formaté ressemble à celle d'un programme de traitement de texte et consomme une quantité considérable de temps de CPU. Elle nécessite également une allocation et une libération fréquentes de la mémoire, ce qui peut nuire aux performances de votre PC. Par conséquent, si vous avez besoin d'un système de traitement de texte à grande vitesse
Si vous souhaitez surveiller la liaison de communication tout en gardant un œil sur les données sérielles transmises, vous devez utiliser le format "Plain Text Output".

Caractères de contrôle (ASCII 0 - 31)
Pour les données de communication qui contiennent à la fois du texte ASCII imprimable et des caractères de contrôle non imprimables, il est parfois utile de voir les noms des caractères de contrôle qui apparaissent dans la fenêtre d'affichage du mode ASCII. Docklight propose un paramètre d'affichage optionnel qui permet cela. Vous pouvez également supprimer les caractères de contrôle (à l'exception de CR et LF) si cela rend l'affichage confus.

Modes d'affichage

Modes de la fenêtre de communication
Docklight affiche par défaut quatre représentations des flux de données série : ASCII, HEX, Décimal et Binaire. Vous pouvez désactiver certains de ces modes afin d'augmenter les performances globales de Docklight. Par exemple, la représentation binaire des données n'est que rarement nécessaire. Si vous désactivez le mode binaire pour la fenêtre de communication, la vitesse de traitement augmente considérablement. Même si le mode binaire pour la fenêtre de communication est désactivé, la journalisation reste possible dans tous les formats.

Voir aussi l'option **"Sortie de texte"** ci-dessus.

Cachet date/heure

Ajouter un horodateur
Docklight appose un horodatage sur toutes les données envoyées ou reçues. Vous pouvez insérer cette date/heure dans la fenêtre de communication et le fichier journal lorsque le sens du flux de données entre le canal 1 et le canal 2 change.

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

Pour les applications où le sens du flux de données ne change pas très souvent, vous souhaitez peut-être avoir des horodateurs supplémentaires à intervalles réguliers. Pour ce faire, activez l'option **Horloge - Horodateur supplémentaire. /Tampon horaire...** et choisissez un intervalle de temps.

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

Sur une ligne semi-duplex (par exemple RS485 à 2 fils), les changements de direction des données sont difficiles à détecter. Cependant, dans la plupart des applications, il y aura une pause sur le bus de communication avant qu'un nouvel appareil ne commence à émettre. Utilisez l'option **Détection des pauses...** pour introduire des horodateurs supplémentaires et rendre les pauses visibles dans votre protocole de communication.

Format de la date et de l'heure
Docklight propose des horodatages avec une résolution allant jusqu'à 1/1000 de seconde (1 milliseconde). Pour la compatibilité avec les versions précédentes de Docklight (V1.8 et antérieures), 1/100e de seconde est également disponible.

REMARQUE : la précision du marquage temporel qui en résulte peut varier considérablement, par exemple, seulement 10-20 millisecondes. La précision réelle dépend de votre équipement de communication série, de la configuration de votre PC, des paramètres d'affichage du Docklight (voir ci-dessus) et des options expertes du Docklight. Pour plus d'informations, consultez la section Comment obtenir la meilleure précision de chronométrage.

Raccourcis pour les caractères de contrôle

Vous pouvez définir ici vos propres raccourcis clavier pour les caractères de contrôle ASCII (code ASCII < 32) ou pour n'importe quel code de caractère > 126. Les raccourcis clavier peuvent être utilisés dans les boîtes de dialogue et fonctions Docklight suivantes

- Dialog : Modifier l'ordre des chaînes
- Dialogue de la fonction : Modifier la séquence de réception Dialogue : Paramètres séquence d'envoi
- Console de clavier

Pour chaque caractère du code décimal 0 à 31 et de 127 à 255, vous pouvez définir une combinaison de touches pour insérer ce caractère dans une séquence (**raccourci**). Vous pouvez également définir une lettre qui sera utilisée pour afficher ce caractère de contrôle lors de l'édition d'une séquence en mode ASCII (**éditeur**).

Double-cliquez pour modifier la valeur d'un champ de raccourci

ou d'édition. Les raccourcis prédéfinis sont
Ctrl+entrée pour retour chariot / <CR> / code décimal 13
Ctrl+Shift+Enter pour le saut de ligne / <LF> / code décimal 10

9.12 de dialogue : Options des experts

Menu **Outils > Options expert...**

Les options expertes sont des paramètres supplémentaires pour des applications spéciales avec des exigences supplémentaires (par exemple une grande précision de la saisie des temps).

Puissance

Mode pilote de communication

Utilisez le mode **Processus externe / Processus à haute priorité** pour contourner un problème courant pour toute application Windows en mode utilisateur : Des

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

retards indéfinis et des imprécisions temporelles peuvent être causés par la planification des tâches/processus de Windows, en particulier si vous exécutez d'autres applications en plus de Docklight.

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

Le mode de processus **Externe / Haute priorité** est recommandé pour une surveillance de haute précision avec une faible latence en utilisant le Docklight Tap.

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

REMARQUE : pour une précision encore plus grande et garantie du chronométrage, utilisez l'accessoire Docklight Tap Pro / Tap 485. Leur précision ne dépend pas des latences de Windows et des pilotes, et le mode de traitement "haute priorité" n'est pas nécessaire pour les applications Tap Pro et Tap 485.

En mode **Externe / Processus hautement prioritaire**, la collecte de données dans Docklight devient un processus Windows distinct avec la classe de priorité "Temps réel". Il s'exécute avec une priorité plus élevée que toute autre application utilisateur ou logiciel d'application supplémentaire tel qu'Internet Security / Antivirus. Pour obtenir des résultats optimaux, Docklight doit **être exécuté en tant qu'administrateur**. Dans le cas contraire, le processus de collecte des données s'exécute avec la priorité maximale autorisée par le système d'exploitation, mais pas avec la classe "temps réel".

Externe / Haute priorité Le mode Processus doit être utilisé avec précaution, en particulier si vous souhaitez surveiller une connexion de données à haut débit avec de grandes quantités de données. Le PC pourrait ne plus répondre aux entrées de l'utilisateur. Pour remédier à une telle situation, il suffit de "débrancher" : commencez par déconnecter les connexions de données ou le câble de surveillance afin de réduire la charge du processeur et de rétablir la réactivité. Ensuite, dans Docklight, sélectionnez ■ **Arrêter la communication**.

REMARQUE : Vous trouverez quelques informations de base sur la précision du timing dans la section Comment obtenir la meilleure précision de timing ?

Mode de surveillance Docklight

Lors de la surveillance de la communication sérielle entre deux appareils, toutes les données reçues d'un port COM sont retransmises par défaut sur le canal d'émission du port COM opposé ("Data Forwarding"). Ceci est destiné à des applications spéciales qui nécessitent la retransmission du trafic de données série par Docklight via un câblage RS232 standard.

Utilisez l'option experte **Pas de transfert de données** pour les applications à deux ports COM série où vous devez éviter d'envoyer des données de transmission. Cette option peut être utilisée pour améliorer les performances lors de l'utilisation d'un câble de surveillance Docklight ou pour contourner les problèmes causés par l'instabilité des pilotes de périphériques série.

Pour les applications Docklight Tap (par ex. avec le canal de communication TAP0 / TAP1), le réglage 'Transfert de données' n'a aucun effet. L'accès à Docklight Tap est toujours en lecture seule et aucune donnée n'est transférée.

Appareils

Périphériques COM Windows

Utilisez **Désactiver la détection d'erreurs E/S** si vous recevez des messages d'erreur répétés comme celui-ci :

```
DOCKLIGHT signale : Erreur générale d'E/S sur COM1
```

REMARQUE : Docklight utilise la communication série Windows en mode "overlapped I/O" pour obtenir la meilleure efficacité et la meilleure précision temporelle, et évalue en permanence les erreurs provenant des appels Win32 API correspondants. Dans de rares cas, comme par exemple pour les périphériques COM avec des pilotes de périphériques COM défectueux ou obsolètes, de telles erreurs peuvent également se produire en mode lecture/écriture normal. Dans ce cas, vous pouvez utiliser cette option pour revenir au comportement de Docklight V2.2 et des versions précédentes : De telles erreurs sont tout simplement ignorées.

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

Taraudage Pro / Taraudage 485

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

Les fonctions de mise à jour du micrologiciel pour notre accessoire matériel Docklight Tap Pro / Docklight Tap 485 ne sont nécessaires que dans de rares cas. Par exemple, si vous utilisez un appareil plus ancien (< année 2017) qui ne supporte pas encore la fonction de balayage de la vitesse de transmission.

9.13 Console de clavier

L'outil Console clavier permet d'envoyer des saisies clavier directement à l'interface série. Il peut être activé via le menu **Tools > Keyboard Console On.** La console clavier n'est disponible que pour le mode de communication Envoyer/Réception.

Après avoir activé la console du clavier, cliquez dans la fenêtre de communication et saisissez quelques caractères.

Docklight transmet les caractères directement via le port série sélectionné. La fenêtre de communication affiche les caractères de la même manière qu'une séquence d'envoi.

REMARQUE : La console du clavier prend en charge le collage et le transfert d'une séquence de caractères à partir du presse-papiers à l'aide de la combinaison de touches **Ctrl + V**. Cela est comparable au collage de données depuis le presse-papiers dans la boîte de dialogue "Modifier la séquence d'envoi". Les contenus du presse-papiers qui dépassent la taille maximale de la séquence de 1024 caractères sont coupés.

REMARQUE : La console clavier n'est pas un terminal à part entière et ne prend pas en charge les standards de terminaux spéciaux tels que VT 100. La touche Enter est transmise sous forme de **<CR>** (ASCII 13) plus **<LF>** (ASCII 10). La touche ESC transmet <ESC> **(ASCII** 27).
Utilisez des raccourcis de caractères de contrôle pour envoyer d'autres caractères de contrôle ASCII.

REMARQUE : La console du clavier ne prend pas en charge l'insertion de caractères ASCII en appuyant/maintenir la touche ALT et l'utilisation du pavé numérique. Veuillez utiliser la boîte de dialogue Modifier les séquences d'envoi en représentation HEX ou décimale pour créer tout code de caractère ASCII > 127.

9.14 Spécification de la somme de contrôle

Les indications de sommes de contrôle sont utilisées dans les boîtes de dialogue Edit Send Sequence et Edit Receive Sequence ainsi que dans la méthode de script Docklight CalcChecksum. Pour une vue d'ensemble générale, voir Calculer et valider les sommes de contrôle.

Spécifications des sommes de contrôle prises en charge / *checksumSpec* Argument

checksumSpec	Algorithme de somme de contrôle appliqué
MOD256	Somme de contrôle simple de 8 bits : somme sur tous les octets, modulo 256.
XOR	Somme de contrôle de 8 bits : XOR sur tous les octets.
CRC-7	CRC de 7 bits de large. Utilisé par exemple dans les applications de cartes MMC/SD. Un texte checksumSpec alternatif pour le même type de somme de contrôle serait : CRC:7,09,00,00,Non,Non Voir la syntaxe décrite à la dernière ligne "CRC:*width*, *polynomial...*".
CRC-8	CRC de 8 bits de large, par exemple pour ATM Head Error Correction. Equivalent à : CRC:8,07,00,00,Nein,Nein

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

CRC-DOW	CRC de 8 bits de large, connu sous le nom de DOW CRC ou CCITT-8 CRC. Peut être trouvé dans les applications Dallas iButton(TM). Équivalent à : CRC:8,31,00,00,Yes,Yes

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

MOD65536	Somme de contrôle simple de 16 bits : somme sur tous les octets, modulo 65536.

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

CRC-CCITT	CRC d'une largeur de 16 bits, tel que défini par le CCITT. Équivalent à : CRC:16,1021,FFFF,0000,No,No
CRC-XMODEM	CRC d'une largeur de 16 bits, similaire au CRC-CCITT, mais le La valeur initiale est zéro. Équivalent à : CRC:16,1021,0000,0000,No,No
CRC-16	CRC avec une largeur de 16 bits, comme utilisé dans IBM Bisynch, ARC. Équivalent à : CRC:16,8005,0000,0000,Yes,Yes
CRC-MODBUS	CRC d'une largeur de 16 bits, comme utilisé dans Modbus. Similaire au CRC-16, mais avec une valeur d'init différente. Même chose que CRC:16,8005,FFFF,0000,Yes,Yes
CRC-32	CRC 32 bits, comme utilisé dans PKZip, AUTODIN II, Ethernet, FDDI. Équivalent à : CRC:32,04C11DB7,FFFFFFFF,FFFFFFFF,Oui,Oui
-MOD256 *ou* LRC	Similaire à MOD256, mais renvoie le résultat négatif sur 8 bits, de sorte que la somme de tous les octets, y compris la somme de contrôle, est nulle. Cela correspond à ce que l'on appelle le LRC (Longitudinal Redundancy Check), utilisé par exemple dans les applications POS.
LRC-ASCII	Comme -MOD256 / LRC, mais il s'attend à ce que les données sources soient des nombres HEX sous forme de texte ASCII lisible. Voir l'exemple MODBUS- ASCII ci-dessous.
CRC:*width*, *polynomial*, *init*, *finalXOR*, *reflectedInput*, *reflectedOutput*	Calculateur CRC générique dans lequel tous les paramètres CRC peuvent être réglés individuellement : *width* : la largeur du CRC de 1..32. *polynomial* : valeur HEX. Le polynôme CRC tronqué. *init* : valeur HEX. Le reste initial avec lequel le calcul commence. *finalXor* : Valeur HEX. Applique une opération XOR sur le reste résultant avant de le renvoyer à l'utilisateur. *reflectedInput* : Oui = Reflète les octets de données (MSB devient LSB) avant qu'ils ne soient injectés dans l'algorithme. *reflectedOutput* : Oui = Reflète le résultat une fois l'algorithme terminé. Cela se fait avant la combinaison XOR finale. fonctionnement.

Remarques

Chacun des algorithmes CRC prédéfinis (CRC-8, CRC-CCITT, ...) peut être remplacé par une chaîne de spécification pour le calcul générique du CRC (CRC:8,07,00...) comme décrit ci-dessus. Nous avons soigneusement testé nos implémentations et les avons comparées à la littérature courante et aux ressources mentionnées dans le glossaire CRC.

Malheureusement, il existe une multitude de variantes et d'algorithmes CRC, et choisir (pour ne pas dire comprendre) la bonne variante CRC peut être une tâche assez difficile. Une bonne façon de s'assurer que votre calcul du CRC est valable est de l'effectuer avec une chaîne de test ASCII de "123456789". Il s'agit de la chaîne de test la plus couramment utilisée et de nombreuses spécifications s'y réfèrent pour vous fournir la somme de contrôle correcte que le CRC devrait donner lorsqu'il est appliqué à cette chaîne.

Modifier les sommes de contrôle dans la séquence d'envoi / Modifier la séquence de réception

Dans l'onglet **Somme de contrôle**, sélectionnez l'une des chaînes de définition

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

prédéfinies dans la liste déroulante ou saisissez votre propre chaîne de définition.

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

définition dans le format suivant :

[(*startPos*, *len*)] ***checksumSpec*** [A *ou* L] [@ *targetPos*] [# *commentaire utilisateur optionnel*], où tout ce qui se trouve à l'intérieur de [ ] est une partie optionnelle.

Partie	Description

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

checksumSpec	Obligatoire. Chaîne de caractères indiquant l'algorithme de somme de contrôle et ses paramètres conformément au tableau "checksumSpec Format" ci-dessus.
(*startPos*, *len*) par exemple (1, 4)	Facultatif . Début et longueur de la zone de caractères utilisée pour calculer la somme de contrôle. Par défaut, tout ce qui précède la somme de contrôle résultat est utilisé.
A	Facultatif. Si elle est utilisée, la valeur de checksum résultante est convertie en un nombre HEX sous forme de texte ASCII lisible. Voir l'exemple MODBUS-ASCII ci-dessous.
L	Facultatif . Little Endian - la valeur de la somme de contrôle qui en résulte est octet de poids faible (LSB) en premier. Le standard est Big Endian / MSB first.
@ *ObjectifPos* par exemple @ -4	Facultatif . Indique la première position de caractère pour l'enregistrement de la valeur de somme de contrôle résultante. Par défaut, Docklight écrit le résultat de la somme de contrôle aux dernières positions de données de la séquence, à moins que vous n'ayez indiqué "A" pour résultat ASCII. Dans ce cas, le résultat est enregistré un caractère avant la fin, ce qui laisse de la place pour un caractère de "fin de ligne", typiquement un CR comme pour l'ASCII Modbus.
# *Commentaire*	Vous pouvez ajouter un commentaire à cette indication de somme de contrôle entrer

Remarques

startPos, *len* et *targetPos* prennent également en charge les valeurs négatives pour indiquer les positions par rapport à la fin de la séquence et non par rapport au début de la séquence.
Exemples :
startPos est -4 : début du calcul au 4ème caractère de la fin.
len est -1 : utiliser tout jusqu'à la fin de la séquence.
targetPos est -1 : le premier (et seul) octet du résultat est stocké à la dernière position de caractère de la séquence.
targetPos est -2 : le résultat est enregistré à partir du 2e caractère de la fin.
targetPos est -3 : le résultat est enregistré à partir du 3ème caractère en partant de la fin.

Exemples

Sommes de contrôle - Spécification	**Exemple de séquence d'émission**	**Dates d'envoi réelles**	**Remarques**
# (désactivé, pas de somme de contrôle)	01 | 02 | 03 | 04 | 05 | 00	01 02 03 04 05 00	après un **#**, vous pouvez saisir n'importe quel commentaire pour Votre somme de contrôle décrire
MOD256 # somme simple d'un octet sur tous les caractères sauf le dernier	01 | 02 | 03 | 04 | 05 | **00**	01 02 03 04 05 **0F**	Un **00** supplémentaire a été ajouté comme caractère de remplacement pour la somme de contrôle, mais vous pouvez utiliser n'importe quelle valeur de **00 à FF**.

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

CRC-MODBUS L # Somme de contrôle Modbus RTU. Octet inférieur en premier ('Little Endian')	01 | 06 | 01 | 02 | 00 | 07 | **FF | FF**	01 06 01 02 00 07 **68 34**	**CRC-MODBUS** est un 16 Somme de contrôle des bits, placée par défaut aux deux dernières positions de caractères dans
			les données de séquence.
(2, -5) LRC-ASCII A @ -4 # MODBUS ASCII La somme de contrôle est LRC sur des données HEX lisibles, à l'exception de Start	: | 1 | 1 | 0 | 3 | 0 | 0 | 6 | b | 0 | 0 | 3 | **X | X |** r | n	**:1103006b00037E<** CR><LF>	**LRC-ASCII** traite de la Données de séquence sous forme de chaîne de caractères HEX lisible, chaque octet de données étant représenté par deux caractères. L'utilisation de l'option **A** génère un message lisible 2

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

':' et fin 'CR/LF			Lettres- Texte de la somme de contrôle au lieu d'un à un chiffre résultat. Le **@ -4** place le Résultat à la 4ème Position des caractères de droite (où la CR/LF intacts à la fin).
CRC:8,07,00,00,No,Oui # CRC avec spécifications définies par l'utilisateur, non standardisées	01 | 02 | 03 | 04 | 05 | **00**	01 02 03 04 05 **3D**	Les saveurs CRC rares ou personnalisées peuvent être calculées par Docklight, mais vous devez connaître les paramètres de calcul CRC requis. Pour plus de détails, consultez les ressources mentionnées dans le glossaire CRC.

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

Référence (scripting)

10 Référence (scripting)

10.1 Bases de VBScript

Si vous connaissez déjà *Visual Basic*® ou *Visual Basic*® for Applications (VBA), VBScript vous sera très familier. Jetez un coup d'œil aux définitions et exemples ci-dessous. Pour commencer, vous pouvez essayer certains des exemples suivants en copiant et collant le code dans l'éditeur de script et en exécutant le script. Docklight Scripting contient également un certain nombre d'exemples de scripts que vous pouvez essayer.

Ce chapitre présente quelques fonctions et caractéristiques de base de VBScript. Vous trouverez une référence complète dans la documentation originale de *Microsoft*® aux endroits suivants :

- Visual Basic Scripting Edition (ou allez sur www.microsoft.com et cherchez "VBScript")
- Manuel de l'utilisateur VBScript
- Référence de langage VBScript.

ASTUCE : Utilisez la fonction ScriptEngine pour savoir quelle version de VBScript est installée sur votre ordinateur.

REMARQUE : Docklight Scripting exécute le code VBScript en "mode sûr" (safe subset) et interdit les actions potentiellement nuisibles. Par exemple, la création d'un "FileSystemObject" (E/S de fichier) est l'une des actions interdites en mode sécurisé de VBScript.
sous-ensemble. Le script Docklight s'interrompt avec un message d'erreur. Veuillez contacter notre support par e-mail si vous avez des besoins spécifiques et si vous devez utiliser des instructions VBScript "non sécurisées". À la demande générale, la saisie de fichiers est désormais possible avec FileInput de Docklight / Objets FileOutput.

Caractéristiques spécifiques au Docklight

- DockliCommandes de script ght - L'objet DL
- Procédures d'événement Docklight OnSend / OnReceive
- Docklight FileInput / FileOutput Objet pour la lecture et l'écriture de fichiers

Fonctions de base VBScript par catégorie

- Structures de contrôle (structures de décision, structures en boucle)
- Variables tableaux, les constantes et les types de données
- Betriver
- Fonctions date/heure
- Autres

Fonctions de base de VBScript par ordre alphabétique

- Datum Fonction
- Tag Fonction
- Tun à ...boucle
- Do...Rectificatione While
- FPour les ...Continuer
- Sfonction heure
- Wsi... alors
- Si... alors... sinon

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

- Fonction InputBox

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

- LFonction Bound
- Mfonction minute
- Mon mois Fonction
- Jfonction actuelle
- ScriptEngifonction
- Deuxièmee fonction
- Fsélectionner tout
- Zeit Fonction
- TFonction imer
- UFonction Bound
- Pendant... Wend
- Année Fonction

10.1.1 Avis de copyright

Les sections suivantes du chapitre "Principes de base de VBScript" sont basées sur le fichier d'aide Script56.CHM *de la* documentation *Microsoft® Windows* Script V5.6. La mention de copyright suivante s'applique à ce fichier d'aide : "© 2001 *Microsoft®* Corporation. Tous droits réservés".

L'utilisation de matériel protégé par les droits d'auteur *de Microsoft®* est régie par la "règle des dix pour cent" de *Microsoft®* (voir http://www.microsoft.com/permission).

10.1.2 Contrôler les structures

Les structures de contrôle VBScript vous permettent de contrôler le déroulement de l'exécution du script. Pour en savoir plus sur certaines structures de contrôle, consultez les rubriques suivantes :

- Structures de décision Une introduction aux structures de décision utilisées pour le branchement conditionnel.
- Structures de boucles Une introduction aux structures de boucles utilisées pour répéter des processus.

être.

10.1.2.1 Structures décisionnelles

- **Si... alors**

  Utilisez une structure If...Then pour exécuter une ou plusieurs instructions sous réserve. Vous pouvez utiliser soit une syntaxe à une ligne, soit une syntaxe en bloc à plusieurs lignes :

```
Instruction si-condition-alors

Si condition Alors
Témoignages
Fin quand
```

  La condition est généralement une comparaison. Si la condition est True, VBScript exécute toutes les instructions qui suivent le mot-clé Then. Vous pouvez utiliser soit une syntaxe à une ligne, soit une syntaxe à plusieurs lignes pour n'exécuter qu'une seule instruction de manière conditionnelle (ces deux exemples sont équivalents) :

```
If anyDate < Now Then anyDate = Now

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


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

Notez que la forme à une ligne de If...Then n'utilise pas d'instruction End If. Si vous souhaitez exécuter plus d'une ligne de code lorsque la condition est vraie, vous devez utiliser la syntaxe de bloc multiligne If...Then...End If.

-If...Then...Else

Utilisez un bloc If...Then...Else pour définir plusieurs blocs d'instructions, dont un seul sera exécuté à la fois :

```
If condition1 Then
[statementblock-1]
[ElseIf condition2 Then
[statementblock-2]] ...
[Exclusion [bloc
d'instructions-n]] Fin
si
```

-Choisir une valise

VBScript propose la structure Select Case comme alternative à If...Then...Else pour l'exécution sélective d'un bloc d'instructions parmi plusieurs blocs d'instructions. Une instruction Select Case offre des possibilités similaires à l'instruction If...Then...Else, mais rend le code plus lisible lorsque plusieurs possibilités sont proposées.

```
' Exemple
Select Case
   Jour de la
      semaine(maintenant) Case
      2 DL.AddComment "Lundi
   Cas 3
      DL.AddComment "Mardi"
      Cas 4
      DL.AddComment "Mercredi"
   Cas 5
      DL.AddComment "Jeudi"
   Cas 6
      DL.AddComment "Vendredi"
   Case Else
      DL.AddComment "Week-end
!" End Select
```

10.1.2.2 Structures en boucle

- **Faire jusqu'à ...Boucle**

```
Exemple
Do Until DefResp = vbNo
   MyNum = Int (6 * Rnd + 1)    ' Crée un nombre entier
  aléatoire compris entre 1 et 6.
   DefResp = MsgBox (MonNombre & "Voulez-vous un autre nombre
  ?", vbYesNo)
Boucle
```

- **Do...Boucle While**

```
Exemple
```


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

```
  Je
MonNombre = Int (6 * Rnd + 1) ' Générer un nombre entier aléatoire
    entre 1 et 6.
      DefResp = MsgBox (MonNombre & "Voulez-vous un autre nombre
    ?", vbYesNo)
  Boucle While DefResp = vbYes
```

- **Pendant... Wend**

```
'Exemple
Dim Counter
Compteur = 0  ' Initialiser la variable.
While Compteur < 20  ' Tester la valeur du compteur.
   Compteur = Compteur + 1    '
   Incrémenter le compteur.
   DL.AddComment Compteur
Retourner   ' Terminer la boucle While si le compteur > 19
```

- **Pour...Suivant**

```
Exemple
Pour I = 1 à 5
      Pour J = 1 à 4
            Pour K = 1 à 3
               DL.AddComment I & " " & J & " " & K
            Next
      Continuer
Continuer
```

10.1.3 Variables, tableaux, constantes et types de données

Lorsque vous effectuez des calculs avec VBScript, vous devez souvent enregistrer temporairement des valeurs. Par exemple, vous pouvez calculer plusieurs valeurs, les comparer et effectuer différentes opérations sur celles-ci en fonction du résultat de la comparaison.

- **Variables**

Les noms de variables suivent les règles standard pour nommer les choses en VBScript. Un nom de variable :
- Doit commencer par un caractère alphabétique.
- Ne doit pas contenir de point intégré.
- Ne doit pas dépasser 255 caractères.

- Doit être unique dans le domaine où il est déclaré.

```
Exemples
ApplesSold = 10 ' La valeur 10 est passée à la variable.
ApplesSold = ApplesSold + 1 ' La variable est incrémentée.
```

Tableaux
Les tableaux vous permettent de faire référence à une série de variables portant le même nom et d'utiliser un nombre (un index) pour les distinguer les unes des autres. Cela vous aide dans de nombreuses situations à créer un code plus petit et plus simple, car vous pouvez mettre en place des boucles qui traitent efficacement n'importe quel nombre de cas en utilisant le numéro d'index.

```
'
Exemple
```


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
```

Fonction LBound
Renvoie le plus petit subscript disponible pour la dimension spécifiée d'un tableau.

Syntaxe

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

Partie	Description
arrayname	Nom de la variable du tableau ; suit les conventions standard des noms de variables.
Dimension	Facultatif . Nombre entier indiquant quelle limite inférieure de la dimension est renvoyée. Utilisez 1 pour la première dimension, 2 pour la deuxième et ainsi de suite. Si la *dimension est* omise, 1 est accepté.

Fonction UBound
Renvoie le plus grand subscript disponible pour la dimension spécifiée d'un tableau.

Syntaxe

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

Partie	Description
arrayname	Nom de la variable du tableau ; suit les conventions standard des noms de variables.
Dimension	Facultatif . Nombre entier indiquant quelle limite inférieure de la dimension est renvoyée. Utilisez 1 pour la première dimension, 2 pour la deuxième et ainsi de suite. Si la *dimension est* omise, 1 est accepté.

```
' Exemple Dim
A(100,3,4)
UBound(A,1) ' donne 100
UBound(A,2) ' donne 3
UBound(A,3) ' donne 4
```

- **Constantes**

Une instruction Const peut représenter une grandeur mathématique ou liée à la date/heure :

```
' Exemple
Const conPi = 3.14159265358979
```

- **Types de données**

En VBScript, il n'existe qu'un seul type de données, la variante. Une variante est un type de données particulier qui peut contenir différentes informations selon son utilisation.
Comme Variant est le seul type de données en VBScript, c'est aussi le type de données qui est renvoyé par toutes les fonctions en VBScript.

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

Sous-types de variantes
Outre les classifications simples en tant que nombre ou chaîne de caractères, une variante peut faire d'autres distinctions sur le type spécifique d'informations numériques. Par exemple, une information numérique peut représenter une date ou une heure. Lorsqu'elle est utilisée avec d'autres informations de date ou d'heure, le résultat est toujours exprimé sous la forme d'une date ou d'une heure. Vous pouvez également avoir une grande variété d'informations numériques, allant de valeurs booléennes à de grands nombres en virgule flottante. Ces différentes catégories d'informations qui peuvent être contenues dans une variante sont appelées sous-types. Dans la plupart des cas, il suffit de saisir le type de données souhaité dans une variante et celle-ci se comportera de la manière la plus appropriée pour les données qu'elle contient.

Le tableau suivant montre les sous-types de données qu'une variante peut contenir.

Sous-type	Description
Vide	La variante n'est pas initialisée. La valeur est 0 pour les variables numériques ou une chaîne de longueur nulle ("") pour les variables de type chaîne.
Zéro	La variante ne contient volontairement pas de données valables.
Booléen	Contient soit Vrai soit Faux.
Octet	Contient un nombre entier compris entre 0 et 255.
Entier	Contient un nombre entier compris entre -32,768 et 32,767.
Monnaie	-922.337.203.685.477.5808 à 922.337.203.685.477.5807.
Lang	Contient un nombre entier compris entre -2 147 483 648 et 2 147 483 647.
Individuellement	Contient un nombre à virgule flottante de précision simple dans la plage de -3,402823E38 à -1,401298E-45 pour les valeurs négatives ; 1,401298E-45 à 3,402823E38 pour les valeurs positives.
Double	Contient un nombre à virgule flottante à double précision compris entre -1,79769313486232E308 et -4,94065645841247E-. 324 pour les valeurs négatives ; 4,94065645841247E-324 à 1,79769313486232E308 pour les valeurs positives.
Date (heure)	Contient un nombre représentant une date comprise entre le 1er janvier 100 et le 31 décembre 9999.
Chaîne de caractères	Contient une chaîne de caractères de longueur variable, qui peut compter jusqu'à environ 2 milliards de caractères.
Objet	Contient un objet.
Erreur	Contient un numéro d'erreur.

10.1.4 Opérateur

- Arithmétique

Description	Symbole
Potentialisation	^
Négation unaire	-

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

Multiplication	*
Département	/

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

Division en nombre entier	\

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

Arithmétique du module	Mod
Complémentaire	+
Soustraction	-
Enchaînement de chaînes	&

- Comparaison

Description	Symbole
Égalité des chances	=
Inégalité	<>
Moins de	<
Plus grand que	>
Inférieur ou égal à	<=
Supérieur ou égal à	>=
Équivalence des objets	Est

- Logique

Description	Symbole
Négation logique	Pas
Conjonction logique	Et
Disjonction logique	Ou
Logiquement exclusif	Xou
Équivalence logique	Eqv
Implication logique	Kobold

10.1.5 Fonctions date/heure

- **Date Fonction**

```
Exemple de date Fonction
DL.ClearCommWindows
DL.AddComment Date     ' indique la date système actuelle
```

- **Temps Fonction**

```
Exemple de fonction temporelle
DL.ClearCommWindows
DL.AddComment Heure   ' indique l'heure actuelle du système.
```

- **Fonction minuterie**

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

```
Exemple de fonction de minuterie
La fonction Timer renvoie le nombre de secondes écoulées.
depuis 12:00 AM (minuit).
Heure de début = minuterie
For i = 1 To 1000
Suivant
DL.AddComment "Durée [millisecondes] = " & (Timer -
  StartTime) * 1000
```

- **Maintenant Fonction**

```
'Exemple Now Function
Dim MyVar
MyVar = Now ' MyVar contient la date et l'heure actuelles.
```

- **Jour Fonction**

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

- **Mois Fonction**

```
'Exemple de fonction de mois
DL.AddComment Month(Now)
```

- **Année Fonction**

```
'Exemple de fonction
d'année Dim MyDate
MaDate = #Décembre 7, 1968#          '
Attribution d'une date. DL.AddComment
Année(MaDate)
```

- **Fonction heure**

```
'Exemple de fonction
d'heure DL.AddComment
Hour(Now)
```

- **Fonction minute**

```
Exemple de fonction minute
DL.AddComment Minute(Now)
```

- **Deuxième fonction**

```
Exemple Deuxième fonction
DL.AddComment Second(Now)
```


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

10.1.6 Autres

-Fonction InputBox

Affiche une invite dans une boîte de dialogue, attend que l'utilisateur saisisse du texte ou clique sur un bouton, et renvoie le contenu de la zone de texte.

ASTUCE : utilisez la méthode spécifique à Docklight DL.InputBox2 pour une boîte de dialogue qui apparaît toujours sur le même écran que la fenêtre principale de Docklight Scripting. Ou consultez la fonction DL.GetKeyState pour savoir comment attendre et réagir aux saisies au clavier ou à la souris.

Syntaxe

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

Partie	Description
Invitation à	Obligatoire. Expression de type chaîne de caractères qui s'affiche sous forme de message dans la boîte de dialogue. La longueur maximale de l'invite de commande est d'environ 1024 caractères, en fonction de la largeur des caractères utilisés. Si l'invite comporte plus d'une ligne, vous pouvez séparer les lignes par un caractère de retour chariot (**Chr(**13**)**), un caractère de saut de ligne (**Chr(**10**)) ou une** combinaison de caractères de retour chariot et de saut de ligne (**Chr(**13**) & Chr(**10**)**) entre chaque ligne.
Titre	Facultatif . Expression de type chaîne de caractères qui s'affiche dans la barre de titre de la boîte de dialogue. Si vous omettez title, le nom de l'application s'affiche dans la barre de titre.
Standard	Facultatif . Expression de type chaîne qui s'affiche dans la zone de texte comme réponse par défaut si aucune autre entrée n'est effectuée. Si vous omettez default, le champ de texte s'affiche vide.
xpos	Facultatif . Expression numérique indiquant la distance horizontale, en twips, entre le bord gauche de la boîte de dialogue et le bord gauche de l'écran. l'écran. Si xpos est omis, la boîte de dialogue centré horizontalement.
ypos	Facultatif . Expression numérique indiquant la distance verticale, en twips, entre le bord supérieur de la boîte de dialogue et le haut de l'écran. Si ypos est omis, la boîte de dialogue est positionnée verticalement environ un tiers de l'écran vers le bas.
Fichier d'aide	Facultatif . Expression sous forme de chaîne de caractères qui identifie le fichier d'aide à utiliser pour fournir une aide contextuelle à la boîte de dialogue. Si *helpfile* est égal à sont mis à disposition, le *contexte* doit également être indiqué être.
Contexte	Facultatif . Expression numérique identifiant le numéro de contexte d'aide attribué par l'auteur de l'aide à la rubrique d'aide correspondante. Si context est spécifié, helpfile doit également l'être.

```
'Exemple de fonction
InputBox Dim MyInput
MyInput = InputBox("Veuillez saisir le texte", "Mon titre",
  "Exemple de texte")
DL.AddComment MaSaisie        ' Ajouter la saisie actuelle comme commentaire
```

Fonction -MsgBox

Affiche une boîte de message, attend que l'utilisateur clique sur un bouton et renvoie une valeur indiquant le bouton sur lequel l'utilisateur

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

a cliqué.

ASTUCE : utilisez la méthode DL.MsgBox2 spécifique à Docklight pour une boîte de message qui apparaît toujours sur le même écran que la fenêtre principale de Docklight Scripting. Ou consultez la fonction DL.SetUserOutput pour savoir comment créer une zone de sortie utilisateur supplémentaire et afficher des informations utilisateur supplémentaires.

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

Syntaxe

result = **MsgBox** (*prompt*[, *buttons*][, *title*][, *xpos*][, *ypos*][, *helpfile*, *context*])

Partie	Description
Invitation e	Nécessaire. Comme la fonction InputBox ci-dessus.
Boutons	Les valeurs communes facultatives sont une combinaison (somme) des constantes suivantes : Constante Valeur Description vbOKOnly 0 Bouton OK uniquement (par défaut) vbOKCancel1Boutons OK et Annuler vbAbortRetryIgnore2Boutons Annuler, Répéter et Ignorer vbYesNoCancel 3Boutons, Non et Annuler vbYesNo 4Boutonset Non vbRetryCancel5Boutons Répéter et Annuler vbCritical16Critique Message vbQuestion 32Requête d'alerte vbExclamation 48Avertissement Message vbInformation 64Message d'information vbDefaultButton1 0Le premier bouton est standard (par défaut) vbDefaultButton2256Second bouton est par défaut vbDefaultButton3512Le troisième bouton est préréglé ASTUCE : Vous trouverez une liste complète de toutes les constantes disponibles dans la documentation Microsoft VBA pour MsgBox.
Titre	En option. Comme la fonction InputBox ci-dessus.
xpos, ypos, helpfile, context e	En option. Comme la fonction InputBox ci-dessus.
Résultat	Renvoie l'action de l'utilisateur : Constante Valeur Description vbOK1OK Touche enfoncée vbCancel 2 Touche Annuler enfoncée vbAbort3Touche Annuler appuyé vbRetry 4Touche Retry appuyée vbIgnore5Touche Ignore appuyée vbYes 6Touche Oui enfoncée vbNo 7 Aucune touche enfoncée

```
'Exemple de fonction MsgBox
result = MsgBox("Effectuer ce test ?", 3,
"Mon titre") If result = 6 Then
    DL.AddComment "Bouton Oui appuyé"
ElseIf result = 7 Then
```


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

```
    DL.AddComment "Aucune touche
enfoncée" Sinon
    DL.AddComment "Annulé" Fin
si
```

Fonction -ScriptEngine

Renvoie une chaîne de caractères indiquant le langage de script utilisé.

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

Utilisez l'exemple de script suivant pour obtenir une description complète du langage de script et du numéro de version.

```
Exemple d'utilisation de la fonction
ScriptEngine                DL.AddComment
GetScriptEngineInfo

Fonction GetScriptEngineInfo
    Dim s
    s = "" ' Créer une chaîne avec les
    informations requises. s = ScriptEngine &
    " Version "
    s = s & ScriptEngineMajorVersion & "."
    s = s & ScriptEngineMinorVersion & "."
    s = s & ScriptEngineBuildVersion
    GetScriptEngineInfo = s ' Retour des résultats.
Quitter la fonction
```

10.2 Commandes de script Docklight - L'objet DL

L'objet global DL est utilisé pour accéder à des fonctions spécifiques à Docklight à partir d'un programme VBScript.

Méthodes DL

DL.AddComment
DL.ClearCommWindows
DL.GetReceiveCounter
DL.GetDocklightTimeStamp
DL.OpenProject
DL.Pause
DL.Quitter
DL.ResetReceiveCounter
DL.SendSequence
DL.StartCommunication
DL.StopCommunication
DL.StartLogging
DL.StopLogging
DL.WaitForSequence

Méthodes DL (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

Propriétés DL DL.NoOfSendSequences
DL.NoOfReceiveSequences

Fonctions de script Docklight supplémentaires
Procédures d'événement OnSend / OnReceive
Objets FileInput / FileOutput pour la lecture et l'écriture de fichiers n Canaux latéraux - utilisation de plusieurs connexions de données

10.2.1 Méthodes

10.2.1.1 AddComment

Ajoute un texte personnalisé à la fenêtre de données de communication et au fichier journal.

Valeur de retour

Vide

Syntaxe

DL.AddComment [*commentaire***] [,** *timeStampAfterComment***] [,** *lineBreakAndPadding***]**

La syntaxe de la méthode AddComment se compose de ces parties :

Partie	Description
Commentaire	Facultatif . Chaîne de caractères contenant le commentaire à ajouter à la/aux fenêtre(s) de communication ou au(x) fichier(s) journal. Si le *commentaire* est omis, crée **AddComment ne fait** qu'un saut de ligne.
timBre de tempsParCiTé	Valeur booléenne facultative. False (par défaut) = pas d'horodatage supplémentaire. True = Ajouter un horodatage après le Commentaire . L'horodatage est ajouté lors du traitement du prochain caractère de données série, et non immédiatement après l'impression du commentaire. Ceci est comparable à l'option "Horodatage supplémentaire..." dans la boîte de dialogue "Ordre de réception".
lineBreakAndPadding	Valeur booléenne facultative. True (par défaut) = des espaces supplémentaires sont ajoutés avant et après le texte afin de le séparer des données de communication. Un saut de ligne est ajouté après le commentaire. False = pas d'espaces ou de retours à la ligne supplémentaires. Ceci est particulièrement utile en combinaison avec l'option Filtre de communication, si vous voulez que la sortie d'écran proprement dite soit entièrement de la méthode **AddComment**.

Remarques

AddComment prend en charge les macros de commentaire de séquence de réception **%_S** (cloche, "bip"), **%_L** (retour à la ligne) et **%_T** (horodatage) pour plus de simplicité. Voir les macros de commentaires pour plus de détails.

Vous ne pouvez pas utiliser les caractères de contrôle ASCII tels que le code décimal 08 (retour arrière) pour émuler les fonctions du terminal ou pour formater l'affichage.

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

Le site

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

la seule exception est le code décimal 07 (cloche), qui peut être utilisé pour générer un "bip", selon le schéma sonore de Windows.

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

Exemple

```
' Exemple AddComment

DL.ClearCommWindows
DL.AddComment "Bonjour le
monde !" ' Saut de ligne
supplémentaire DL.AddComment
Utilisez l'opérateur '&' pour concaténer des chaînes de
caractères et d'autres variables.
r1 = 10
r2 = 20
DL.AddComment "Résultat 1 = " & r1 & " Résultat2 = " &
r2 ' La constante VBScript vbCrLf peut être utilisée
pour une
' retour à la ligne supplémentaire,
également DL.AddComment
DL.AddComment "Résultat 1 = " & r1 &
vbCrLf DL.AddComment "Résultat 2 = " & r2

' La désactivation des retours à la ligne et des caractères
de remplissage vous donne ' un meilleur contrôle sur la
sortie réelle
DL.AddComment vbCrLf + "Voici quelques informations", False,
False DL.AddComment "rmation. " + vbCrLf, False, False

' Un "bip" pour informer
l'utilisateur DL.AddComment
Chr(7)
```

10.2.1.2 ClearCommWindows

Efface le contenu de la fenêtre de communication. Ceci est valable pour les quatre représentations (ASCII, HEX, décimale, binaire) de la fenêtre de communication.

Valeur de retour

Vide

Syntaxe

DL.ClearCommWindows

Exemple

```
' Exemple ClearCommWindows

' Redémarrage
DL.ClearCommWindows
DL.AddComment "Test lancé !"
```

10.2.1.3 GetReceiveCounter

Renvoie la valeur actuelle du compteur d'occurrences pour la séquence de réception indiquée. Le compteur est incrémenté à chaque fois que la séquence de réception est détectée dans le flux de données entrant. Il peut être remis à zéro avec la commande ResetReceiveCounter. Les commandes OpenProject et StartCommunication remettent également le compteur d'occurrences à zéro.

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

Valeur de retour

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

Lang

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

Syntaxe

Résultat **= DL.GetReceiveCounter(** *NomOuIndex* **)**

La syntaxe de la méthode **GetReceiveCounter** se compose des parties suivantes :

Partie	Description
nameOrIndex	Obligatoire. Chaîne de caractères contenant le nom ou l'index de séquence d'une séquence de réception.

Remarques

Voir aussi WaitForSequence

Exemple

Voir WaitForSequence

10.2.1.4 GetDocklightTimeStamp

Fournit l'horodatage actuel du Docklight selon les paramètres suivants :

1. Le format du timbre à date/heure Docklight sélectionné dans la boîte de dialogue Options :
   - Horodatage
   Horodateur
   - utilisation de l'horodatage avec une précision de 1/100 de seconde
2. Les paramètres Windows pour la **région et la langue > Formats > Date courte** et **heure longue**

La fonction **GetDocklightTimeStamp** est particulièrement utile pour imprimer des informations horaires supplémentaires avec la méthode AddComment.

Chaîne de

retour

Syntaxe 1

result **=**

DL.GetDocklightTimeStamp()

Remarques (syntaxe 1)

GetDocklightTimeStamp ajoute un espace à la fin de la date/heure. Ceci pour des raisons historiques et de compatibilité. Voir la syntaxe 2 pour une version abrégée. Voir aussi la méthode AddComment.

Exemple 1

```
' Exemple GetDocklightTimeStamp

DL.ClearCommWindows
DL.StartCommunication
DL.AddComment "La communication a commencé à " &
DL.GetDocklightTimeStamp()
DL.AddComment "J'attends des données..."
```


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

```
' Boucle infinie pour éviter que le script ne se termine
immédiatement
Je
    DL.Pause 1 ' (la pause réduit la charge du CPU au
ralenti) Boucle
```

Syntaxe 2

result **= DL.GetDocklightTimeStamp([** *myDateTime* **] [,** *millisecondes* **] [,** *trimmed* **])**

Partie	Description
myDateTime	Facultativement. une variable de date (d'heure) VBScript qui /fournit des informations sur l'heure avec une résolution de "1 seconde". 0 (par défaut) = utiliser la propre base de temps de Docklights.
Millisecondes	Valeur entière optionnelle avec millisecondes correspondantes de 0..999. -1 (par défaut) = utiliser la propre base de temps de Docklights. -2 = N'ajoutez PAS la partie milliseconde.
paré	En option. True = Supprimer l'espace à la fin (voir syntaxe 1). False (par défaut) = utiliser le format original pour des raisons de compatibilité.

Remarques (syntaxe 2)

La syntaxe étendue est généralement utilisée pour formater les informations temporelles sur la séquence de réception obtenue au sein d'une procédure d'événement Sub DL_OnReceive(). Voir l'exemple 2.

L'argument *millisecondes* = -2 est utile lors de la création d'un fichier d'exportation pour un logiciel tel qu'Excel, qui attend un format d'heure Windows standard sans millisecondes.

10.2.1.5 OpenProject

Ouvre un fichier de projet Docklight existant (fichier .ptp).

Valeur de retour

Vide

Syntaxe

DL.OpenProject *filePathName*

La syntaxe de la méthode OpenProject se compose de ces parties :

Partie	Description
filePathName	Obligatoire. Chaîne de caractères contenant le chemin d'accès (répertoire et nom de fichier) du fichier de projet Docklight à ouvrir (fichier .ptp). L'extension de fichier .ptp peut être omise. Si aucun répertoire n'est indiqué, Docklight utilise le répertoire de travail actuel.

Remarques

Si *filePathName n'*est pas un fichier de projet Docklight valide ou n'existe pas, Docklight signale une erreur et l'exécution du script est interrompue.

Si *filePathName* est une chaîne de caractères vide, une boîte de dialogue de fichier

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

s'affiche pour sélectionner un fichier de projet.

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

Tous les compteurs pour l'ordre de réception sont remis à zéro lors de la (ré)ouverture d'un projet Docklight, voir la fonction ResetReceiveCounter.

Exemple

```
' Exemple OpenProject

' Chargement d'un fichier de projet Docklight
DL.OpenProject "D:\Mes fichiers Docklight\Test.ptp".

Télécharger le fichier 'Test.ptp' depuis le répertoire de
travail actuel is DL.OpenProject "Test"
```

10.2.1.6 Pause

Met en pause l'exécution du script pendant un certain nombre de millisecondes.

Valeur de retour

Vide

Syntaxe

DL.Pause *Millisecondes*

La syntaxe de la méthode Pause se compose de ces parties :

Partie	Description
Millisecondes	Obligatoire. Valeur longue pour le délai en millisecondes. La valeur minimale est 0 (**la pause** revient immédiatement). La valeur maximale est 86000000 (23,88 heures).

Remarques

Docklight en général et la fonction de pause n'offrent pas une mesure exacte du temps avec une précision de l'ordre de la milliseconde, de sorte que le délai réel peut différer de la valeur de la milliseconde.

Pendant une **pause,** aucun appel DL_OnReceive() ne peut être traité. Si vous devez traiter des événements **DL_OnReceive()** pendant la pause, consultez le code **pauseWithEvents()** décrit dans l'exemple 2.

Exemple

```
' Exemple de pause

' Envoyer une
instruction de test
DL.SendSequence "Test1"
' Délai de 5 secondes
DL.Pause 5000
' Envoyer une autre
commande DL.SendSequence
"Test2

' Boucle principale typique pour
le traitement des données Do
   DL.Pause 1 ' Réduire la charge CPU
   countSomeThings = DL.GetReceiveCounter(1)
```


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

```
    ... faire d'autres choses
Boucle
```


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

10.2.1.7 Quitter

Arrête immédiatement le script Docklight.

Valeur de retour

Void

Syntax

DL.Quit

Remarque

s

Si la communication a été lancée avec une commande de script (voir StartCommunication), la communication est également fermée. Si un fichier journal a été ouvert avec StartLogging, le fichier est fermé. Les fichiers qui ont été ouverts avec FileInput ou FileOutput sont également fermés.

L'utilisation de l'instruction "Stop" intégrée dans VBScript ou d'autres fonctions de débogage VBScript qui modifient le déroulement du programme n'est pas possible dans Docklight Scripting. Utilisez toujours l'instruction **DL.Quit** pour arrêter l'exécution du script.

10.2.1.8 ResetReceiveCounter

Réinitialise un ou tous les compteurs de concordance de séquence de réception. Réinitialise également l'algorithme de recherche qui vérifie le flux de caractères pour une séquence de réception correspondante (voir l'exemple de code ci-dessous).

Valeur de retour

Vide

Syntaxe

DL.ResetReceiveCounter [*nomIndex***]**

La syntaxe de la méthode ResetReceiveCounter se compose des parties suivantes :

Partie	Description
nameOrIndex	Facultatif . Chaîne de caractères contenant le nom ou l'index de séquence d'une séquence de réception. Si elle est indiquée, seul le compteur correspondant est remis à zéro. Si *nameOrIndex* est omis, tous les compteurs sont remis à zéro.

Remarques

Voir aussi GetReceiveCounter et WaitForSequence

Exemple

Voir WaitForSequence pour un exemple simple.

Une deuxième application est démontrée plus loin - la réinitialisation de la détection de séquence de réception chaque fois qu'une nouvelle séquence d'émission est transmise. Ceci est particulièrement utile lorsque Docklight teste un périphérique série et que la détection de séquence ne doit pas être perturbée par des paquets incomplets ou erronés reçus précédemment. Voir aussi DL_OnSend().

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

```
' Exemple ResetReceiveCounter
' Réinitialiser la reconnaissance de la séquence lorsqu'une nouvelle
séquence est envoyée
```


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

```
' Boucle infinie pour éviter que le script ne se termine
immédiatement
Je
    DL.Pause 1 ' (la pause réduit la charge du CPU au
ralenti) Boucle

Sub DL_OnSend()
    DL.ResetReceiveCounter
End Sub
```

10.2.1.9 SendSequence

Envoie une séquence d'envoi ou une séquence de données définie par l'utilisateur. Démarre la communication si elle n'est pas déjà en cours (voir StartCommunication).

Valeur de retour

Vide

Syntaxe 1

DL.SendSequence *nameOrIndex* **[,** *Paramètre***] [,** *Représentation***]**

Envoie la séquence d'envoi correspondant à *nameOrIndex*. La **SendSequence**
La syntaxe de la méthode 1 se compose de ces parties :

Partie	Description
nameOrIndex	Obligatoire. Chaîne de caractères contenant le nom de la séquence d'envoi. La première séquence d'envoi de la liste dont le nom correspond à *nameOrIndex* sera utilisée. Il est également possible de passer une valeur entière indiquant l'index de la séquence. La plage valable pour l'index des séquences va de 0 à (NoOfSendSequences - 1).
Paramètres	Facultatif . Chaîne de caractères contenant une ou plusieurs valeurs de paramètres pour une séquence d'envoi avec caractères génériques. Par défaut, les paramètres sont passés en représentation ASCII. L'espace est utilisé pour séparer plusieurs paramètres différents pour différentes plages de caractères génériques. Pour passer des paramètres en représentation HEX, décimale ou binaire, utilisez l'argument de représentation optionnel décrit ci-dessous. En représentation HEX, décimale ou binaire, la virgule (",") est utilisée comme séparateur. Séparateur entre plusieurs paramètres différents.
Représentation	Facultatif . Valeur de la chaîne pour définir le format de la liste des paramètres "A" = ASCII (standard), "H" = Hex, "D" = Décimal ou "B" = Binaire.

Remarques (syntaxe 1)

Si l'argument *parameters* indique *le* mauvais nombre de paramètres ou si la longueur des paramètres ne correspond pas à la région joker correspondante, Docklight ne génère pas d'erreur, mais applique les règles suivantes :

Si trop peu de paramètres sont indiqués ou si la chaîne de caractères des paramètres est trop courte, tous les caractères de remplacement restants sont remplis par un espace.

Si vous choisissez la *représentation* =
"A" (ASCII), les jokers sont remplis avec des espaces (code ASCII 32). Pour tous

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

les autres formats, les caractères génériques sont remplis avec le code ASCII 0.

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

Si trop de paramètres sont indiqués ou si la chaîne de caractères des paramètres est trop longue, les paramètres sont tronqués ou ignorés.

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

Syntaxe 2

DL.SendSequence "", *customSequence* **[,** *representation* **]**

Envoie une séquence de données définie par l'utilisateur. La méthode SendSequence Syntaxe 2 se compose des parties suivantes :

Partie	Description
Séquences personnalisées	Obligatoire. Chaîne de caractères contenant la séquence à envoyer. Par défaut, la séquence est transmise en représentation ASCII. Pour les données de séquence HEX, décimales ou binaires, utilisez l'argument de représentation optionnel décrites ci-dessous.
Représentation	Facultatif . Valeur de chaîne pour définir le format pour *customSequence*. "A" = ASCII (standard), "H" = HEX, "D" = décimal ou "B" = binaire.

Exemple

```
' Exemple SendSequence

' Prédéfini
Séquences d'émission ' (0) Test :
Test
' (1) Un : un<#><#><##><CR><LF>
' (2) Deux : un<?><?><?>deux<#><#><#>.

DL.StartCommunication
DL.ClearCommWindows
' Séquence d'envoi sans
paramètres DL.SendSequence "Test
' Séquence d'envoi avec un paramètre
DL.SendSequence "One", "100
' Séquence d'envoi avec deux paramètres
DL.SendSequence "Zwei", "100 20
' Transmission de deux paramètres en représentation
HEX, y compris les espaces et les caractères de
contrôle DL.SendSequence "Zwei", "20 31 20, 30 0D
0A", "H
' Envoyer des données de séquence définies par l'utilisateur, pas via une
séquence d'envoi prédéfinie
DL.SendSequence "", "Données définies par l'utilisateur

' Et maintenant une boucle et la variable de
boucle ' pour la séquence d'envoi-
Utiliser les valeurs des paramètres For i = 1 To 10
      parString = i & " " & i+1 ' Utiliser un espace pour
séparer les paramètres
      DL.SendSequence "Deux", parString
Suivant

DL.StopCommunication
```

Après l'exécution du script, la fenêtre de communication de Docklight pourrait ressembler à ceci :

```
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
```


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

```
08/05/2008 13:50:35.682 [TX] - Un 1 Deux0<CR><LF>
```


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

```
08/05/2008 13:50:35.699 [TX]        -
                            Données
                             personnalisé
                             es
08/05/2008 13:50:35.713 [TX] - Un1 Deux2
08/05/2008 13:50:35.745 [TX] - Un2 Deux3
08/05/2008 13:50:35.771 [TX] - Un3 Deux4
08/05/2008 13:50:35.807 [TX] - Un4 Deux5
08/05/2008 13:50:35.846 [TX] - Un5 Deux6
08/05/2008 13:50:35.878 [TX] - Un6 Deux7
08/05/2008 13:50:35.907 [TX] - Un7 Deux8
08/05/2008 13:50:35.922 [TX] - Un8 Deux9
08/05/2008 13:50:35.955 [TX]    - Un9
                                 Deux10
08/05/2008 13:50:35.987 [TX]    - Un10
                                 Deux11
```

10.2.1.10 StartCommunication

Ouvre le(s) port(s) de communication et active le transfert de données. Cela correspond au menu Docklight **Exécuter > Démarrer la communication**

Valeur de retour

Vide

Syntaxe

DL.StartCommunication

Remarques

Les méthodes SendSequence, WaitForSequence et UploadFile ouvrent automatiquement le(s) port(s) de communication s'ils n'ont pas été ouverts auparavant avec la méthode **StartCommunication**.

Voir aussi StopCommunication.

10.2.1.11 StopCommunication

Arrête la transmission des données et ferme le(s) port(s) de communication. Cela correspond au menu Docklight **Run > Stop communication.**

Valeur de retour

Vide

Syntaxe

DL.StopCommunication

Remarques

Vous trouverez de plus amples informations dans la méthode StartCommunication.

10.2.1.12 StartLogging

Crée de nouveaux fichiers journaux et démarre l'enregistrement des données sérielles entrantes et sortantes. Cela correspond au menu Docklight **Démarrer la journalisation des communications ...**

Valeur de retour

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

Vide

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

Syntaxe

DL.StartLogging *baseFilePath* **[,** *appendData***] [,** *représentations***] [,** *format***] [,** *grande vitesse***] [,** *noHeaders***]**

La syntaxe de la méthode StartLogging se compose des parties suivantes :

Partie	Description
baseFilePath	Obligatoire. Chaîne de caractères contenant le répertoire et le nom du fichier de base pour le ou les fichiers journaux.
appendData	Valeur booléenne facultative. True (par défaut) = Joindre les nouvelles données au(x) fichier(s) journal existant(s). False = le ou les fichiers journaux existants sont écrasés. Les données de journalisation précédemment enregistrées sont perdues.
Représentations	Chaîne de caractères facultative permettant de sélectionner la présentation du fichier journal. "A" (ASCII), "H" (HEX), "D" (décimal) et/ou "B" (binaire). La valeur par défaut est "AHDB". (créer les quatre représentations ASCII, HEX, décimale, binaire).
Format	Facultatif Valeur entière. 0 (par défaut) = créer des fichiers texte simples (.txt) 1 = créer des fichiers HTML pour les navigateurs web (.htm) 2 = créer des fichiers RTF Rich Text Format (.rtf) REMARQUE : Pour des raisons de compatibilité avec la V2.2. et les versions antérieures, il est également possible d'utiliser : Faux = texte simple (.txt) Vrai = HTML (.htm)
Haute vitesse	Valeur booléenne facultative. False (par défaut) = non utilisé True = désactiver la fenêtre de communication pendant la journalisation (par exemple, pour surveiller les connexions à haut débit sur un PC lent).
noHeaders	Valeur booléenne facultative. False (Default) = crée un en-tête standard "Docklight Log File started..." après l'ouverture du fichier. Lors de la fermeture du fichier, un pied de page "Docklight Log File stopped" est créé. True = Ne pas créer d'informations d'en-tête ou de pied de page supplémentaires.

Remarques

Pour plus d'informations sur les fonctions et les arguments de **StartLogging** décrits ci-dessus, consultez la rubrique Consignation et analyse d'un test et la boîte de dialogue Créer un ou plusieurs fichiers journaux.

Si *baseFilePath* est une chaîne vide, une boîte de dialogue de fichier s'affiche pour permettre de sélectionner le chemin d'accès au fichier journal et le nom du fichier de base.

Si **StartLogging** est appelé alors qu'un autre fichier journal est encore ouvert par un appel précédent à StartLogging, le fichier est fermé et le nouveau fichier est créé/ouvert. Cela permet de modifier le nom du fichier journal sans perdre de données.

L'indicateur noHeaders est particulièrement utile si vous souhaitez créer des données de journal sans horodatage. Vous pouvez alors facilement comparer le résultat avec les tests précédents en utilisant un outil de comparaison de fichiers.

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

Exemple

```
' Exemple de StartLogging

DL.ClearCommWindows
DL.StartLogging "C:\DocklightLogging"
```


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

```
' - ouvre quatre fichiers journaux :
' 'C:\DocklightLogging_asc.txt'
' 'C:\DocklightLogging_hex.txt'
' 'C:\DocklightLogging_dec.txt'
' 'C:\DocklightLogging_bin.txt'
' Attendre 5 secondes DL.Pause
5000
' Fermez les quatre
Fichiers journaux
DL.StopLogging
```

Exemple 2

L'exemple suivant montre comment insérer un horodateur dans le nom du fichier journal et créer un nouveau fichier journal toutes les heures

```
Exemple 'Un fichier journal par heure'.

Il s'agit du chemin de base et de l'emplacement
où le(s) fichier(s) de journalisation doit
(doivent) être enregistré(s) Const
BASE_FILE_PATH = "logfile_"
' Créer des fichiers journaux
ASCII et HEX Const
LOG_REPRESENTATIONS = "AH

currentLogFileName =
""
DL.StartCommunication
Do
    newLogFileName = getFileName()
    ' Heure de début d'un nouveau
    fichier ?
    If newLogFileName <> currentLogFileName Then
        DL.StartLogging newLogFileName, True,
LOG_REPRESENTATIONS
     currentLogFileName = newLogFileName
    End If
    DL.Pause 1 ' Réduire la
charge CPU Boucle

Fonction getFileName()
    dt = Maintenant
    ' Rédigez un nom de fichier.
    ' Les fonctions Right() garantissent que tous les mois,
    jours, ' heures sont imprimés avec deux décimales.
    getFileName = BASE_FILE_PATH & Year(dt) & "_" & Right("0" &
Month(dt), 2) & "_" & Right("0" & Day(dt), 2) & "_" & Right("0"
& Hour(dt), 2) & "H"
Quitter la fonction
```

10.2.1.13 StopLogging

Arrête la journalisation et ferme le(s) fichier(s) de journalisation actuellement ouvert(s). Cela correspond au menu Docklight **Arrêter la journalisation des communications**.

Valeur de retour

Vide

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

Syntaxe

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

DL.StopLogging

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

Remarques

Pour plus d'informations sur les fichiers journaux, consultez la méthode StartLogging.

10.2.1.14 WaitForSequence

Attend un ou plusieurs événements d'une Receive Sequence et renvoie la valeur de compteur correspondante (voir GetReceiveCounter). Démarre la communication, si elle n'est pas déjà en cours (voir StartCommunication).

Valeur de retour

Lang

Syntaxe

Résultat **= DL.WaitForSequence(** *nameOrIndex* **[,** *maxCounter***] [,** *timeout***])**

La syntaxe de la méthode WaitForSequence se compose de ces parties :

Partie	Description
nameOrIndex	Obligatoire. Chaîne de caractères contenant le nom de la chaîne de réception à compter. La première séquence de réception de la liste dont le nom correspond à *nameOrIndex* sera utilisée. Il est également possible de passer une valeur entière indiquant l'index de la séquence. Index de séquence valide est compris entre 0 et (NoOfReceiveSequences - 1).
maxCounter	Facultatif . Nombre long contenant la valeur limite du compteur jusqu'à ce que la fonction revienne. La valeur par défaut est 1 (un) : **WaitForSequence** revient après que la première occurrence de la séquence de réception a été détectée. Dans ce cas, la valeur de retour est 1. Si *maxCounter* est -1, **WaitForSequence n**'utilise pas de limite de compteur. Il ne revient qu'après un timeout (voir ci-dessous). Utiliser *maxCounter* = -1 pour compter toutes les occurrences d'une séquence de réception dans un temps limité du temps.
Délai d'attente	Facultatif . Nombre long indiquant un délai d'attente supplémentaire en millisecondes. La valeur par défaut est -1 (pas de dépassement de temps). La valeur maximale est 86000000 (23,88 heures).

Remarques

La méthode **WaitForSequence** vérifie le nombre de "hits" pour cette séquence de réception depuis que la communication a été lancée (voir StartCommunication) ou que le compteur a été remis à zéro (voir ResetReceiveCounter). **WaitForSequence** attend que le nombre de "hits" indiqué par le *maxCounter* soit constaté.

Une application de base de **WaitForSequence** est l'attente d'une réponse spécifique après l'envoi d'une commande de test à votre périphérique série. Pour être sûr de ne pas manquer une réponse très rapide de votre périphérique, utilisez la séquence de commandes suivante :
1. Remettez d'abord le(s) compteur(s) à zéro avec ResetReceiveCounter.
2. Envoyez votre commande de test avec SendSequence
3. Utilisez maintenant **WaitForSequence** pour attendre la réponse attendue

Il est très important d'utiliser **ResetReceiveCounter** avant **SendSequence. ResetReceiveCounter ne** remet pas seulement le compteur de reconnaissance à zéro, mais réinitialise également la comparaison des caractères, de sorte que tous les caractères reçus précédemment ne sont pas pris en compte lors de la recherche

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

d'une correspondance de séquence. Voir aussi

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

les notes sur la recherche de jokers pour obtenir des informations supplémentaires sur la manière dont Docklight gère la comparaison de motifs de Receive Sequence.

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

Pendant une **WaitForSequence,** les appels DL_OnReceive() ne peuvent pas être traités. Si vous devez traiter des événements **DL_OnReceive()** pendant l'attente, consultez le code **pauseWithEvents()** décrit dans OnReceive exemple 2.

Si vous devez attendre qu'*une* des séquences de réception soit déclenchée, la procédure DL_OnReceive() offre la solution. Voir l'exemple OnReceive 3.

Exemple

```
' Exemple WaitForSequence

' Comptez le nombre d'occurrences de
la première séquence de réception dans un
intervalle de ' 10 secondes.
Nécessite au moins la définition d'une séquence de réception

DL.StartCommunication
DL.ClearCommWindows
Résultat = DL.WaitForSequence(0 , -1, 10000)
DL.AddComment vbCrLf & vbCrLf & "Séquence de réception #0, nombre d'occurrences
= " & résultat
' Autre possibilité de lire ensuite le compteur DL.AddComment
"Receive Sequence #0, hit count = " &
DL.GetReceiveCounter(0)

' Envoyez la première séquence d'envoi et attendez une réponse
de l'appareil (pas de timeout)
DL.AddComment vbCrLf & vbCrLf & "Sending data and waiting for
Receive Sequence #0" (Envoi de données et attente de la
séquence de réception #0)
DL.ResetReceiveCounter
DL.SendSequence 0
DL.WaitForSequence 0
```

10.2.2 Méthodes (avancés)

10.2.2.1 CalcChecksum

Renvoie une somme de contrôle ou une valeur CRC pour une séquence donnée ou une partie de séquence.

La méthode CalcChecksum est une fonction de script Docklight avancée qui nécessite quelques connaissances sur les sommes de contrôle dans les protocoles d'application série et sur la manière dont Docklight gère les données d'envoi/réception en général.

ASTUCE : Nous vous recommandons de lire la section Calcul et validation des sommes de contrôle en guise d'introduction. Si les termes et paramètres spécifiques au CRC vous semblent confus, vous trouverez quelques informations de base dans le glossaire CRC.

Chaîne de

caractères de

retour

Syntaxe

result **= DL.CalcChecksum(** *checksumSpec*, *dataStr*, **[**, *representation***] [**, *startPos***]**
[, *endPos***] [**, *bigEndian* **])**

La syntaxe de la méthode CalcChecksum se compose des parties suivantes :

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

Partie	Description

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

checksumSpec	Obligatoire. Chaîne de caractères indiquant l'algorithme de somme de contrôle et ses paramètres. **CalcChecksum** prend en charge les noms prédéfinis pour les algorithmes de somme de contrôle courants, ou vous pouvez passer une spécification CRC générique pour le calcul de CRC plus exotiques. Les noms prédéfinis sont : "MOD256", "XOR", "CRC-8", "CRC-CCITT", "CRC-16", "CRC-MODBUS" et "CRC-32". Voir le format checksumSpec pour la version complète Spécification du format.
dataStr	Obligatoire. Valeur de chaîne contenant la séquence d'entrée pour le calcul de la somme de contrôle, telle qu'elle est renvoyée par la fonction OnSend_GetData(), par exemple.
Représentation	Facultatif . Valeur de la chaîne pour définir le format de la séquence *dataStr* : "H" = Hex (par défaut), "A" = ASCII , "D" = Décimal ou "B" = Binaire.
startPos	Facultatif Valeur entière. Indique la position des caractères à partir de laquelle le calcul doit commencer. La valeur par défaut est 1 (début de la *dataStr* Sequence). *startPos* accepte également des valeurs négatives, par exemple -1 pour "dernier caractère", -2 pour "2e caractère de la fin", -3 pour "3e caractère de la fin".
endPos	Facultatif Valeur entière. Indique le dernier caractère à inclure dans le calcul. La valeur par défaut est la taille de la dataStr Sequence. *endPos* accepte également les valeurs négatives, voir *startPos* ci-dessus.
bigEndian	Facultatif. Valeur booléenne permettant de définir l'ordre des octets pour le *résultat*. True (par défaut) : Utiliser l'ordre des octets big-endian (le premier caractère est le plus significatif) False (faux) : Utiliser l'ordre des octets Little-Endian (le premier caractère est de poids faible)

Remarques

La valeur de retour est une chaîne de caractères contenant le CRC/somme de contrôle au format de séquence HEX Docklight, par exemple "CB F4 39 26". Le nombre d'octets HEX renvoyés dépend de la largeur de l'algorithme de somme de contrôle. Voir l'exemple de script et la sortie dans la fenêtre de communication ci-dessous.

A l'aide de **CalcChecksum**, il est possible de générer à la volée des sommes de contrôle spécifiques pour les séquences d'envoi ou d'utiliser des validations de sommes de contrôle étendues pour les données reçues. Voir la procédure d'événement Sub DL_OnSend() pour plus de détails.

Les sommes de contrôle standard peuvent déjà être traitées sans code de script, en utilisant la partie somme de contrôle des boîtes de dialogue Editer la séquence d'envoi / Editer la séquence de réception. Voir aussi l'exemple correspondant pour le protocole Modbus.

Exemple

```
' Exemple CalcChecksum

DL.ClearCommWindows

DL.AddComment
DL.AddComment "Somme de contrôle simple (Mod 256) pour
```


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

```
'123456789'". DL.AddComment "CalcChecksum = " &
DL.CalcChecksum("MOD256", "123456789", "A")
```


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

```
DL.AddComment
DL.AddComment "CRC à 8 bits (CRC-DOW) pour '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 bits CRC (CRC-16) pour '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-16",
"123456789", "A")

DL.AddComment
DL.AddComment "16 bits CRC (CRC-Modbus) pour '123456789' dans
'LittleEndian' - l'octet inférieur en premier".
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-Modbus",
"123456789", "A", , ,False)

DL.AddComment
DL.AddComment "CRC à 16 bits (CRC-CCITT) pour '123456789'".
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-CCITT",
"123456789", "A")
DL.AddComment "Maintenant, faites la même chose, mais
indiquez vous-même tous les détails du CRC..."
DL.AddComment "Somme de contrôle = " &
DL.CalcChecksum("CRC:16,1021,FFFF,0000,No,No", "123456789",
"A")

DL.AddComment
DL.AddComment "CRC 32 bits (CRC-32) pour '123456789'"
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-32",
"123456789", "A")

DL.AddComment
DL.AddComment "Un CRC 32 bits (CRC-32) sur les 5 premiers
octets de la séquence HEX 01 02 03 04 05 FF FF FF, le
résultat est Little Endian / l'octet le plus faible en
premier".
DL.AddComment "CalcChecksum = " & DL.CalcChecksum("CRC-32", "01
02 03 04 05", "H", 1, 5, False)
```

Le code de script ci-dessus génère la sortie suivante dans la fenêtre de communication du Docklight :

```
Somme de contrôle simple (mod 256) pour
'123456789' CalcChecksum = DD

CRC 8 bits (CRC DOW) pour '123456789
CalcChecksum = A1

16 bits CRC (CRC-16) pour '123456789
Somme de contrôle = BB 3D

16 bits CRC (CRC-Modbus) pour '123456789' dans 'LittleEndian'
- l'octet inférieur en premier
CalcChecksum = 37 4B

16 bits CRC (CRC-CCITT) pour '123456789
CalcChecksum = 29 B1
Maintenant, faites la même chose, mais indiquez vous-même tous
les détails du CRC... Somme de contrôle CRC = 29 B1

32 bits CRC (CRC-32) pour '123456789
CalcChecksum = CB F4 39 26
```


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

```
 Un CRC de 32 bits (CRC-32) sur les 5 premiers octets de la
séquence HEX 01 02 03 04 05 FF FF FF, le résultat est Little
Endian / l'octet le plus bas en premier
 Somme de contrôle = F4 99 0B 47
```

10.2.2.2 ConvertSequenceData

Convertit les données de séquence en/à partir d'un nombre à virgule flottante, d'un nombre entier ou d'autres types de données courants dans les applications techniques.

Chaîne de

caractères de

retour

Syntaxe

result **= DL.ConvertSequenceData(** *conversionType*, *source*, **[**, *representation*] **[**, *bigEndian* **])**

La syntaxe de la méthode ConvertSequenceData se compose des parties suivantes :

Partie	Description
conversionType	Obligatoire. Chaîne de caractères indiquant le type de conversion et la direction. Voir ci-dessous pour la liste des conversions et des exemples.
Source	Nécessaire. Chaîne de données d'entrée pour la conversion. Il peut s'agir d'une séquence de docklight, par exemple "4B 06 9E 3F", ou d'une chaîne de caractères contenant la valeur de l'application, z. B. "1,234567". Voir ci-dessous pour plus de détails.
Représentation	Facultatif : il est facultatif. Format de la chaîne de caractères de la séquence (soit *source*, soit *résultat*, selon le *conversionType*) : "H" = Hex (par défaut), "D" = Décimal ou "B" = Binaire.
bigEndian	Facultative. Valeur booléenne permettant de définir l'ordre des octets pour les conversions en nombres entiers ou en virgule flottante. True (par défaut) : Utiliser l'ordre des octets big-endian (le premier caractère est le plus significatif) False (faux) : Utiliser l'ordre des octets Little-Endian (le premier caractère est de faible valeur)

L'argument *conversionType* prend en charge les valeurs et les types de conversions suivants :

Valeur	Description
"toSingle"	Convertit la *source* en un nombre à virgule flottante avec une précision simple. *Source* : Séquence IEEE de précision simple (32 bits) *Résultat* : Chaîne de nombres à virgule flottante au format non localisé, utilisant le point (".") comme séparateur décimal. exemple : `DL.ConvertSequenceData("toSingle", "3F 9E 06 4B")` Rendement : `1,234567`

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

"fromSingle"	Conversion de la *source* en une séquence IEEE de précision simple (32 bits) *Source* : Chaîne avec nombre en virgule flottante. Les points (".") et les virgules (",") sont tous deux acceptés comme séparateurs décimaux. *Résultat* : données de séquence 32 bits Exemple : `DL.ConvertSequenceData("fromSingle", "1.234567")` retour : `3F 9E 06 4B`

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

"toDouble"	Convertit la *source* en un nombre à virgule flottante avec une précision double. *Source* : Séquence IEEE à précision double (64 bits) *Résultat* : Chaîne avec nombre à virgule flottante dans un format non localisé (voir ci-dessus) exemple :

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

	`DL.ConvertSequenceData("toDouble", "103 154 149 160 081 161 036 075", "D", False)` Rendement : `9,87987987987E+53`
"fromDouble"	Conversion en une séquence IEEE à double précision (64 bits) *Source* : chaîne de caractères avec nombre en virgule flottante. Le point (".") et la virgule (",") sont tous deux acceptés comme séparateurs décimaux. *Résultat* : données de séquence 64 bits Exemple : `DL.ConvertSequenceData("fromDouble", "9.87987987E+53", "D", False)` Revenus : `103 154 149 160 081 161 036 075`
"deTexte"	Convertit un texte en clair en une séquence hexagonale, décimale ou binaire. par ex. `DL.AddComment DL.ConvertSequenceData("fromText", "Bonjour le monde")` Retours : `48 65 6C 6C 6F 20 57 6F 72 6C 64` *bigEndian* = false : Si cette option est utilisée pour "fromText", la séquence résultante est sans séparateur, par exemple `48656C6C6F20576F726C64`

"jusqu'au nombre entier16 " "du nombre entier1 6"	Conversion vers/depuis une valeur entière signée de 16 bits Exemples :
	`DL.ConvertSequenceData("toInteger16", "80 00")`
	Retour : `-32768`
	`DL.ConvertSequenceData("fromInteger16", "-1")`
	retour : `FF FF`

"toUnsigned16" "fromUnsigned1 6"	Comme "toInteger16" / "fromInteger16", mais pour des données entières non signées de 16 bits
	Exemples :
	`DL.ConvertSequenceData("toUnsigned16", "80 00")`
	retour : `32768`
	`DL.ConvertSequenceData("fromUnsigned16", "65535", "D")`
	Retour : `255 255`

"toInteger32" "fromInteger32"	Conversion vers/depuis une valeur entière signée de 32 bits Exemples :
	`DL.ConvertSequenceData("toInteger32", "00 00 00 80", "H", False)`
	Retour : `-2147483648`
	`DL.ConvertSequenceData("fromInteger32", "-2", "H", False)`
	retour : `FE FF FF FF`

"toUnsigned32" "fromUnsigned3 2"	Comme "toInteger32" / "fromInteger32", mais pour des données entières non signées de 32 bits
	Exemples :
	`DL.ConvertSequenceData("toUnsigned32", "FF 00 FF 00")`
	retour : `4278255360`
	`DL.ConvertSequenceData("fromUnsigned32", "21121977", "D")`
	Retour : `001 066 075 185`

"toBool"	Renvoie "True" si le premier caractère source est <> 0 Exemple :
	`DL.ConvertSequenceData("toBool", "00")`
	renvoie : `False`
	`DL.ConvertSequenceData("toBool", "01 00")`
	retour : `True`

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

"àTexte"	Convertit les données de séquence en une chaîne de texte contenant uniquement des caractères d'impression (voir jeu de caractères ASCII). Les codes ASCII 0 - 31 et 127 - 255 sont filtrés et n'apparaissent pas dans le résultat. *Source* : séquence contenant les données originales, y compris les codes de caractères non imprimables *Résultat* : le texte ASCII qui n'utilise que les codes ASCII 32 - 126 Exemple : `DL.ConvertSequenceData("toText", "FF 48 65 6C 6C 6F 21 0D 00 00")` retour : `Bonjour !`

Remarques

Vérifiez soigneusement votre spécification de protocole concernant le format des données, y compris l'endianness (little endian / big endian).

Si vous utilisez le résultat d'une conversion "toSingle" ou "toDouble" pour d'autres calculs, n'oubliez pas que *le résultat* peut être une chaîne de caractères non numérique comme "NaN" (pas de nombre) ou "Inf" (infini).

Notez que "toText" n'est pas la même chose que la lecture d'une chaîne de données en représentation ASCII ("A"). Exemple :

```
DL.AddComment DL.OnSend_GetData("A")
DL.AddComment DL.ConvertSequenceData("toText",
DL.OnSend_GetData("H"))
```

pourrait renvoyer ce qui suit :

```
-Bonjour!<CR><NUL><NUL>
Bonjour !
```

10.2.2.3 GetChannelSettings

Renvoie les paramètres actuels du canal de communication (numéro de port COM ou adresse TCP, paramètres du port série).

REMARQUE : **GetChannelSettings** est un complément à la méthode SetChannelSettings et est destiné aux applications de script Docklight avancées qui nécessitent un contrôle des paramètres du canal de communication.

Chaîne de

caractères de

retour

Syntaxe

Resultat **= DL.GetChannelSettings([***channelNo***])**

La syntaxe de la méthode GetChannelSettings se compose des parties suivantes :

Partie	Description
channelNo	Facultatif . Nombre entier indiquant le canal de communication si le mode de communication : surveillance est utilisé. La valeur par défaut est 1 (canal 1).

Remarques

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

GetChannelSettings renvoie une chaîne de caractères contenant les paramètres série ou TCP actuels pour le canal de communication indiqué.

Si le canal est une interface série, la valeur de retour a le format suivant : *COMxxx* : *BaudRate*, *Parité*, *DataBits*,
StopBits, *FlowControl*, *ParityErrorChar*
e.g. "COM1:9600,NONE,8,1,OFF,63"

Si le canal est un client TCP, la valeur de retour est l'adresse IP actuelle et le numéro de port TCP, par exemple "192.0.0.1:3001".

Si le canal est un serveur TCP, la valeur de retour est la chaîne de caractères "SERVER :" plus le numéro de port TCP, par exemple "SERVER:3001".

Voir également la méthode SetChannelSettings pour un aperçu détaillé du format de données de la valeur de retour et un exemple plus complexe de manipulation des paramètres de canal pendant l'exécution du script.

Exemple

```
' Exemple GetChannelSettings

DL.AddComment "Comm. Réglages du canal 1 = " &
DL.GetChannelSettings()
La commande suivante ne fonctionnera que
' lorsque le mode de communication du docklight est
'Surveillance (réception uniquement)
DL.AddComment "Comm. Réglages du canal 2 = " &
DL.GetChannelSettings(2)
```

Cet exemple pourrait générer la sortie suivante dans la fenêtre de communication Docklight :

```
Comm. Canal 1 Réglages = COM1:9600,NONE,8,1,OFF,63

Comm. Canal 2 Réglages

= SERVER:10001
```

10.2.2.4 GetChannelStatus

Renvoie l'état actuel du canal de communication (fermé, ouvert, en attente de connexion TCP ou erreur).

Valeur de

retour Entier

Syntaxe

Resultat **= DL.GetChannelStatus([***channelNo***])**

La syntaxe de la méthode GetChannelStatus se compose des parties suivantes :

Partie	Description
channelNo	Facultatif . Nombre entier indiquant le canal de communication lorsque le mode de communication : surveillance ou les canaux latéraux sont utilisés. La valeur par défaut est 1 (canal 1).

Remarques

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

GetChannelStatus renvoie les valeurs suivantes :

Résultat	Description
0	Le canal est fermé, la communication est arrêtée (voir aussi StopCommunication)
1	Le canal est ouvert et prêt à envoyer/recevoir des données. Mode TCP serveur ou TCP client : connexion établie.
2	Mode TCP serveur ou TCP client : en attente de connexion. Port COM avec contrôle de flux matériel RTS/CTS : attente du signal Handshake.
3	Erreur de canal, par exemple après une commande SetChannelSettings spécifiant un numéro de port COM inexistant.

Voir aussi SetChannelSettings et GetChannelSettings.

Exemple

```
' Exemple GetChannelStatus
' (nécessite un docklight en mode émission/réception)

DL.ClearCommWindows

DL.AddComment "Accès au port COM" DL.SetChannelSettings
"COM3:9600,NONE,8,1", 1
DL.AddComment "GetChannelStatus avant StartCommunication = " &
DL.GetChannelStatus(1)
DL.StartCommunication
DL.AddComment "GetChannelStatus après StartCommunication = " &
DL.GetChannelStatus(1)
DL.StopCommunication

DL.AddComment
DL.AddComment "Mode client TCP
DL.AddComment "Se connecter à docklight.fr ..."
DL.SetChannelSettings "docklight.de:80", 1
DL.StartCommunication
' attendre d'être
connecté Do
    commStatus = DL.GetChannelStatus(1)
    DL.AddComment "GetChannelStatus = " & commStatus
    DL.Pause 10
Boucle jusqu'à commStatus <>
2 If commStatus = 1 Then
    DL.AddComment "Connecté".
Else
    DL.AddComment
"Erreur !" End If
DL.StopCommunication
```

Après avoir exécuté le script sur un ordinateur avec un port COM3 intégré (par exemple un modem) et une connexion Internet, la fenêtre de communication pourrait ressembler à ceci :

```
Accès au port COM
GetChannelStatus avant StartCommunication = 0
GetChannelStatus après StartCommunication = 1

Mode client TCP
```


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

```
Se connecter avec docklight.fr ...
```


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

```
GetChannelStatus = 2
GetChannelStatus = 2
GetChannelStatus = 2
GetChannelStatus = 2
GetChannelStatus = 1
Connecté.
```

10.2.2.5 GetCommWindowData

Renvoie le contenu cumulé de la mémoire tampon de la fenêtre de

communication. REMARQUE : Cette méthode est destinée à des applications

spécifiques. Pour de nombreux
Les cas d'utilisation standard sont les procédures d'événement OnSend / OnReceive ou les méthode GetReceiveComments est la solution préférée.

Chaîne de

caractères de

retour

Syntaxe

Résultat = **DL.GetCommWindowData([***Représentation***])**

La syntaxe de la méthode **GetCommWindowData** se compose des parties suivantes :

Partie	Description
Représentation	Obligatoire. Valeur de la chaîne pour définir le format de tampon de fenêtre souhaité : "A" = ASCII (par défaut), "H" = Hex, "D" = Décimal ou "B" = Binaire.

Remarques

Seule une *représentation* activée dans Options Docklight - Modes de la fenêtre de communication peut être utilisée. Par défaut, il s'agit de l'ASCII, du HEX et du décimal. Si nécessaire, chargez d'autres options avec LoadProgramOptions**.**

La taille maximale du tampon GetCommWindowData est de 128000 caractères. S'il y a plus de données de communication sans que **GetCommWindowData** soit appelé, les données les plus anciennes sont effacées.

10.2.2.6 GetEnvironment

Donne la valeur d'une variable d'environnement Windows dans le profil utilisateur actuel ou la valeur d'une des variables d'environnement spécifiques à Docklight décrites ci-dessous.

Chaîne de

caractères de

retour

Syntaxe

Résultat **= DL.GetEnvironment(** *Nom* **)**

La syntaxe de la méthode GetEnvironment se compose des parties suivantes :

Partie	Description

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

Nom	Nécessaire. *nom* peut être : 1) Le nom de la variable d'environnement Windows. (Sans les Caractères % utilisés dans le shell de commande Windows **cmd.exe**).

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

	2) Un des noms spécifiques au Docklight ci-dessous

Variables d'environnement Docklight Scripting

Nom	Description
DOCKLIGHT_VERSION	Version de l'application Docklight Scripting
DOCKLIGHT_SCRIPTDIR	le dossier dans lequel le script sera exécuté
DOCKLIGHT_DIALOGDIR	le dossier utilisé pour la dernière boîte de dialogue de fichier de script utilisée
DOCKLIGHT_PORTLIST	Liste des ports COM disponibles sur ce PC
DOCKLIGHT_SENDSEQ	Liste de tous les noms de séquences d'envoi dans le projet Docklight actuel (fichier .ptp)
DOCKLIGHT_RECEIVESEQ	Liste de tous les noms de séquences de réception
DOCKLIGHT_SENDSEQDEF	Liste de toutes les séquences d'émission **Nom** et **séquence** au format HEX. **Le nom** et **la séquence** sont retournés dans des lignes de texte séparées
DOCKLIGHT_SENDSEQDEF : *Nom de la séquence*	Répertorie la définition uniquement pour les noms de séquence correspondant à *SequenceName*. *SequenceName* peut contenir des jokers, par exemple, vous pouvez utiliser : DOCKLIGHT_SENDSEQDEF:Test*.
DOCKLIGHT_RECEIVESEQDEF	comme DOCKLIGHT_SENDSEQDEF, mais pour les séquences Receive

Remarques

La liste des variables d'environnement utilisées dans l'exemple ci-dessous n'est

qu'un exemple. ASTUCE : vous pouvez obtenir une liste des variables disponibles

dans votre profil utilisateur actuel en créant un compte *Windows*
Fenêtre du processeur de commande (touche Windows + R, puis taper **cmd**), puis taper **SET** et appuyez sur Entrée.

REMARQUE : Dans Docklight Scripting V2.0 et avant, cette méthode s'appelait **GetEnvironmentVariable**. L'ancien nom est encore pris en charge pour des raisons de compatibilité. Il a été modifié afin d'éviter toute confusion avec la fonction API Windows du même nom.

Exemple

```
' Exemple

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 en cours
" & 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

Suivant

Sur un ordinateur portable *Windows* 10, la sortie de la fenêtre de communication pourrait ressembler à ceci :

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

Exécution de Docklight Scripting Docklight Scripting V2.4.5
ALLUSERSPROFILE = C:\ProgramData
APPDATA = C:\NUtilisateurs\Docklight\AppData\Roaming
COMPUTERNAME = DOCK-OH
HOMEDRIVE = C :
HOMEPATH =
\Utilisateurs\Docklight
LOCALAPPDATA =
C:\NUtilisateur\NDocklight\NAppData\NLocal LOGONSERVER
= \\DOCK-OH
NOMBRE_DE_PROCESSEURS = 4
OS = Windows_NT
PROCESSOR_ARCHITECTURE = x86
PROCESSOR_IDENTIFIER = Intel64 Famille 6 Modèle 78 Stepping 3,
GenuineIntel
PROCESSOR_LEVEL = 6
PROCESSOR_REVISION = 4e03
PUBLIC = C:\NUtilisateurs\Npublic
TEMP = C:\Users\docklight\AppData\Local\Temp
TMP = C:\Users\docklight\AppData\Local\Temp
USERDOMAIN = DOCK-OH
USERNAME = dockligh t PROFIL
UTILISATEUR =
C:\Utilisateur\Ndocklight

10.2.2.7 GetHandshakeSignals

Renvoie l'état actuel du signal de Handshake (CTS, DSR, DCD, RI) sous forme de valeur binaire entière, comme le fait le caractère de fonction de séquence de réception ' !

Valeur de

retour Entier

Syntaxe

Résultat **= DL.GetHandshakeSignals()**

Remarques

Le résultat est une valeur de bit avec les composantes suivantes :

N° de bit	Valeur décimale	Signal Handshake
0	001	CTS = élevé
1	002	DSR = élevé
2	004	DCD = élevé
3	008	RI (Ring Indicator) = élevé

Dans les applications Tap Pro / Tap 485, **GetHandshakeSignals** renvoie l'ensemble étendu suivant d'états de signaux de poignée de main :

N° de bit	Valeur décimale	Signal Handshake
0	001	CTS = High (côté DCE / canal de réception Docklight 2)
1	002	DSR = haut (côté DCE / canal 2)
2	004	DCD = High (côté DCE / canal 2)
3	008	RI (Ring Indicator) = haut (côté DCE / canal 2)
4	016	RTS = High (côté DEE / canal 1)

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

5	032	DTR = High (côté DEE / canal 1)

Voir aussi SetHandshakeSignals pour contrôler l'état des lignes RTS et DTR.

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

Exemple

```
' Exemple GetHandshakeSignals
DL.StartCommunication
Je
    DL.AddComment DL.GetDocklightTimeStamp() & " -
GetHandshakeSignals() = " & DL.GetHandshakeSignals()
    DL.Pause 200
Boucle
```

Exemple de sortie dans la fenêtre de communication :

```
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
```

REMARQUE : Il peut s'écouler 5 à 10 millisecondes après **le démarrage de la communication** avant que le message ne s'affiche.
GetHandshakeSignals signale l'état correct du signal.

10.2.2.8 GetKeyState

Renvoie l'état d'une touche de clavier, d'un bouton de souris ou d'une entrée similaire.

Syntaxe

Résultat **= DL.GetKeyState(** *virtKey* **)**

La syntaxe de la méthode GetKeyState se compose des parties suivantes :

Partie	Description
virtKey	Valeur entière requise. Un code clé ASCII ou virtuel. Si la clé demandée est un chiffre de 0 à 9 ou une lettre de A à Z, le code ASCII correspondant est utilisé. Vous trouverez d'autres touches et boutons dans le tableau **des codes des touches virtuelles** dans les *Microsoft Windows* Docs.

Remarques

L'argument **GetKeyState** et la valeur de retour correspondent à la fonction Windows GetKeyState de la bibliothèque **Winuser.h**.

Pour les touches virtuelles (par ex. F1-F12, touches multimédia, boutons de la souris, etc.), vous pouvez utiliser les déclarations VBScript dans le fichier **GetKeyState_ImportFile_VirtualKeys.pts** du dossier ScriptSamples\Extras\GetKeyState_Example. Nous vous recommandons de copier le fichier dans votre propre dossier de script et de l'inclure avec la directive #include.

Exemple

```
' Exemple GetKeyState

' Deux déclarations de clés virtuelles,
extraites de
GetKeyState_ImportFile_VirtualKeys.pts
Const VK_F2=&h71
```


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

```
Const. VK_LBUTTON= &h01
```


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

Je

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

```
      DL.AddComment "Exemple pour la touche de fonction F2 :
DL.GetKeyState(VK_F2)=" & DL.GetKeyState(VK_F2)
      If keyDown(Asc("D")) Then
            DL.AddComment "La touche D
            est enfoncée".
      Fin quand
      Si toucheAb(VK_F2) Alors
            DL.AddComment "La touche F2
            est
      enfoncé" End If
      If keyDown(VK_LBUTTON) Then
            DL.AddComment "Le bouton gauche de la souris
      est enfoncé" End If
      DL.Pause 500
Boucle

Fonction keyDown(virtKey)
      keyDown = (DL.GetKeyState(virtKey) And 128) > 0
Quitter la fonction
```

Exemple de sortie dans la fenêtre de communication :

```
Exemple pour la touche de fonction F2 :
DL.GetKeyState(VK_F2)=1 Exemple pour la touche de
fonction F2 : DL.GetKeyState(VK_F2)=1 Exemple pour
la touche de fonction F2 : DL.GetKeyState(VK_F2)=1
Le bouton gauche de la souris est enfoncé
Exemple pour la Touche de fonction F2
: DL.GetKeyState(VK_F2)=1 Le bouton gauche de la
souris est enfoncé Exemple pour la
touche Touche de fonction F2
: DL.GetKeyState(VK_F2)=1 Le bouton D est enfoncé
Exemple de touche de fonction F2 :
DL.GetKeyState(VK_F2)=1 Exemple pour la touche de
fonction F2 : DL.GetKeyState(VK_F2)=-128 La touche F2
est enfoncée Exemple pour la touche
Touche de fonction F2 :
DL.GetKeyState(VK_F2)=0 Exemple pour la
touche de fonction F2 : DL.GetKeyState( VK_F2)=0
Exemple pour la touche de fonction F2 :
DL.GetKeyState(VK_F2)=0
```

10.2.2.9 GetReceiveComments

Renvoie une liste chronologique de tous les commentaires d'ordre de réception émis, comme alternative au traitement de Sub DL_OnReceive().

Chaîne de

caractères de

retour

Syntaxe

Résultat **= DL.GetReceiveComments()**

Remarques

result contient tous les Receive Sequence Comments dans l'ordre chronologique, séparés par un saut de ligne, depuis le dernier appel à **GetReceiveComments**. En

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

utilisant les macros Receive Sequence Comment, vous pouvez implémenter un analyseur syntaxique pour toutes les données Receive Sequence entrantes, comme alternative à Sub DL_OnReceive().

REMARQUE : Un maximum de 10000 événements de séquences de réception est enregistré et utilisé par

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

GetReceiveComments, ce qui devrait suffire pour toutes les applications pratiques.

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

10.2.2.10 Champ de saisie2

Alternative à la méthode VBScript InputBox originale.

Affiche une invite de commande dans une boîte de dialogue, attend que l'utilisateur saisisse du texte ou clique sur un bouton et renvoie le contenu de la zone de texte. Ce dialogue apparaît toujours sur le même écran que la fenêtre principale de Docklight Scripting. Elle ne prend pas en charge les arguments optionnels (rarement utiles) *xpos*, *ypos*, *helpfile* et *context* de la méthode VBScript InputBox.

ASTUCE : Vous pouvez également utiliser la fonction DL.GetKeyState pour attendre et réagir aux saisies clavier ou souris.

Chaîne de

caractères de

retour

Syntaxe

Résultat **= DL.InputBox2 (***Invite***[,** *Titre***][,** *Par défaut***])**

Partie	Description
Invitation	Obligatoire. Expression de type chaîne de caractères qui s'affiche comme message dans la boîte de dialogue. La longueur maximale de l'invite de commande est d'environ 1024 caractères, en fonction de la largeur des caractères utilisés. Si l'invite comporte plus d'une ligne, vous pouvez séparer les lignes par un caractère de retour chariot (**Chr(**13**)**), un caractère de saut de ligne (**Chr(**10**)) ou une** combinaison de caractères de retour chariot et de saut de ligne (**Chr(**13**) & Chr(**10**)**) entre chaque ligne.
Titre	Facultatif . Expression de type chaîne qui s'affiche dans la barre de titre de la boîte de dialogue. Si vous omettez title, le nom de l'application s'affiche dans la barre de titre.
Standard	Facultatif . Expression de type chaîne qui s'affiche dans la zone de texte comme réponse par défaut si aucune autre entrée n'est effectuée. Si Si vous omettez default, le champ de texte s'affiche vide.

```
Exemple de fonction DL.InputBox2
MyInput = DL.InputBox2("Veuillez saisir le texte", "Mon titre",
  "Exemple de texte")
DL.AddComment MaSaisie          ' Impression de la saisie de l'utilisateur
```

10.2.2.11 MsgBox2

Alternative à la méthode MSgBox originale de VBScript.

Affiche une fenêtre de message, attend que l'utilisateur clique sur un bouton et renvoie une valeur indiquant quel bouton l'utilisateur a cliqué. Ce dialogue apparaît toujours sur le même écran que la fenêtre principale de Docklight Scripting. Elle ne prend pas en charge les arguments facultatifs *helpfile* et *context* de la méthode VBScript MsgBox.

ASTUCE : Vous pouvez également utiliser la fonction DL.SetUserOutput pour créer une zone de sortie utilisateur supplémentaire et afficher des informations supplémentaires sur l'utilisateur.

Valeur de

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

retour Entier

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

Syntaxe

Résultat **= DL.MsgBox2 (***Invite***[,** *Boutons***][,** *Titre***])**

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

Partie	Description
Invitation	Obligatoire. Expression de type chaîne de caractères qui s'affiche comme message dans la boîte de dialogue. La longueur maximale de l'invite de commande est d'environ 1024 caractères, en fonction de la largeur des caractères utilisés. Si l'invite comporte plus d'une ligne, vous pouvez séparer les lignes par un caractère de retour chariot (**Chr(**13**)**), un caractère de saut de ligne (**Chr(**10**)) ou une** combinaison de caractères de retour chariot et de saut de ligne (**Chr(**13**) & Chr(**10**)**) entre chaque ligne.
Boutons	Les valeurs communes facultatives sont une combinaison (somme) des constantes suivantes : Constante Valeur Description vbOKOnly 0 Bouton OK uniquement (par défaut) vbOKCancel1Boutons OK et Annuler vbAbortRetryIgnore2Boutons Annuler, Répéter et Ignorer vbYesNoCancel 3Boutons, Non et Annuler vbYesNo 4Boutonset Non vbRetryCancel5Boutons Répéter et Annuler vbCritical16Critique Message vbQuestion 32Requête d'alerte vbExclamation 48Avertissement Message vbInformation 64Message d'information vbDefaultButton1 0Le premier bouton est standard (par défaut) vbDefaultButton2256Second bouton est par défaut vbDefaultButton3512Le troisième bouton est préréglé ASTUCE : Vous trouverez une liste complète de toutes les constantes disponibles dans la documentation Microsoft VBA pour MsgBox.
Titre	Facultatif . Expression de type chaîne qui s'affiche dans la barre de titre de la boîte de dialogue. Si vous omettez title, le nom de l'application s'affiche dans la barre de titre.
Résultat	Renvoie l'action de l'utilisateur : Constante Valeur Description vbOK1OK Touche enfoncée vbCancel 2 Touche Annuler enfoncée vbAbort3Touche Annuler appuyé vbRetry 4Touche Retry appuyée vbIgnore5Touche Ignore appuyée vbYes 6Touche Oui enfoncée vbNo 7 Aucune touche enfoncée

```
'Exemple de fonction MsgBox2
result = DL.MsgBox2("Exécuter ce test ?", 3,
"Mon titre") If result = 6 Then
      DL.AddComment "Bouton Oui appuyé"
ElseIf result = 7 Then
      DL.AddComment "Aucune touche n'a été enfoncée".
Sinon
```


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

```
            DL.AddComment "Annulé
        Fin quand
```

10.2.2.12 LoadProgramOptions

Charge les options du programme Docklight à partir d'un fichier créé avec SaveProgramOptions.

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

Valeur de retour

Vide

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

Syntaxe

DL.LoadProgramOptions *filePathName*

La syntaxe de la méthode LoadProgramOptions se compose des parties suivantes :

Partie	Description
filePathName	Obligatoire. Chaîne de caractères contenant le chemin d'accès (répertoire et nom de fichier) du fichier de configuration Docklight à charger. Si aucun répertoire n'est indiqué, Docklight utilise le répertoire de travail actuel. Si *filePathName* est un est vide, une boîte de dialogue de sélection de fichier s'affiche.

Remarques

Pour plus d'informations sur l'enregistrement et le chargement des options du programme Docklight, consultez la méthode SaveProgramOptions.

10.2.2.13 PlaybackLogFile

Ouvre un fichier journal Docklight existant (représentation HEX, décimale ou binaire) et reproduit (retransmet) les données d'une direction de communication de ce fichier journal.

Démarre la communication si elle n'est pas déjà en cours (voir StartCommunication).

Valeur de retour

Vide

Syntaxe

DL.PlaybackLogFile *filePathName* **[,** *dataDirection***] [,** *timeInterval* **]**

La syntaxe de la méthode PlaybackLogFile se compose de ces parties :

Partie	Description
filePathName	Obligatoire. Chaîne de caractères contenant le chemin d'accès (répertoire et nom de fichier) du fichier journal. Si aucun répertoire n'est indiqué, Docklight utilise le répertoire de travail actuel. Si *filePathName* est une chaîne vide, une boîte de dialogue de fichier s'affiche pour sélectionner un fichier.
dataDirection	Valeur de chaîne optionnelle. Indique lequel des deux canaux de communication enregistrés (TX ou RX ? COM1 ou COM2 ?) doit être reproduit. Si *dataDirection* est une chaîne de caractères vide, le premier canal à être lu est celui qui a été sélectionné. apparaît dans le fichier journal est utilisé.
timeInterval	Valeur entière facultative. Utilise un temps de pause en millisecondes entre deux messages au lieu du timing initial du fichier journal (voir remarques ci-dessous).

Remarques

La lecture n'est possible qu'en mode de communication Envoi/Réception et uniquement pour les fichiers journaux en représentation HEX, décimale ou binaire. Pour la lecture, il est possible d'utiliser aussi bien des fichiers HTML (.htm) que des fichiers de texte pur (.txt).

Si *filePathName* n'existe pas, Docklight signale une erreur et l'exécution du script est interrompue.

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

Le fichier journal utilisé doit contenir des horodatages pour les deux sens de communication.

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

filePathName doit contenir l'extension de nom originale dans le style Docklight pour déterminer le type de fichier journal, par exemple "log1_hex.txt", "log1_dec.txt" ou "log1_bin.txt". Si *filePathName* a un autre format, un fichier journal HEX est accepté.

PlaybackLogFile évalue les horodatages du fichier journal et émule le timing de la communication initialement consignée. Si vous souhaitez modifier cela, par exemple pour ralentir les choses à des fins de débogage, vous pouvez utiliser l'argument optionnel *timeInterval*.

Exemple

```
' Exemple PlaybackLogFile

' Lecture de la première direction de données d'un fichier log
d'exemple DL.AddComment "Playback TX side
DL.PlaybackLogFile "modbus_logfile_hex.txt".

' Même fichier, mais maintenant les réponses sont lues depuis
le côté RX DL.AddComment
DL.AddComment
DL.AddComment "Lecture page RX" DL.PlaybackLogFile
"modbus_logfile_hex.txt", "RX

Le même fichier, mais avec un intervalle de temps fixe entre
les différentes séquences.
DL.AddComment
DL.AddComment
DL.AddComment "Lecture TX avec intervalle fixe de 500
millisecondes".
DL.PlaybackLogFile "modbus_logfile_hex.txt", "", 500
```

Nous supposons que le fichier journal modbus_logfile_hex.txt a été créé lors d'une session de communication Modbus précédente et qu'il contient les informations suivantes :

```
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
```

Après l'exécution de l'exemple de script, la fenêtre de communication pourrait se présenter comme suit :

```
  Lecture Côté TX
        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

  Lecture Côté RX
        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

```
Lecture TX à intervalle fixe de 500 millisecondes
```


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

Enregistre les options actuelles du programme Docklight (tout ce qui peut être défini dans la boîte de dialogue Options) et le mode de fenêtre de communication actif (ASCII, HEX, décimal ou binaire) dans un fichier.

Valeur de retour

Vide

Syntaxe

DL.SaveProgramOptions *filePathName*

La syntaxe de la méthode SaveProgramOptions se compose des parties suivantes :

Partie	Description
filePathName	Obligatoire. Chaîne de caractères contenant le chemin d'accès (répertoire et nom de fichier) du fichier de configuration Docklight à créer. Si aucun répertoire n'est indiqué, Docklight utilise le répertoire de travail actuel. Si *filePathName* est un est vide, une boîte de dialogue de sélection de fichier s'affiche.

Remarques

Un fichier créé avec **SaveProgramOptions** peut être chargé avec LoadProgramOptions. **SaveProgramOptions** crée des fichiers XML (extension de fichier .xml).

SaveProgramOptions et **LoadProgramOptions** sont très utiles pour garantir que Docklight utilise certains paramètres d'affichage et d'horodatage pour l'exécution de votre script Docklight. Cela convient parfaitement aux outils de test automatisés destinés à d'autres utilisateurs qui ne sont pas familiers avec Docklight. Vous pouvez préparer l'affichage approprié (par exemple, mode HEX uniquement) et vous assurer que les autres utilisateurs reçoivent la même sortie d'affichage que vous.

REMARQUE : La communication doit être arrêtée (voir StopCommunication) avant de **SaveProgramOptions** ou **LoadProgramOptions**.

Exemple

```
' Exemple SaveProgramOptions
DL.StopCommunication DL.SaveProgramOptions
"myFavoriteSettings" DL.Quit
```

Effectuez maintenant quelques modifications dans les options de Docklight ou modifiez la fenêtre de communication en choisissant par exemple l'onglet Décimal. Exécutez ensuite le script suivant :

```
' Exemple LoadProgramOptions
DL.LoadProgramOptions "myFavoriteSettings
```

Docklight revient alors aux paramètres d'affichage précédents.

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

10.2.2.15 SetChannelSettings

Modifier les paramètres actuels du canal de communication : indiquer un nouveau numéro de port COM ou une nouvelle adresse TCP/IP, ou modifier les paramètres du port série (vitesse de transmission, paramètres de parité, ...).

Les réglages du port série peuvent être modifiés à la volée, alors que le canal de communication est ouvert. Pour d'autres modifications, par exemple le numéro de port COM lui-même, il faut appeler StopCommunication avant d'utiliser **SetChannelSettings**.

REMARQUE :

Valeur de

retour

Booléen

Syntaxe

Résultat **= DL.SetChannelSettings(** *newSettings* **[,** *channelNo***] [,** *dontTest***])**

La syntaxe de la méthode **SetChannelSettings** se compose des parties suivantes :

Partie	Description
newSettings	Requis. Chaîne de caractères indiquant le nouveau canal de communication et/ou les paramètres série. Voir ci-dessous pour les détails
channelNo	Facultatif . Valeur entière indiquant le canal de communication si le mode de communication : surveillance est utilisé. La valeur par défaut est 1 (canal 1).
dontTest	Facultativement, il s'agit d'une option. Valeur booléenne. Si *dontTest* a la valeur True, **SetChannelSettings n'**ouvre ni ne ferme le canal de communication à des fins de test. Voir la section "Remarques" ci-dessous. La valeur par défaut est False (le canal est testé pour déterminer la valeur de retour).

L'argument *newSettings* accepte les valeurs suivantes :

Valeur	Description
"*COMxxx*"	Sélectionnez un nouveau port de communication série, par exemple "COM7".
"*RemoteHost*:*RemotePort*" (*hôte distant)*	Faites de ce canal un client TCP et établissez une connexion avec l'adresse IP et le numéro de port TCP indiqués, par exemple "192.0.0.1:3001" (voir les paramètres du projet).
"SERVER:*LocalPort*"	Faire de ce canal un serveur TCP qui gère les connexions sur le port TCP spécifié, par exemple "SERVER:3001" (voir les paramètres du projet)
"UDP:*RemoteHost*:*Port*"	Transforme ce canal en pair UDP et transfère les données vers *RemoteHost*:*Port* et écoute du *port* local (voir les paramètres du projet)
"USBHID:*vendorId*:*productId"*	Accès aux rapports d'entrée/sortie USB-HID (voir les paramètres du projet).
"PIPE:*myNamedPipe"*	Connexion client à un tube nommé avec lecture/écriture Accès (voir Paramètres du projet).

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

"COMxxx:*BaudRate,Parité, DataBits*,*StopBits*"	Sélectionnez un nouveau port série et des paramètres de communication série *La parité* peut être NONE, EVEN, ODD, MARK, SPACE. Beispiel: "COM18:9600,EVEN,8,1"
"BaudRat,*Parity*,*DataBits*, *StopBits*"	Modifier les paramètres série sans connaître/modifier le port série actuel. Exemple : "38400,AUCUN,8,1".
"BaudRat,*Parity*,*DataBits*, *StopBits,FlowControl ParityErrorChar*"	Syntaxe avancée pour modifier en plus les options de contrôle de flux matériel :

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

	FlowControl peut être OFF, RTSCTS, XONXOFF, RTSSEND *Caractère d'erreur de parité* : Le code ASCII décimal pour le caractère d'erreur de parité (voir les paramètres du projet). La valeur par défaut est 63. Beispiel: "9600,NONE,8,2,RTSCTS,35"
">"	Recherchez le prochain port série COM disponible sur ce PC. Si le port actuellement sélectionné est COM1, **SetChannelSettings** commence la recherche à partir de COM2.

Remarques

Pour la plupart des applications, il n'est pas nécessaire d'utiliser **SetChannelSettings** ou son homologue GetChannelSettings. Les paramètres de communication peuvent être sélectionnés dans la boîte de dialogue Paramètres du projet et enregistrés dans le fichier de projet Docklight (voir Enregistrement et chargement des données de votre projet et la méthode Ouvrir un projet).

La méthode **SetChannelSettings** est destinée aux applications de script Docklight avancées qui nécessitent de contrôler les paramètres des canaux de communication pendant l'exécution du script. Elle permet de créer des scripts qui accèdent à différents ports COM (voir exemple ci-dessous) ou qui parcourent une liste d'adresses IP.

SetChannelSettings génère une erreur si une valeur non autorisée est définie avec *newSettings*.

Si l'argument *newSettings* est valide (et que le drapeau *dontTest* n'est pas activé), le canal de communication est immédiatement ouvert puis fermé pour un test.

Si *dontTest* est True, **SetChannelSettings n'**ouvrira/fera pas le canal pour le tester et renverra toujours True. Ceci est utile pour les applications réseau où des tentatives supplémentaires de connexion/déconnexion pourraient perturber l'autre hôte/périphérique. Des problèmes sont apparus, par exemple, dans les applications de serveur Telnet.

La valeur de retour de **SetChannelSettings** est True si le canal a pu être ouvert avec succès (ou si les nouveaux paramètres sont corrects et *que dontTest* est true).
La valeur de retour est False si les paramètres ne sont pas valides ou si une erreur s'est produite lors de la tentative d'accès au port (par exemple, le port COM est déjà utilisé ou le pare-feu a refusé l'accès TCP/IP).

REMARQUE : La modification du paramètre *FlowControl* lorsque Project Settings : Flow Control a une valeur autre que "Désactivé" peut entraîner un comportement indéfini.

Voir aussi GetChannelSettings et GetChannelStatus.

Exemple

```
' Exemple SetChannelSettings /
GetChannelSettings ' (nécessite un docklight
en mode émission/réception)

DL.ClearCommWindows

DL.AddComment "Rechercher le premier port COM disponible sur ce
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()
    ' Essayer le port COM suivant
    portAvailable = DL.SetChannelSettings(">")
    newPort = DL.GetChannelSettings()
```


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

```
    ' Avez-vous déjà essayé tous les ports COM
    sur ce PC ? If (oldPort = newPort) Then
        DL.AddComment "Pas de port COM
        disponible" DL.Quit
    Fin
Quand
Retourner
DL.AddComment "Utiliser le port COM " & DL.GetChannelSettings()

' Essayez quelques débits en bauds
différents baudRatesStr =
"9600,14400,57600,115200".
baudRatesArray = Split(baudRatesStr, ",")
For i = 0 To UBound(baudRatesArray)
    ' Adapter les réglages du port série
    DL.SetChannelSettings(baudRatesArray(i) + ",NONE,8,1")
    DL.AddComment
    DL.AddComment
    DL.AddComment "Tester avec les réglages " &
DL.GetChannelSettings()
    ' Envoyez une commande de test du modem et attendez un peu la
réponse
    DL.StartCommunication
    DL.SendSequence "", "ATI3" + Chr(13) + Chr(10)
    DL.Pause 200
    DL.StopCommunication
Suivant
```

Une fois le script exécuté sur un ordinateur équipé d'un modem intégré sur COM3, la fenêtre de communication du Docklight pourrait ressembler à ceci :

```
 Rechercher le premier port COM disponible sur ce
 PC... Utilisation du port COM COM3:9600,NONE,8,1


 Test avec les paramètres 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 avec les paramètres

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>
```


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

Test avec réglages

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 avec les paramètres 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

Utilisez un paramètre de **filtre de contenu** différent de celui défini dans la boîte de dialogue Paramètres du projet - Filtre de communication.

Valeur de retour

Vide

Syntaxe

DL.SetContentsFilter *newContentsFilter*

La syntaxe de la méthode **SetContentsFilter** se compose des parties suivantes :

Partie	Description
newContentsFilter	Obligatoire. Valeur entière pour la sélection du nouveau filtre : 0 = afficher toutes les données de communication initiales (canal 1 et canal 2) 1 = Afficher uniquement le canal 1 ou les données [TX]. 2 = Afficher uniquement le canal 2 ou les données [RX]. 3 = Masquer toutes les données sérielles d'origine, afficher uniquement les commentaires supplémentaires

Remarques

Une fois l'exécution du script terminée, le **filtre de contenu** reprend le réglage initial du projet, défini sous Paramètres du projet - Filtre de communication.

Exemple

```
' Nécessite le projet d'exemple de base Docklight "PingPong" et
un loopback sur l'élément sélectionné.
' Canal de communication
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
```

Après l'exécution du script, la fenêtre de communication de Docklight pourrait ressembler à ceci :

```
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
 "Recevoir un "ping
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
 "Obtenir "Pong
7/30/2012 17:42:31.499 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
SetContentsFilter(1)
 "Recevoir un "ping
7/30/2012 17:42:31.523 [TX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Obtenir "Pong
7/30/2012 17:42:31.547 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
 "Recevoir un "ping
7/30/2012 17:42:31.572 [TX] - 6F 2D 2D 2D 2D 20 50 6F 6E 67
 "Obtenir "Pong
7/30/2012 17:42:31.594 [TX] - 2D 2D 2D 2D 6F 20 50 69 6E 67
 "Recevoir un "ping
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 20 50 6F 6E 67
 "Pong" recevoir 2D 2D 2D 6F 20 50 69 6E 67 "Ping" recevoir 6F
2D 2D 2D 20 50 6F 6E 67 "Pong" recevoir 2D 2D 2D 6F
20 50 69 6E 67 "Ping" reçu 6F 2D 2D 2D 20 50 6F 6E 67
 "Pong" obtenu
SetContentsFilter(3)
 "Ping" recevoir "Pong" recevoir "Ping" recevoir "Pong "
recevoir "Ping" recevoir "Pong
```

10.2.2.17 SetHandshakeSignals

Définit les signaux de poignée de main RTS et DTR. Autorisé uniquement si Flow Control : Manuel est utilisé.

Syntaxe

DL.SetHandshakeSignals *rts*,

dtr

La syntaxe de la méthode **SetHandshakeSignals** se compose des parties suivantes :

Partie	Description
rts	Obligatoire. Valeur booléenne pour définir RTS = Haut (True) ou RTS = Bas (False)
dtr	Obligatoire. Valeur booléenne pour définir DTR = High (True) ou DTR = Low (Faux)

Remarques

Voir aussi la fonction GetHandshakeSignals pour la lecture de l'état actuel de l'appareil.

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

état des signaux de Handshake.

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

SetHandshakeSignals peut être utilisé avant l'ouverture du canal de communication afin de garantir un certain état des lignes RTS et DTR lors de l'initialisation.

Exemple

```
' Exemple SetHandshakeSignals
DL.SetHandshakeSignals true, false
DL.StartCommunication
DL.Pause 1000
DL.SetHandshakeSignals false, true
DL.Pause 1000
```

10.2.2.18 SetUserOutput

Créez un onglet supplémentaire pour les sorties utilisateur dans la zone de documentation/script de la fenêtre principale de Docklight. Ajoutez des sorties en texte simple ou en RTF formaté pour permettre un retour visuel de l'utilisateur et/ou une interaction de type menu (avec GetKeyState).

Syntaxe

DL.SetUserOutput *text* **[,** *rtfFormat***] [,** *append***] [,** *readFromFile***]**

La syntaxe de la méthode **SetUserOutput** se compose des parties suivantes :

Partie	Description
Texte	Obligatoire. Votre sortie à afficher ou à ajouter. Si le *texte* est une chaîne de caractères vide, la languette d'édition est à nouveau supprimée de la fenêtre principale de Docklight.
rtfFormat	Valeur booléenne facultative. False (par défaut) = *le texte* est du texte pur True = *Le texte* est un document RTF valide, par exemple le contenu d'un fichier de document RTF.
joindre.	Valeur booléenne facultative. True (par défaut) = ajouter *du texte* au contenu déjà existant. False = remplacer le contenu existant par *du texte*.
readFromFile	Valeur booléenne facultative. False (par défaut) = le *texte* contient le texte proprement dit ou le document RTF True = *le texte* est un fichier et son contenu doit être affiché. REMARQUE : Si un fichier n'existe pas ou ne peut pas être chargé, **SetUserOutput** utilise à la place une chaîne de caractères vide et n'indique pas d'erreur.

Remarques

SetUserOutput peut vous aider à créer une sortie personnalisée ou à proposer des interactions utilisateur de type menu. Un exemple pourrait être la sélection parmi différents tests prédéfinis dans un test de dispositif automatisé.

Voir l'exemple de script **SetUserOutput_Example.pts** dans le dossier ScriptSamples\Extras\SetUserOutput_Example pour les différentes possibilités de chargement / ajout de texte formaté.

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

Exemple

```
DL.SetUserOutput "" ' Effacer la sortie
DL.Pause 1000
```


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

```
DL.SetUserOutput "Un bon vieux 'Hello
World !" DL.Pause 5000
DL.SetUserOutput "... plus de texte ..."
DL.Pause 5000

Vous pouvez créer vos documents RTF à l'aide de Windows
Wordpad et
Ouvrez le fichier .rtf résultant avec Windows Notepad pour
voir le code RTF.
rtfExample =
"{\rtf1\ansi\ansicpg1252\deff0{\fonttbl{\f0\fnil\fcharset0
Harlow Solid Italic;}}{\colortbl
;\red75\green172\fontblue198;}
\cf1\f0\fs32 A Curly Hello!\par}" DL.SetUserOutput
"... un peu plus coloré ..." DL.SetUserOutput
rtfExample, True

DL.Pause 5000
DL.SetUserOutput "La fin. L'onglet de sortie sera bientôt
 supprimé...", False, False
DL.Pause 5000
DL.SetUserOutput ""
```

Exemple 2 (lire et afficher un fichier RTF)

```
DL.SetUserOutput "C:\mytext.rtf", True, False, True
```

10.2.2.19 SetWindowLayout

Contrôle l'apparence de la fenêtre principale de Docklight, de la même manière que le menu **Outils > Réduire/Restaurer...** et les touches de raccourci F12.

Syntaxe

DL.SetWindowLayout [*mise en page* **]**

La syntaxe de la méthode **SetWindowLayout** se compose des parties suivantes :

Partie	Description
Mise en page	Argument facultatif sous forme de chaîne. Une combinaison des lettres suivantes : S - plage de séquence visible D - Zone de document/script visible La valeur par défaut est "SD" - affiche à la fois les listings de séquences et la zone de document/script.

Remarques

SetWindowLayout est utile lorsque vous avez un script/projet prêt à l'emploi et que vous ne voulez pas que l'utilisateur final soit distrait par les détails de votre projet Docklight et les définitions de votre script. En masquant certaines zones de la fenêtre principale de Docklight, vous pouvez mettre l'accent sur la sortie de la fenêtre de communication ou sur un écran SetUserOutput que vous avez créé.

Exemple

```
' Exemple SetWindowLayout
affiche uniquement la fenêtre de communication. Pas de document
```


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

```
/zone de script, pas de zone d'ordre d'envoi/réception
```


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

```
DL.SetWindowLayout ""
DL.Pause 4000
' Afficher la fenêtre de communication et les listes de suivi
d'envoi/réception, mais pas la zone de document/script
DL.SetWindowLayout "S
DL.Pause 4000
' tout afficher (par défaut, identique à <Layout> = "SD")
DL.SetWindowLayout
```

10.2.2.20 ShellRun

Démarre une application externe ou exécute une opération *Windows* Shell.

Valeur de

retour Entier

Syntaxe 1

result **= DL.ShellRun(***operation***,** *file* **[,** *parameters***] [,** *directory***] [,** *showCmd***])**

La méthode ShellRun Syntaxe 1 se compose des parties suivantes :

Partie	Description
Exploitation	Obligatoire. Chaîne de caractères indiquant le type d'action à effectuer, correspondant au paramètre *Windows* ShellExecute lpOperation. Les valeurs habituellement utilisées sont : "Ouvrir" - ouvre un fichier ou un dossier "Imprimer". - imprime un fichier "Modifier" - lance un éditeur et ouvre le fichier pour le modifier Vous trouverez d'autres valeurs et de la documentation dans la documentation Windows ShellExecute.
Fichier	Obligatoire. Chaîne de caractères contenant le chemin d'accès au fichier à exécuter *fonctionnement*.
Paramètres	Facultativement, la valeur. Valeur de la chaîne de caractères. Si le *fichier* est un fichier exécutable, des arguments de ligne de commande peuvent être passés ici.
Annuaire	Facultativement, la valeur. Valeur de la chaîne de caractères. Définit le répertoire de travail pour l'action. Si non utilisé, le répertoire de travail actuel de Docklight est utilisé.
showCmd	Valeur entière facultative. Indique l'apparence de l'application correspondant au paramètre *Windows* ShellExecute nShowCmd. La valeur par défaut est 10 - SW_SHOWDEFAULT d'autres valeurs courantes : 0 - SW_HIDE 1 - SW_SHOWNORMAL 2 - SW_SHOWMINIMIZED 3 - SW_SHOWMAXIMISÉ

Remarques (syntaxe 1)

result contient la valeur de retour de *Windows* ShellExecute.

En cas de succès, le *résultat* est **> 32**.

Voir l'exemple ci-dessous pour une définition des constantes d'erreur pertinentes pour le *résultat*.

Syntaxe 2

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

result **= DL.ShellRun(**"wait<timeout**>", [,** *file***] [,** *parameters***] [,** *directory***] [,** *showCmd***])**

La méthode ShellRun Syntaxe 1 se compose des parties suivantes :

Partie	Description
"wait<timeout**>"**	Au lieu de l'opération comme dans la syntaxe 1, utilisez : "attendre" - lancer une application externe et attendre qu'elle se termine ou "wait<timeout>", par exemple "wait60000" - démarre une application externe et attend qu'elle se termine ou que le délai indiqué en millisecondes s'écoule.
Fichier *Paramètres* *Répertoire* *showCmd*	comme dans la syntaxe 1

Remarques (syntaxe 2) :

La commande "wait<timeout>" démarre et surveille une application externe via Windows ShellExecuteEx, de manière similaire à la commande "open" de la syntaxe 1.

Les valeurs de retour dans la syntaxe 2 sont différentes de celles de la syntaxe 1 :

Résultat	**Description**
0	avec succès
-1	Erreur au démarrage de l'application
-2	Délai d'attente

Exemple

```
' Exemple
ShellRun '
Voici une liste de codes d'erreur, correspondant au fichier
d'en-tête C shellapi.h original de Windows Kit 8.1 :
' codes réguliers WinExec() */
const SE_ERR_FNF = 2          ' fichier non trouvé
const SE_ERR_PNF = 3          ' chemin non trouvé
trouvé const SE_ERR_ACCESSDENIED = 5 ' accès
refusé const SE_ERR_OOM = 8                '
pas de mémoire disponible const
SE_ERR_DLLNOTFOUND = 32
Valeurs d'erreur pour ShellExecute() au-delà des codes
WinExec() normaux
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

' Exemple de syntaxe 2
DL.AddComment "Ouvrir le bloc-notes et attendre la fin de
l'application. Fermer obligatoirement après un délai d'attente
de 60 secondes maximum :" DL.AddComment "DL.AddComment
DL.ShellRun(" & Chr(34) & "wait60000" & Chr(34) & ", " & Chr(34)
& "notepad.exe" &
Chr(34) & ", " & Chr(34) & "test.txt" & Chr(34) & ")"
```


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

```
DL.AddComment "Retour = " & DL.ShellRun("wait60000",
"notepad.exe", "test.txt")
```


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

```
DL.Pause 1000
' Exemples de syntaxe 1
DL.AddComment "Demander à Windows d'ouvrir le même fichier
avec l'application par défaut pour l'édition de fichiers .txt,
continuer immédiatement :" DL.AddComment "DL.ShellRun(" &
Chr(34) & "edit" & Chr(34) & ", " & Chr(34) & "test.txt" &
Chr(34) & ")"
DL.AddComment "Retour = " & DL.ShellRun("edit", "test.txt")
DL.Pause 3000
DL.AddComment "Ouvrir le site web Docklight dans le navigateur
par défaut :"
DL.AddComment "DL.ShellRun(" & Chr(34) & "open" & Chr(34) & ",
" & Chr(34) & "https://docklight.de" & Chr(34) & ")"
DL.AddComment "Retour = " & DL.ShellRun("open",
"https://docklight.de")
DL.Pause 3000
DL.AddComment "Ouvrir le Gestionnaire de périphériques Windows :"
DL.AddComment "DL.ShellRun(" & Chr(34) & "open" & Chr(34) & ",
" & Chr(34) & "devmgmt.msc" & Chr(34) & ")"
DL.AddComment "Retour = " & DL.ShellRun("open", "devmgmt.msc")
DL.Pause 1000
DL.AddComment "terminé
```

10.2.2.21 Fichier de téléchargement

Ouvre un fichier existant et envoie son contenu. Démarre la communication si elle n'est pas déjà en cours (voir StartCommunication).

Valeur de retour

Vide

Syntaxe

DL.UploadFile *filePathName* **[,** *representation***]**

La syntaxe de la méthode UploadFile se compose de ces parties :

Partie	Description
filePathName	Obligatoire. Chaîne de caractères contenant le chemin d'accès (répertoire et nom de fichier) du fichier à envoyer. Si aucun répertoire n'est indiqué, Docklight utilise le répertoire de travail actuel. Si *filePathName* est une chaîne de caractères vide, un fichier une boîte de dialogue permettant de sélectionner un fichier s'affiche.
Représentation	Facultatif . Valeur de chaîne pour *indiquer* le format du fichier *filePathName.* définir les caractères. "A" = ASCII (par défaut) : *filePathName* est un fichier texte qui est envoyé directement, sans autre analyse syntaxique. "H" = HEX : *filePathName* contient des données de séquence HEX, par exemple 5F 54 65 73 74 ... "D" = décimal : *filePathName* contient des données de séquence décimales, par exemple 095 084 101 115 ... "B" = binaire : *filePathName* contient des données de séquence binaires, par exemple 01011111 01010100 ...

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

	"R" = données brutes : *filePathName* est un fichier binaire qui doit être envoyé tel quel.

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

	doit

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

Remarques

Le téléchargement de fichiers n'est possible qu'en mode de communication Envoyer/Réception.

Si *filePathName* n'existe pas, Docklight signale une erreur et l'exécution du script est interrompue.

La représentation ASCII standard "A" permet d'envoyer des fichiers texte sans autre modification. Pour les fichiers binaires non traités qui doivent être envoyés sans modification, utilisez la
Option "R" (données brutes). Elle ne doit pas être confondue avec la représentation "B" (Binary) utilisée par Docklight pour représenter les données uniquement avec des 0 et des 1.

Vous pouvez utiliser la méthode UploadFile pour transférer le contenu d'un fichier journal Docklight. Veuillez vous assurer que votre fichier journal est en mode texte seul (voir Paramètres du fichier journal) et que le fichier ne contient que les données brutes, sans commentaires supplémentaires et sans horodatage (voir Options).

La méthode UploadFile ne prend pas en charge les formats de fichiers de sortie spécifiques au compilateur, comme par exemple "Intel HEX File". Si vous avez des besoins spécifiques, veuillez vous adresser à notre support par e-mail.

REMARQUE : Les données sont envoyées par blocs de 512 octets maximum. 512 octets envoyés. Si vous envoyez manuellement une séquence d'envoi pendant un téléchargement de fichier, la séquence sera envoyée entre l'un de ces blocs et endommagera le transfert de données.

Exemple

```
' Exemple de fichier

à télécharger '

Envoi d'un fichier

texte
DL.UploadFile "helloworld.txt", "A

' Envoyer directement des
données brutes binaires
DL.UploadFile "test.dat", "R

' Analyse et envoi d'un fichier HEX
DL.UploadFile "hexfile.txt", "H
```

10.2.3 Caractéristiques

10.2.3.1 NoOfSendSequences

Renvoie le nombre de séquences d'émission définies dans le projet Docklight actuel.

Valeur de

retour Entier

Syntaxe

Résultat = **DL.NoOfSendSequences**

Remarques

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

La propriété **NoOfSendSequences** est très utile pour créer des structures de boucle qui utilisent toutes les séquences d'envoi disponibles. Voir l'exemple ci-dessous.

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

Exemple

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

```
' Exemple NoOfSendSequences

' Envoyer toutes les séquences d'émission définies avec un
délai de 1 seconde ' entre les différentes séquences
For i = 0 To (DL.NoOfSendSequences - 1)
   DL.SendSequence i
   DL.pause 1000
Suivant
```

10.2.3.2 NoOfReceiveSequences

Renvoie le nombre de séquences de réception définies dans le projet Docklight actuel.

Valeur de

retour Entier

Syntaxe

Résultat =

DL.NoOfReceiveSequences

Remarques

Voir NoOfSendSequences.

10.3 Procédures d'événement OnSend / OnReceive

Docklight Scripting prend en charge deux procédures spéciales qui sont appelées par le moteur de script Docklight avant l'envoi d'une nouvelle séquence d'envoi ou après la détection d'une séquence de réception.

Définition de la procédure	Description
Sub DL_OnSend() *... mon code de script ...* **End Sub**	**DL_OnSend()** est appelé après qu'un nouvel envoi a été déclenché (envoi manuel ou DL.SendSequence). Des fonctions spéciales de manipulation sont disponibles pour lire et modifier les données avant qu'elles ne soient effectivement transmises. Voir Manipulation des données de la séquence d'envoi.
Sub DL_OnReceive() *... mon code de script ...* **End Sub**	**DL_OnReceive()** est appelé après la détection d'une séquence de réception. Pour lire et traiter les données reçues, on dispose de des fonctions de manipulation spéciales sont disponibles. Voir l'évaluation des données de séquence reçues.

Les procédures peuvent être définies à n'importe quel endroit du code de script au niveau du module (pas au sein d'une classe). Vous trouverez un exemple sous Manipulation des données de la séquence d'envoi.

REMARQUE : Les codes **DL_OnSend()** et **DL_OnReceive()** ne sont exécutés que pendant l'exécution du script. L'envoi d'une séquence d'envoi n'exécute pas automatiquement le **code DL_OnSend()** correspondant. Le script doit être lancé manuellement via le menu **Scripting > Run Script**. Toute erreur pendant l'exécution du script arrête le script et empêche de faire d'autres appels de procédure **DL_OnSend()** / **DL_OnReceive**().

REMARQUE : Les événements **DL_OnSend()** et **DL_OnReceive() se trouvent** dans une

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

file d'attente et peuvent être traités ultérieurement. Pour plus d'informations, voir Timing et déroulement du programme.

ASTUCE : Si votre script ne comporte que les procédures **DL_OnSend()** et **DL_OnReceive()** et rien d'autre, utilisez une simple boucle infinie au niveau du module pour éviter que le script ne se termine immédiatement. Voir l'exemple de manipulation des données de la séquence d'envoi.

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

10.3.1 Sub DL_OnSend() - Manipulation des données de la séquence d'envoi

Pour permettre des calculs et des algorithmes supplémentaires (par exemple des sommes de contrôle) sur les données de la séquence d'envoi, la procédure suivante peut être définie dans un script Docklight :

Sub DL_OnSend()
... mon code de script ...
End Sub

Avant d'envoyer une nouvelle séquence d'envoi, la procédure **DL_OnSend()** est appelée par le moteur de script Docklight. Au sein de la procédure **DL_OnSend(), les** fonctions suivantes sont disponibles pour lire et manipuler les données actuelles de la séquence :

Fonction	Description
Résultat = **DL.OnSend_GetSize()**	Renvoie la taille des données envoyées / le nombre de caractères
Résultat = **DL.OnSend_GetName()**	Renvoie le nom de la séquence d'émission à transmettre. S'il s'agit d'une séquence de données définie par l'utilisateur et créée par une commande DL.SendSequence, la valeur de retour est une chaîne de caractères vide (""). -1.
Résultat = **DL.OnSend_GetIndex()**	Renvoie son index dans la liste des séquences d'envoi. S'il s'agit d'une séquence de données définie par l'utilisateur et créée par une commande DL.SendSequence, la valeur de retour est la suivante -1.
Résultat = **DL.OnSend_GetData(** **[***Représentation***])** Syntaxe 2 : *Résultat* = **DL.OnSend_GetData(** **[***représentation***]** **[,** *Début***] [,** *Longueur***])**	Renvoie une chaîne de caractères contenant les données d'envoi réelles. *La représentation* indique le format du *résultat* à : "A". = ASCII (standard), "H" = HEX, "D" = décimal ou "B" = binaire. Les données renvoyées ne contiennent pas de jokers. Toutes les positions des caractères génériques ont déjà été remplacées par des caractères réels. REMARQUE : Si la séquence d'envoi initiale '#' contient des caractères génériques (zéro ou un caractère), la longueur de la séquence **DL.OnSend_GetData()** peut être plus courte que la séquence initiale contenant des caractères génériques. Syntaxe 2 : Renvoie une chaîne de caractères contenant un certain nombre de caractères issus des données reçues. *Début :* Plage : 1 ... **DL.OnSend_GetSize()**, ou -1 = départ au dernier caractère, -2 = départ à l'avant-dernier caractère, ... La valeur par défaut est 1. *length :* nombre de caractères, ou -1 = jusqu'au dernier caractère, -2 = jusqu'à l'avant-dernier caractère, Le La valeur par défaut est -1.

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

DL_OnSend_SetData *newData* **[,** *Représentation***]**	Remplace les données à transmettre par les données indiquées dans la chaîne *newData*. *representation* indique le format de *newData* "A" = ASCII (standard), "H" = HEX, "D" = décimal ou "B" = binaire. Après avoir terminé la procédure **DL_OnSend()** envoie Docklight *newData*, indépendamment de

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

	à quoi ressemblait la séquence d'envoi initiale. La longueur de *newData* peut être différente de la longueur de la séquence d'envoi initiale. REMARQUE : si *newData* est une chaîne de caractères vide, la transmission de la séquence d'envoi initiale est effectivement supprimée.
DL.OnSend_Poke *charNo*, *valeur*	Définit le caractère à la position *charNo* sur *value*. *value* est le nouveau caractère sous forme d'un nombre entier de 0..255. Voir aussi **DL.OnSend_Peek(...)**
Résultat = **DL.OnSend_Peek(** *charNo* **)** Syntaxe 2 : *result* = **DL.OnSend_Peek(** *charNo*, *representation* **)**	Donne un caractère des données d'envoi comme valeur entière de 0..255. *charNo* est la Position dans les données d'envoi. valide Plage pour *charNo* : 1 ... **DL.OnSend_GetSize()**, ou -1 = départ au dernier caractère, -2 = départ à l'avant-dernier caractère, ... Syntaxe 2 : Renvoie une chaîne de caractères au lieu d'un entier. *La représentation* indique le format : "A" = ASCII, "H" = HEX, "D" = décimal ou "B". = Binaire décimal.

Remarques

Les fonctions **DL.OnSend_GetSize()**, **DL.OnSend_Peek(**..**)** et **DL.OnSend_Poke** permettent d'implémenter facilement des calculs de somme de contrôle et d'autres algorithmes. Voir l'exemple ci-dessous.

La procédure **DL_OnSend()** n'est exécutée que pendant l'exécution du script. Pendant l'exécution du code **DL_OnSend()**, aucun autre traitement de communication et aucune mise à jour de l'affichage ne sont effectués. Pour éviter les problèmes de performance et de temps, il est préférable de maintenir le temps d'exécution à un niveau bas. Évitez par exemple les boucles imbriquées et n'effectuez pas de calculs qui prennent beaucoup de temps.

Comment Docklight gère les événements de données d'envoi et la section de code **DL_OnSend()** pour en savoir plus sur la planification et le déroulement du programme.

Exemple

```
' Exemple de code d'événement

DL_OnSend() ' Séquences d'envoi

prédéfinies
' (0) Test : TestX<CR><NUL>

' Boucle infinie pour éviter que le script ne se termine
immédiatement
Je
    DL.Pause 1 ' (la pause réduit la charge du CPU au
ralenti) Boucle

Sub DL_OnSend()
```


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

```
' Somme de contrôle simple : dernier octet de la séquence
```


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

```
' est une somme de contrôle sur tous les octets
précédents, mod 256 seqSize =
DL.OnSend_GetSize()
nous avons besoin d'au moins une séquence de trois octets
```


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

```
    Si seqSize > 2 Alors
        ' insérer un caractère aléatoire à la place du "X"
        après le test DL.OnSend_Poke seqSize - 2, 65 + Rnd * 25
        ' calculer une somme de contrôle simple sur la nouvelle
        séquence chksumHex = DL.CalcChecksum("MOD256",
DL.OnSend_GetData("H"), "H", 1, seqSize -1)
        ' Écrasement du dernier caractère de la séquence
d'envoi avec la valeur de la somme de contrôle actuelle
        DL.OnSend_Poke seqSize, CInt("&h" +
        chkSumHex) ' Utilisation de la fonction Peek
        pour documentation supplémentaire
        DL.AddComment vbCrLf & vbCrLf DL.AddComment
        "Somme de contrôle activée", False, False For i
        = 1 To seqSize - 1
            DL.AddComment " " & DL.OnSend_Peek(i, "H"), False,
  Faux
        Continuer
        DL.AddComment " est " & DL.OnSend_Peek(seqSize, "H") &
"(Hex), " & DL.OnSend_Peek(seqSize, "D") & "(Decimal)"
    End If
End Sub
```

Après avoir lancé le script et envoyé manuellement deux fois la séquence "test", la fenêtre de communication ASCII de Docklight a pu afficher le résultat suivant :

```
  Somme de contrôle sur 54   53 0D est 00(hex), 000(décimal)
          65 73
            74

23.06.2015 11:28:31.695 [TX] - 54 65 73 74 53 0D 00

  Somme de contrôle sur 54   4E 0D est FB(Hex), 251(Décimal)
          65 73
            74

23.06.2015 11:28:32.568 [TX] - 54 65 73 74 4E 0D FB
```

REMARQUE : Le calcul et la validation des sommes de contrôle et l'exemple pour le protocole Modbus décrivent comment les CRC courants et autres sommes de contrôle peuvent être calculés et validés sans le code DL_OnSend() / DL_OnReceive(). Ce traitement est effectué avant que les données de la séquence ne soient transmises à la procédure DL_OnSend(). Cependant, si vous souhaitez modifier vos données de séquence d'envoi avant de les envoyer *et* que vous avez besoin d'une somme de contrôle pour les données modifiées, l'exemple ci-dessus ge est la bonne solution.

10.3.2 Sub DL_OnReceive() - Évaluation des données de séquence reçues

Pour analyser les données de Receive Sequence (par exemple, vérifier les valeurs effectivement reçues pour une plage de jokers) ou pour effectuer des tâches supplémentaires après la réception de la séquence, la procédure suivante peut être définie dans un script Docklight :

Sub DL_OnReceive()
... mon code de script ...
End Sub

Après la détection d'une nouvelle séquence de réception et l'exécution des actions prédéfinies (ajouter un commentaire, envoyer une séquence, ...), la procédure **DL_OnReceive()** est appelée par le Docklight Script Engine. Au sein de la procédure **DL_OnReceive(),** les fonctions suivantes **sont** disponibles pour lire les données de la séquence de réception :

Fonction	Description

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

Résultat = **DL.OnReceive_GetSize()**	Renvoie la taille des données reçues / le nombre de caractères

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

Résultat = **DL.OnReceive_GetName()**	Renvoie le nom de la séquence de réception correspondante.
Résultat = **DL.OnReceive_GetIndex()**	Renvoie l'index dans la liste des séquences de réception
Résultat = **DL.OnReceive_GetData([***représentation***])** Syntaxe 2 : *Résultat* = **DL.OnReceive_GetData([***Représentation***] [,** *Début***] [,** *Longueur***])**	Renvoie une chaîne de caractères contenant les données effectivement reçues. *representation indique* la représentation du *résultat* : "A" = ASCII (standard), "H" = HEX, "D" = décimal ou "B" = binaire. Les données renvoyées ne contiennent pas de caractères génériques. Les caractères effectivement reçus sont renvoyés aux positions des caractères de remplacement. REMARQUE : Si la séquence de réception initiale contient des caractères génériques '#' (zéro ou un caractère), la longueur de la séquence DL.OnReceive _GetData() peut être plus courte que la séquence initiale contenant des caractères génériques. Syntaxe 2 : Renvoie une chaîne de caractères contenant un certain nombre de caractères issus des données reçues. *Début :* Plage : 1 .. **DL.OnReceive_GetSize()**, ou - 1 = départ au dernier caractère, -2 = départ à l'avant-dernier caractère, ... La valeur par défaut est 1. *length :* nombre de caractères, ou -1 = jusqu'au dernier caractère, -2 = jusqu'à l'avant-dernier caractère, Le La valeur par défaut est -1.
Résultat = **DL.OnReceive_GetChannel()**	Renvoie le numéro du canal de communication sur lequel cette séquence a été détectée. En mode de communication "Surveillance", la valeur de retour est 1 ou 2. En mode de communication "Envoi/réception", la valeur de retour est toujours 2 (canal RX).
Résultat = **DL.OnReceive_Peek(** *charNo* **)** Syntaxe 2 : *Résultat* = **DL.OnReceive_Peek(** *charNo*, *représentation* **)**	Renvoie un caractère des données reçues sous forme d'un nombre entier de 0 à 255 *charNo* est la position dans les données reçues. Plage valable pour *charNo* : 1 ... **DL.OnReceive_GetSize()**, ou -1 = départ au dernier caractère, -2 = départ à l'avant-dernier caractère, ... Syntaxe 2 : Renvoie une chaîne de caractères au lieu d'un entier. *La représentation* indique le format : "A" = ASCII, "H" = HEX, "D" = décimal ou "B". = Binaire décimal.

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

myDateTime = **DL.OnReceive_GetDateTime()** *Millisecondes* = **DL.OnReceive_GetMilliseconds()**	Ces fonctions renvoient la date/l'heure Docklight réelle à laquelle cette séquence de réception a été déclenchée. Le résultat est enregistré dans deux types de données VBScript standard distincts : *myDateTime :* valeur de date VBScript contenant la date/l'heure avec une résolution d'une seconde *millisecondes :* Valeur entière avec la information correspondante en millisecondes de 0..999

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

Remarques

La méthode **DL.OnReceive_GetData()** est un bon moyen d'analyser les données réellement reçues si vous utilisez des protocoles ASCII avec des caractères d'imprimerie uniquement. Si vous avez besoin de la valeur HEX ou décimale de caractères individuels, vous pouvez utiliser la fonction **DL.OnReceive_Peek(** .. **)** comme alternative pratique. Voir la procédure d'événement DL_OnSend() pour un exemple correspondant.

La procédure **DL_OnReceive()** n'est exécutée que pendant l'exécution du script. Pendant l'exécution du code **DL_OnReceive()**, aucun autre traitement de communication et aucune mise à jour de l'affichage ne sont effectués. Pour éviter les problèmes de performance et de temps, il est préférable de maintenir le temps d'exécution à un niveau bas. Évitez par exemple les boucles imbriquées et n'effectuez pas de calculs qui prennent beaucoup de temps.

Les procédures **DL_OnReceive()** ne sont pas exécutées pendant qu'une méthode Pause ou WaitForSequence bloque le déroulement du programme. Lorsqu'une séquence de réception est détectée, l'appel DL_OnReceive() est mis en file d'attente et exécuté après le retour de **Pause** (ou **WaitForSequence**). Voir l'exemple 2 ci-dessous pour une solution à ce problème.

Voir aussi Timing et déroulement du programme pour un aperçu de la manière dont Docklight gère les événements de réception de données et exécute la section de code **DL_OnReceive()**.

Exemple

```
' Exemple DL_OnReceive() Code
événement '
' Séquence d'envoi
prédéfinie ' (0) Valeur
d'envoi :
' VALUE=<?><?><CR><LF>
'
' Séquence de réception
prédéfinie ' (0) Valeur
reçue :
' VALUE=<?><?><CR><LF>
'
' Effectuez ce test sur un port COM avec un port de
bouclage ' (TX connecté à RX du même port).

finished = False
DL.ClearCommWindows
Do
    DL.Pause 1 ' (la pause réduit la charge du CPU au
ralenti) Boucle jusqu'à la fin

Sub DL_OnReceive()
    If DL.OnReceive_GetName() = "Valeur reçue" Then
        DL.AddComment "Valeur reçue = " &
DL.OnReceive_GetData("A", 7, -3)
        ' Lire la valeur des données reçues, mais seulement la
partie "valeur" qui change
        myValue = Mid(DL.OnReceive_GetData(), 7, 2)
        ' S'assurer qu'il s'agit d'une valeur
        numérique
        If IsNumeric(myValue) Then
```


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

```
                ' augmenter
                myValue = myValue + 1
                If myValue < 100 Then
                    Si la valeur est toujours inférieure à 100, envoyez-la. newValueStr
à nouveau
                = CStr(myValue)
```


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

```
                DL.SendSequence "Envoyer la valeur", newValueStr
            Else
                DL.AddComment "VALUE=99, s'arrête..." terminé
                = True
            End If
        End If
    End If
End Sub
```

Après le lancement du script et l'envoi manuel d'une séquence "Send Value" avec la valeur de paramètre "95", la fenêtre de communication pourrait se présenter comme suit :

```
7/29/2012 15:43:43.823 [TX] - VALEUR=95

7/29/2012 15:43:43.826 [RX] - VALEUR=95
 Valeur obtenue = 95

7/29/2012 15:43:43.879 [TX] - VALEUR=96

7/29/2012 15:43:43.880 [RX] - VALEUR=96
 Valeur obtenue = 96

7/29/2012 15:43:43.926 [TX] - VALEUR=97

7/29/2012 15:43:43.927 [RX] - VALEUR=97
 Valeur obtenue = 97

7/29/2012 15:43:43.977 [TX] - VALEUR=98

7/29/2012 15:43:43.978 [RX] - VALEUR=98
 Valeur obtenue = 98

7/29/2012 15:43:44.025 [TX] - VALEUR=99

7/29/2012 15:43:44.026 [RX] - VALEUR=99
 Valeur reçue = 99
 VALUE=99, Arrêt...
```

Exemple 2

```
' Exemple d'utilisation de DL_OnReceive() dans un code avec

des instructions de pause ' Séquence d'envoi prédéfinie
' (0) Bonjour :
'
Bonjour<CR><LF>
'
' Séquence de réception
prédéfinie ' (0) Bonjour :
'
Bonjour<CR><LF>
'
' Effectuez ce test sur un port COM avec un port de
bouclage ' (TX connecté à RX du même port).

DL.ClearCommWindows
' Démarrer la communication started =
True
```


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

```
DL.SendSequence "Salut"
```


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

```
' Attendez environ 1 seconde, mais assurez-vous que les
événements DL_OnReceive()
' sont traités entre-temps
pauseWithEvents 1000
' Arrêter l'envoi et attendre que toutes les données reviennent
correctement started = False
DL.Pause 20
' Débit de données ?
DL.AddComment
DL.AddComment "Nombre de séquences 'Hello' reconnues : " &
DL.GetReceiveCounter("Bonjour")

Sub DL_OnReceive()
    Si démarré Alors
        myDate = DL.OnReceive_GetDateTime()
        msec = DL.OnReceive_GetMilliseconds()
        DL.AddComment " Horodatage reçu = " &
DL.GetDocklightTimeStamp(myDate, msec)
        ' Envoyez la même séquence que celle que vous venez de
recevoir
        DL.SendSequence DL.OnReceive_GetIndex()
    End If
End Sub

Sub pauseWithEvents(millisecondes)
    Contrairement à la commande DL.Pause, cette fonction
DL_OnReceive() permet de
    ' Instructions à traiter pendant le temps
    d'attente startTime = Timer While (Timer -
    startTime) < milliseconds / 1000
        Saut de minuit' / Réinitialisation de la minuterie
Prendre en compte les variables
        If Timer < (startTime - 1) Then startTime = startTime -
86400
        DL.Pause 1
    Reto
urner
End Sub
```

Après le lancement du script, Docklight envoie et reçoit la séquence "Hello" pendant environ 1 seconde. Le nombre total de séquences envoyées et reçues dépend des paramètres du port COM (vitesse de transmission), de la vitesse du PC et des paramètres d'affichage de Docklight. La fenêtre de communication pourrait se présenter comme suit :

```
8/1/2012 11:00:41.830 [TX] - Salut<CR><LF

8/1/2012 11:00:41.834 [RX] - Salut<CR><LF
  Horodatage de la réception = 8/1/2012

11:00:41.834 8/1/2012 11:00:41.846 [TX] -

Bonjour<CR><LF>

8/1/2012 11:00:41.849 [RX] - Bonjour<CR><LF>.
  Horodatage de la réception = 8/1/2012

11:00:41.849 8/1/2012 11:00:41.861 [TX] -
```


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

Bonjour<CR><LF>

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

...

```
8/1/2012 11:00:42.825 [TX] - Bonjour<CR><LF>.
```


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

```
8/1/2012 11:00:42.827 [RX] - Salut<CR><LF
  Horodatage reçu = 8/1/2012 11:00:42.827

8/1/2012 11:00:42.839 [TX] - Bonjour<CR><LF>

8/1/2012 11:00:42.841 [RX] - Salut<CR><LF
  Horodatage de réception = 8/1/2012

11:00:42.841 8/1/2012 11:00:42.852 [TX] -

Bonjour<CR><LF>

8/1/2012 11:00:42.855 [RX] - Bonjour<CR><LF>.

Nombre de séquences "Hello" reconnues : 70
```

Exemple 3

```
' Exemple avec Sub DL_OnReceive(), pour attendre TOUTE

séquence trouvée = False
foundName = ""
foundDate =
Maintenant
foundMSec
= 0

Je
     DL.Pause 1 ' (la pause réduit la charge du CPU au ralenti)
Boucle jusqu'à trouvée

DL.AddComment
DL.AddComment "Séquence reçue : " & foundName
DL.AddComment "Date/heure reçue : " &
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 - Déroulement temporel et flux de programmes

Sub DL_OnSend() Timing

Pendant l'exécution d'un script, la procédure événementielle DL_OnSend() est exécutée une fois pour chaque nouvelle séquence d'envoi. Cela vaut aussi bien pour les séquences envoyées en cliquant sur le bouton "Envoyer" que pour les appels DL.SendSequence.

La procédure d'événement **DL_OnSend()** n'est introduite qu'après l'exécution de la ligne de code du script en cours. Entre-temps, les requêtes "d'envoi" sont mises en mémoire tampon.

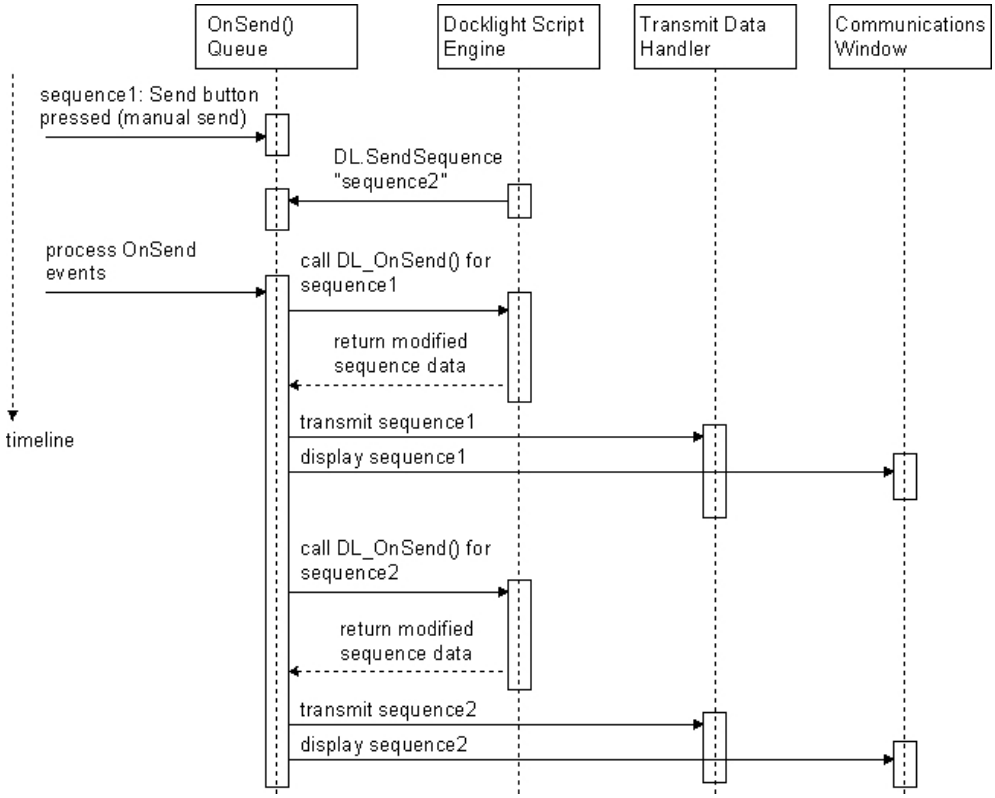
Le diagramme de séquence suivant montre le comportement temporel résultant pour

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

un exemple avec un ordre d'envoi "manuel" (sequence1) et une deuxième séquence d'envoi déclenchée par un code de script (DL.SendSequence "sequence2").

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

REMARQUE : Les champs de remplacement de paramètres et les champs de somme de contrôle sont traités avant que les données de la séquence n'entrent dans la file d'attente OnSend().



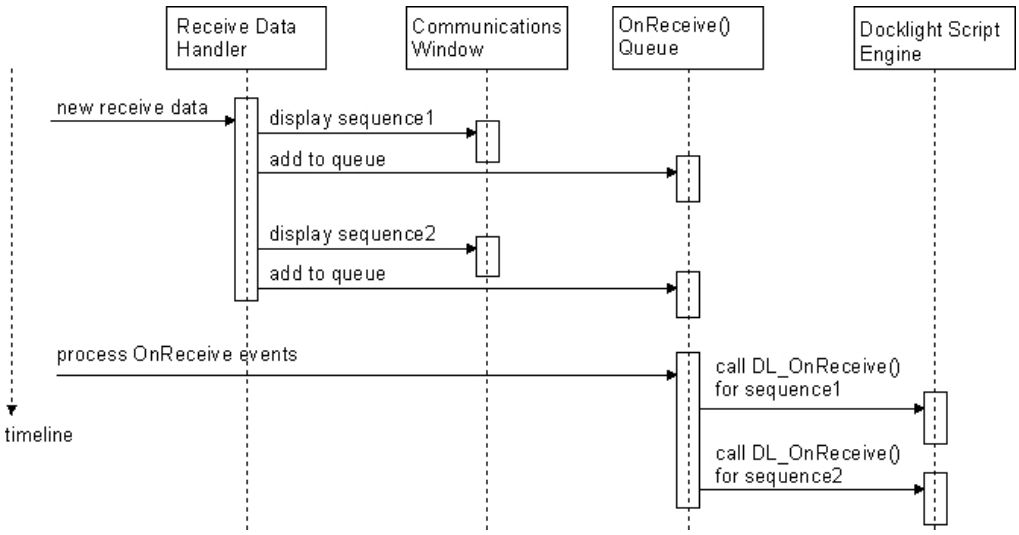
Sub DL_OnReceive() Timing

Comme **DL_OnSend(),** la procédure d'événement DL_OnReceive() n'est pas exécutée immédiatement après que Docklight a détecté une nouvelle correspondance de la séquence de réception. Au lieu de cela, les événements sont mis en mémoire tampon et exécutés après l'exécution de la ligne de code du script en cours.

Le diagramme de séquence ci-dessous montre le timing pour un exemple où deux séquences de réception différentes sont détectées en même temps et où le code **DL_OnReceive()** est exécuté ultérieurement.

REMARQUE : Les champs de somme de contrôle sont traités dans le gestionnaire de données de réception avant que les données de séquence n'entrent dans la file d'attente OnReceive().

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



10.4 Objets FileInput / FileOutput pour la lecture et l'écriture de fichiers

Docklight Scripting propose des objets supplémentaires qui vous permettent de lire un fichier contenant des données textuelles ou binaires ou de créer votre propre fichier de sortie.

Nom de l'objet	Description
FileInput	Ouvrir les fichiers existants pour une saisie séquentielle, le fichier étant lu soit caractère par caractère, soit ligne par ligne. Voir Lire des fichiers.
FileOutput	Créez un nouveau fichier ou ajoutez des données à un fichier existant. Vous pouvez utiliser des données binaires et des des fichiers texte peuvent également être créés. Voir Écrire des fichiers.

10.4.1 FileInput - Lire les fichiers

L'objet global FileInput offre une interface simple pour traiter les fichiers existants, par exemple pour les transférer sur la ligne série avec des sommes de contrôle et un formatage supplémentaires.

Méthodes et propriétés disponibles pour **FileInput** :

Méthode / Propriété	Description
FileInput.OpenFile *filePathName* **[,** *rawData***]**	Ouvre un fichier existant pour la saisie. *rawData* = False (par défaut) : Ouvrir en tant que fichier texte. *rawData* = True : ouverture en tant que fichier binaire de données brutes.
FileInput.CloseFile	Ferme le fichier.
Résultat = **FileInput.GetLine()**	Renvoie une chaîne de caractères avec la ligne de texte suivante. Le *résultat ne* contient pas les caractères de retour à la ligne (CR / LF). La méthode **GetLine** ne peut être utilisée que pour du texte fichiers (*rawData* = False).
Résultat = **FileInput.GetByte()**	Renvoie l'octet suivant.
Résultat = **FileInput.IsOpen**	Renvoie True si un fichier est ouvert, False dans le cas contraire.

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

Résultat = **FileInput.EndOfFile**	Renvoie True si toutes les données ont été lues et si la marque de fin de fichier a été atteinte.

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

result = **FileInput.Dialog([***caption***,] [,** *defaultPath***] [,** *fileFilter***])**	Affiche une boîte de dialogue "Ouvrir un fichier" et renvoie le chemin d'accès au fichier sélectionné ou une chaîne de caractères vide si l'opération est annulée. La valeur par défaut de *fileFilter* est :

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

	Tous les fichiers (*.*)|*.*
Résultat = **FileInput.FileExists**(*filePath*)	Renvoie True si *filePath* existe.

Remarques

Si un fichier ne peut pas être ouvert, **FileInput.OpenFile** génère une erreur et l'exécution du script s'arrête. Utilisez **FileInput.FileExists** pour vérifier si le fichier est disponible.

Voir aussi l'objet FileOutput.

Exemple

```
' FileInput / FileOutput Exemple

DL.ClearCommWindows

' Créer un simple fichier texte
FileOutput.CreateFile "C:\Test.txt"
FileOutput.WriteLine "Hello World !"
FileOutput.WriteLine "Au revoir, le
monde !"
FileOutput.CloseFile

Ouvrez le fichier et imprimez son contenu. path =
FileInput.Dialog("Ouvrir un texte pour Docklight", "C:\",
"Fichiers texte (*.txt)|*.txt")
Si Len(chemin) > 0 Alors
    DL.AddComment "Lire le fichier
    texte..." FileInput.OpenFile Chemin
    Do Until FileInput.EndOfFile
        DL.AddComment FileInput.GetLine()
    Boucle
    FileInput.CloseFile
Fin quand

' Essayez maintenant de créer un fichier
de données brutes FileOutput.CreateFile
"C:\test.bin", True For i = 0 To 255
    FileOutput.WriteByte i
Prochain
FileOutput.CloseFile

' Et le télécharger...
DL.AddComment
DL.AddComment "Lire le fichier de
données brutes..." FileInput.OpenFile
"C:\Ntest.bin", True Do Until
FileInput.EndOfFile
    DL.AddComment Right("0" + Hex(FileInput.GetByte()), 2) + "
", False, False
Boucle
FileInput.CloseFile
```

Le code de script ci-dessus génère la sortie suivante dans la fenêtre de communication du Docklight :

```
 Créer un fichier texte
 C:\test.txt Lire un fichier
```


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

```
texte : C:\test.txt Bonjour le
monde !
```


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

```
Au revoir, le monde !
```


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

```
Créer un fichier de données

binaires C:\\test.txt Lire le
fichier de données brutes.
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 FR 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 - Écrire des fichiers

L'objet global FileOutput offre une interface simple pour la création de fichiers, par exemple pour l'écriture de formats de données définis par l'utilisateur.

Méthodes et propriétés disponibles pour **FileOutput** :

Méthode / Propriété	Description
FileOutput.CreateFile *filePathName* **[,** *rawData***] [,** *appendData***]**	Créer un nouveau fichier de sortie. *rawData* = False (valeur par défaut) : Ouvrir comme fichier texte. *rawData* = True : Ouvrir comme fichier binaire de données brutes. *appendData* = False (valeur par défaut) : Ecrase le fichier s'il existe. *appendData* = True : ajouter des données à une base de données existante. Joindre un fichier.
FileOutput.CloseFile	Ferme le fichier.
FileOutput.WriteLine *Données* **[,** *appendLineBreak***]**	Ecriture des *données de la chaîne de caractères* dans le fichier. *appendLineBreak* = True (par défaut) : ajouter un saut de ligne CR/LF après la chaîne de données *appendLineBreak* = False : ne pas créer de saut de ligne La méthode **WriteLine** ne peut être utilisée que pour du texte fichiers (*rawData* = False).
FileOutput.WriteByte *données*	Écrivez les *données en octets* dans le fichier.
Résultat = **FileOutput.IsOpen**	Renvoie True si un fichier est ouvert, False dans le cas contraire.
result = **FileOutput.Dialog([**caption**]** **[,** *defaultPath***] [,** *fileFilter***])**	Affiche une boîte de dialogue "Enregistrer le fichier" et renvoie le chemin d'accès au fichier sélectionné ou une chaîne de caractères vide si l'opération est annulée. La valeur par défaut de *fileFilter* est : `Tous les fichiers (*.*)|*.*`
Résultat = **FileOutput.FileExists(***filePath***)**	Renvoie True si *filePath* existe.

Remarques

Si **FileInput.CreateFile** ne peut pas ouvrir le fichier, une erreur est générée et

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

l'exécution du script est suspendue.

Voir l'objet FileInput pour un exemple.

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

10.4.3 Plusieurs fichiers d'entrée / plusieurs fichiers de sortie

Si vous devez lire ou écrire plus d'un fichier à la fois, vous pouvez ouvrir jusqu'à 4 fichiers d'entrée et 4 fichiers de sortie en même temps en utilisant d'autres objets globaux en plus de **FileInput** et **FileOutput**. La liste des objets disponibles est la suivante :

Nom de l'objet	Description
FichierEingabe **FichierEingabe2** **FichierEingabe3** **FichierEingabe4**	Ouvrez jusqu'à 4 fichiers différents pour les lire. Voir Lire les fichiers.
Sortie de fichiers ng **Sortie de fichiers ng2** **Sortie de fichiers ng3** **FichierSortie4**	Ouvrez jusqu'à 4 fichiers différents pour écrire. Voir Écrire des fichiers.

Exemple

```
' Sortie de

plusieurs fichiers '

Création de 4

fichiers texte
DL.AddComment "Écrire 4 fichiers texte en même
temps..." FileOutput.CreateFile "file1.txt"
FileOutput2.CreateFile "file2.txt"
FileOutput3.CreateFile "file3.txt"
FileOutput4.CreateFile "file4.txt"
' Écrire en même temps
For i = 1 To 10
    FileOutput.WriteLine "Fichier 1 : ligne de texte " &
    CStr(i) FileOutput2.WriteLine "Fichier 2 : ligne de texte "
    & CStr(i) FileOutput3.WriteLine "Fichier 3 : ligne de texte
    " & CStr(i) FileOutput4.WriteLine "Fichier 4 : ligne de
    texte " & CStr(i)
Continuer
' Fermer les 4
fichiers
FileOutput.CloseFile
FileOutput2.CloseFile
FileOutput3.CloseFile
FileOutput4.CloseFile
DL.AddComment "Done
!"
```

10.5 Canaux latéraux - utilisation de plusieurs connexions de données

Docklight Scripting offre une série de méthodes de script avancées pour des applications multicanales simples. Il vous permet de créer jusqu'à 8 connexions de données secondaires pour l'envoi et la réception de données, en plus des canaux de

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

communication principaux de Docklight.

Les connexions de données supplémentaires / secondaires sont appelées canaux latéraux. Ils sont contrôlés par les méthodes OpenSideChannel / CloseSideChannel. Pour

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

de données sur un canal latéral, la méthode DirectSend est utilisée.

10.5.1 OpenSideChannel / CloseSideChannel - gestion de plusieurs canaux

Les méthodes **OpenSideChannel / CloseSideChannel** permettent d'utiliser plusieurs connexions de données supplémentaires dans une instance de Docklight Scripting. Les données entrantes des canaux latéraux peuvent être différenciées / marquées avec l'argument *rxChannelTag* L'envoi de données sur les canaux latéraux est possible via la méthode DirectSend.

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

DL Méthodes de gestion des canaux latéraux/multi-canaux :

Méthode	Description
Résultat = **DL.OpenSideChannel(** *newSettings* **[,** *channelNo***] [,** *rxChannelTag***])**	Ouvrez un canal de communication de page sous Utilisation de la Paramètres du canal de communication désactivés *newSettings*. *channelNo* = 3 (standard) ou 4-10 *rxChannelTag* = "Canal3" (valeur par défaut) : Les données de réception de ce canal sont identifiées comme suit : `<Canal3>`*... Données sur ce canal latéral recevoir...* `</Canal3>` *rxChannelTag* = "norx" : Les données de communication reçues ne sont pas du tout affichées. *rxChannelTag* = "" (chaîne de caractères vide) : Aucune balise de canal n'est utilisée et vous ne pouvez pas distinguer les données entrantes des données Docklight RX "normales". *result* = True : le canal a été ouvert avec succès. *result* = false : Le canal n'a pas pu être ouvert, par exemple parce que les paramètres ne sont pas valides ou parce que le canal n'a pas été ouvert. le port COM n'est pas disponible.
DL.CloseSideChannel [*channelNo***] (fermer le canal)**	Ferme le canal latéral.

Remarques

Les canaux latéraux dépendent de l'état de la connexion principale et inversement :
OpenSideChannel exécute automatiquement une StartCommunication si nécessaire.
CloseSideChannel ne ferme que le canal latéral indiqué. Les autres canaux de communication poursuivent la transmission des données.
StopCommunication ferme les canaux de communication principaux et tous les canaux secondaires ouverts.
Voir aussi StartCommunication / StopCommunication.

Exemple

```
DL.SetChannelSettings "LOCALHOST:10001"
DL.StartCommunication
DL.OpenSideChannel "SERVER:10001"
DL.ResetReceiveCounter
DL.SendSequence "", "Test", "A"
DL.Pause 1000
DL.AddComment "Arrêter le canal3. Cela devrait provoquer
une erreur de connexion du client TCP..."
DL.CloseSideChannel
DL.Pause 4000
' fermer tous les canaux
DL.StopCommunication
```

La sortie de la fenêtre de communication pourrait se présenter comme suit :

```
02.10.2019 12:44:23.622 [TX] - Test
02/10/2019 12:44:23.634 [RX] - <Channel3>Test</Channel3> [Channel3]

Arrêter le Channel3. Cela devrait déclencher une erreur de client TCP.
```


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

Provoquer des erreurs de connexion...

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

02.10.2019 12:44:24.179 DOCKLIGHT signale : Erreur sur le canal LOCALHOST:10001 :
Connexion TCP/IP fermée par l'ordinateur distant

10.5.2 DirectSend

La méthode **DirectSend** est une alternative à SendSequence pour certaines applications. Sa syntaxe est similaire à celle de SendSequence 2 et est utilisée pour

Envoi de données via des canaux secondaires - connexions de données secondaires ouvertes avec OpenSideChannel.

Transmission/injection de données supplémentaires en contournant la file d'attente des données OnSend et sans ajouter la sortie dans la fenêtre de communication.

Voir **les remarques** et **l'exemple 2** ci-dessous pour un exemple pratique.

Syntaxe

Résultat **= DL.DirectSend** *channelNo*, *customSequence* **[**, *representation* **]**

La syntaxe de la méthode DirectSend se compose de ces parties :

Partie	Description
channelNo	*channelNo* = 3 - 10 : Envoyer des données sur un canal latéral (voir OpenSideChannel). *channelNo* = 1 : envoi sur le canal 1 du Docklight (envoi/réception ou mode de surveillance) *channelNo* = 2 : Envoi sur le canal 2 de la Docklight (uniquement en mode surveillance)
Séquences personnalisées	Obligatoire. Chaîne de caractères contenant la séquence à envoyer. Par défaut, la séquence est transmise en représentation ASCII. Pour les données de séquence HEX, décimales ou binaires, utilisez l'argument de représentation optionnel décrites ci-dessous.
Représentation	Facultative. Valeur de chaîne pour définir le format pour *customSequence*. "A" = ASCII (standard), "H" = HEX, "D" = décimal ou "B" = binaire.

Remarques

Le résultat est true si *channelNo* est valide et si les données ont pu être transmises. La

principale application de **DirectSend** est la combinaison avec OpenSideChannel. **En outre, DirectSend** peut être utile lorsque vous surveillez une connexion de données et que vous devez injecter des données supplémentaires dans le flux de données entre les deux appareils. Vous pouvez effectivement transformer l'approche de "surveillance passive" de Docklight en une "surveillance active", dans laquelle Docklight peut par exemple générer des états d'erreur supplémentaires qui n'apparaissent pas dans la communication initiale.

REMARQUE : **DirectSend ne** génère aucune sortie dans la fenêtre de communication et n'utilise pas la file d'attente des données OnSend. Il transmet simplement vos données textuelles ou binaires "telles quelles".

Exemple 1

```
' Condition préalable : Docklight Mode de communication = émission/réception
```


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

```
DL.StopCommunication
' Utiliser un retour UDP pour le canal 1 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
' et attendre les réactions
DL.Pause 100
```

La sortie de la fenêtre de communication pourrait se présenter comme suit :

```
02.10.2019 11:51:26.749 [RX] - Données TX sur le canal 1<CR><LF
<canal3>Données TX sur le canal latéral<CR><LF
</Canal3>
```

REMARQUE : Les données apparaissent uniquement sur l'affichage des données RX. Aucune sortie de communication TX n'est générée.

Exemple 2

```
' Condition préalable : Docklight Mode de communication =
surveillance DL.StopCommunication
DL.SetChannelSettings "UDP:LOCALHOST:10001", 1
DL.SetChannelSettings "UDP:LOCALHOST:10002", 2
DL.StartCommunication
DL.DirectSend 2, "Bounce"
(rebond)
' et laissez cela continuer un moment
DL.Pause 1000
```

La sortie de la fenêtre de communication pourrait se présenter comme suit :

```
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
```

REMARQUE : Les données sont reflétées de manière répétée entre le canal 1 et le canal 2, car nous utilisons des loopbacks UDP des deux côtés et le mode de surveillance Docklight utilise le transfert de données par défaut.

10.6 Objet de débogage/débogage de script

Docklight Scripting offre des fonctions de débogage supplémentaires via l'objet Debug.

Méthode / Propriété	Description
Debug.Mode = *newValue*	Définit le mode de débogage des scripts : *newValue* = 0 : Pas de débogage, toutes les méthodes de débogage sont ignorées. *newValue* = 1 : Mode de débogage. Les méthodes de débogage décrites ci-dessous sont exécutées.
Debug.Assert *assertCondition*	Interrompt l'exécution du script si *assertCondition* est False. L'exécution du script peut être poursuivie manuellement avec la barre d'outils **Reprendre le script**.
Debug.Break	Interrompt l'exécution du script sans condition.
Debug.PrintMsg *debugMsg*	Ajoute un texte de débogage supplémentaire à l'affichage de la fenêtre de communication, y compris un horodatage et les

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

	ligne actuelle du code du script.

Remarques

Les méthodes **PrintMsg** et **Assert** sont très utiles pour imprimer et surveiller les valeurs des variables à différents points de l'exécution.

Pour que les méthodes de débogage aient un effet, vous devez d'abord activer le mode de débogage en définissant la propriété **Mode à** un : `Debug.Mode = 1`

Exemple

```
' Exemple d'objet de débogage

Debug.Mode = 1

Count = 0
Do
    Nombre = Nombre + 1
    ' afficher quelques informations de débogage : la valeur de
la variable count
    Debug.PrintMsg "Nombre = " & Nombre
    ' Interrompre l'exécution du script lorsque
    le nombre 5 est atteint Debug.Assert (Count
    <> 5)
Boucle jusqu'à nombre = 10

' maintenant la même chose avec le mode de débogage 'off' -
les méthodes de débogage n'ont aucun effet

Debug.Mode = 0
Debug.PrintMsg "ceci ne sera jamais
imprimé" Debug.Break ' ceci ne sera
jamais exécuté

DL.AddComment "Fin du test de débogage".
```

Après l'exécution de ce script, la fenêtre de communication pourrait se présenter comme suit :

```
07.04.2009 15:45:06.078 Ligne #9 Débogag Nombre = 1
                                 e :
07.04.2009 15:45:06.100 Ligne #9 Débogag Nombre = 2
                                 e :
07.04.2009 15:45:06.119 Ligne #9 Débogag Nombre = 3
                                 e :
07.04.2009 15:45:06.131 Ligne #9 Débogag Nombre = 4
                                 e :
07.04.2009 15:45:06.145 Ligne #9 Débogag Nombre = 5
                                 e :

07/04/2009 15:45:06.158 Ligne #11 Debug : Assert est faux
```

(ici, l'utilisateur **continue** manuellement en cliquant sur le bouton **Continuer le script**)

```
07.04.2009 15:45:07.781 Ligne #9 Débogag Nombre = 6
                                 e :
07.04.2009 15:45:07.805 Ligne #9 Débogag Nombre = 7
                                 e :
```


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

```
07.04.2009 15:45:07.830 Ligne #9 Débogage : Nombre = 8
07.04.2009 15:45:07.853 Ligne #9 Débogage : Nombre = 9
```


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

```
07/04/2009 15:45:07.881 Ligne #9 Debug : Nombre = 10
 Test de débogage terminé
```

10.7 directive #include

Indique au préprocesseur du script Docklight d'insérer le contenu du fichier spécifié à l'endroit où apparaît la directive #include.

Syntaxe

#include *filePathName*

La syntaxe #include se compose de ces parties :

Partie	Description
filePathName	Obligatoire. Chaîne de caractères contenant le chemin d'accès (répertoire et nom de fichier) du fichier script Docklight à inclure (fichier .pts). L'extension de fichier .pts peut être omise. Si aucun répertoire n'est spécifié, Docklight utilise le répertoire de travail actuel Répertoire.

Remarques

Si *filePathName n'*est pas un fichier de script Docklight valide ou n'existe pas, Docklight signale une erreur et le script ne démarre pas.

La directive #include indique au préprocesseur de traiter le contenu d'un fichier spécifié comme si ce contenu était inclus dans le programme source à l'endroit où la directive apparaît.

Vous pouvez organiser les déclarations de constantes et les définitions de fonctions dans des fichiers include, puis utiliser les directives #include pour ajouter ces définitions à chaque script. Les fichiers include sont également utiles pour inclure des déclarations de variables externes et de types de données complexes.

Exemple

```
Exemple d'instruction
#include '.
#include "myIncludeFile.pts"
DL.AddComment " Pi = " & conPi
```

Le **fichier myIncludeFile.pts** contient la définition suivante :

```
Const conPi = 3.14159265358979
```

La sortie de la fenêtre de communication se présenterait alors comme suit :

```
  Pi = 3,14159265358979
```

10.8 Syntaxe de la ligne de commande

L'application Docklight Scripting prend en charge les arguments de ligne de commande pour charger (et exécuter) des fichiers de projet ou de script prédéfinis. Utilisez la syntaxe de commande suivante :

Docklight_Scripting.exe [-r] [-m] [-i] [*NomducheminProjet.ptp* **] [** *NomducheminScript.pts* **]**

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

La ligne de commande Docklight se compose de ces parties :

Partie	Description
-r	Argument facultatif utilisé en combinaison avec *scriptPathName.pts.* Exécute le script immédiatement. Si aucune erreur d'exécution ou d'arrêt de l'utilisateur ne se produit, l'application Docklight Scripting fermé après la fin de l'exécution du script.
-m	Argument facultatif. Réduit la fenêtre de l'application Docklight- Scripting au démarrage.
-i	Argument facultatif. Opération invisible / pas de fenêtre principale. Utile en combinaison avec l'option **-r** et *scriptPathName.pts.*
Nom du chemin du projet.ptp	Facultatif . Charge le fichier de projet Docklight *projectPathName.ptp*
NomducheminScript.pts	Facultatif . Charge le fichier de script Docklight *scriptPfadName.pts*

Remarques

Si votre script utilise l'interface StartLogging ou FileInput / FileOutput et que vous ne fournissez qu'un nom de fichier, mais pas de chemin d'accès complet au répertoire en tant que paramètre, Docklight Scripting utilise le répertoire actuel du script/projet.

Exemple

```
Docklight_Scripting.exe -r C:\myScript.pts
```

Charge le fichier de script Docklight **C:\myScript.pts** et l'exécute.

10.9 Dialogue de la page : Personnaliser / Éditeur externe

Menu Scripting > Personnaliser / Éditeur externe Utiliser un programme externe comme éditeur de script Docklight

Activez cette option pour désactiver l'éditeur de script intégré et lancer une application d'édition externe à cette fin.

Une syntaxe de configuration flexible vous permet de travailler avec presque tous les éditeurs qui permettent au moins d'ouvrir un fichier via une ligne de commande comme `monEditor.exe tempScriptFile.vbs`

Contrôle de l'application

Ce fichier de configuration détermine la manière dont Docklight Scripting contrôle l'éditeur externe.

Charger le préréglage pour...

Fichiers de configuration prédéfinis pour trois éditeurs largement utilisés.

CONSEIL : Nous recommandons l'éditeur Notepad++, que vous trouverez à l'adresse https://notepad-plus- plus.org/. L'exemple du Notepad de Windows n'est donné qu'à titre d'illustration et explique le fonctionnement des fichiers de configuration. Vous pouvez l'utiliser comme point de départ pour intégrer votre propre éditeur.

Comment intégrer votre propre éditeur favori

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

Vous pouvez définir le chemin de l'application au début du fichier de configuration en utilisant l'option

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

syntaxe path=. Exemple de ligne :

```
path=C:\NFichiers de programme\NNotepad++
```


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

Toutes les lignes suivantes du fichier de configuration ont la syntaxe suivante :
<Action d'édition> <Contrôle d'application>

Exemple de ligne :
`Ouvrir : notepad.exe "%FICHIER%"`

<Edit Action> peut être l'une des actions d'édition du Docklight suivantes :

Modifier une action	Description
`ouvert :`	Ouvrez un nouveau fichier de code de script
`goto :`	Saut à un numéro de ligne dans le fichier de script
`économiser :`	Enregistrer le fichier actuellement ouvert
`fermer :`	Fermer le fichier actuellement ouvert

<Contrôle de l'application> peut être l'une des opérations suivantes :

Contrôle de l'application	Description
`Clé d'émission`	Envoie une ou plusieurs frappes de touches à l'éditeur externe. Elle utilise la même syntaxe d'arguments que la méthode SendKeys de l'hôte de script Windows. Pour plus de détails, voir la documentation Microsoft correspondante. Exemple : `goto : sendkeys +^{HOME}{DOWN %LINE%}+{UP}`
`endtask`	Quittez l'application externe. Exemple : `fermer : endtask`
`activer .`	Activez la fenêtre de l'application externe. Exemple : `goto : activer`
`dormir`	Attendez jusqu'à 500 millisecondes pour donner à l'application externe un peu plus de temps pour clarifier les choses. Cela peut être nécessaire si vous travaillez avec l'opération `sendkeys :` décrite ci-dessus. Exemple : `ouvrir : dormir 100`
Ligne de commande	Outre les opérations mentionnées ci-dessus, vous pouvez exécuter n'importe quelle ligne de commande Windows, par exemple pour lancer votre éditeur externe. Exemple : `Ouvrir : notepad++.exe -nosession -lvb -n%LINE% "%FICHIER %"`

Pour chaque *<Edit Action>*, vous pouvez définir plusieurs lignes de commande, par exemple
`goto : sendkeys +^{HOME}{DOWN %LINE%}+{UP}`
`goto : activate`

Les caractères de remplacement suivants sont disponibles pour *<Contrôle* d'application>.

Espace réservé	Description
`%FILE%`	Chemin d'accès à un fichier temporaire contenant le code du script à traiter. Docklight Scripting crée et gère le fichier temporaire.
`%FILE_UNIX %`	Comme `%FILE%`, mais utilise un '/' de style UNIX comme séparateur de chemin. Ceci est utile pour certains paquets d'éditeurs open source qui ont des problèmes avec le backslash ("\") de Windows comme séparateur.

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

%FILE_ESC%	Comme %FILE%, mais avec une double barre oblique inverse ('séquence d'échappement') pour le séparateur de chemin. C'est par exemple le cas lorsque l'on travaille avec l'outil libre Éditeur de code source SciTE nécessaire.

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

%LINE%	Le numéro de ligne actuel du code source. Il est utilisé pour la `goto` est utilisé : Action.

Remarques

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

L'External Editor Support est une solution flexible et ouverte pour nos utilisateurs qui travaillent avec de grands projets de script et qui préfèrent travailler avec un paquet d'édition complet.

L'interface de contrôle des applications décrite ci-dessus vous offre une certaine flexibilité, mais nous sommes conscients des limites du contrôle des applications tierces, qui ne sont pas réellement conçues pour être contrôlées de l'extérieur.

Si vous trouvez un fichier de configuration intelligent pour votre éditeur personnel préféré ou si vous avez des problèmes avec l'interface susmentionnée, notre service clientèle serait heureux d'en entendre parler.

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

Soutien à

11 Soutien

11.1 Assistance et dépannage sur le web

Vous trouverez les dernières FAQ et des informations sur le dépannage sur nos pages d'assistance en ligne, disponibles à l'adresse suivante

www.docklight.de/support/

Vous trouverez des nouvelles de Docklight et des informations sur les mises à jour de

maintenance gratuites sur : www.docklight.de/news.htm

11.2 Prise en charge du courrier électronique

Nous offrons à nos clients enregistrés une assistance individuelle par e-mail. Veuillez mentionner votre numéro de clé de licence Docklight dans votre demande. Nous vous contacterons dans les plus brefs délais afin de trouver une solution à votre problème. Envoyez votre demande d'assistance à

support@docklight.de

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

Annexe

12 Annexe

12.1 Tables de caractères ASCII

Caractère de commande

Caractère	Dez	Hex	ASCII.	Signification
0	00	NUL		Zéro
1	01	SOH		Début du titre
2	02	STX		Début du texte
3	03	ETX		Pause/fin du texte
4	04	EOT		Fin de la transmission
5	05	ENQ		Demande
6	06	ACK		Réaction positive
7	07	BEL		Cloche
8	08	BS		Bouton de retour
9	09	HT		Onglet horizontal
10	0A	LF		Saut de ligne
11	0B	VT		Onglet vertical
12	0C	FF		Avance du formulaire
13	0D	CR		Rapatriement de la voiture
14	0E	SO		Pousser vers l'extérieur
15	0F	SI		Shift in/XON (continuer la sortie)
16	10	DLE		Erreur de transmission des données
17	11	DC1		XON - Caractère de commande d'appareil 1
18	12	DC2		Marque de contrôle des appareils 2
19	13	DC3		XOFF - Caractère de commande d'appareil 3
20	14	DC4		Caractère de commande d'appareils 4
21	15	ENA		Acquittement négatif
22	16	SYN		Ralenti synchrone
23	17	ETB		Fin du bloc de transmission
24	18	CAN		Annuler
25	19	EM		Fin du support
26	1A	SUB		Remplacement/fin du fichier
27	1B	ESC		Fuite
28	1C	FS		Séparateur de fichiers
29	1D	GS		Indicateur de groupe
30	1E	RS		Séparateur d'enregistrement
31	1F	US		Séparateur d'unités

Imprimer les caractères

Caractère	Dez	Hex	ASCII.	Signification
32	20			Tr mondi al aum
33	21	!		!
34	22	"		"
35	23	#		#
36	24	$		$
37	25	%		%

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

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	Zéro
49	31	1	Une
50	32	2	Deux
51	33	3	Trois
52	34	4	Quatre
53	35	5	Cinq
54	36	6	Six
55	37	7	Tamisn
56	38	8	Huit
57	39	9	Neuf
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	Efface r n

12.2 Raccourcis clavier

Raccourcis clavier généraux

S'applique à
Fenêtre de communication (ASCII, HEX, décimal, binaire)
- Boîte de dialogue "Modifier la séquence d'envoi" / "Modifier la séquence de réception

- Domaine de la documentation

Fonction	Combinaison de touches
Aide contextuelle	F1
Coupe	Ctrl+X
Copier	Ctrl+C
Colle	Ctrl+V
Supprimer	Del
Sélectionner tout	Ctrl+A

Touches de raccourci spécifiques au contexte

Menu Docklight

Menu	Fonction	Combinaison de touches

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

Fichier	Nouveau projet	`Ctrl+N`

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

Fichier	Projet ouvert	Ctrl+O
Fichier	Enregistrer le projet	Ctrl+S
Fichier	Communication imprimée	Ctrl+P
éditer	Rechercher une séquence dans Comm.Window	Ctrl+F
Laisse r courir	Démarrer la communication	F5
Laisse r courir	Arrêter la communication	F6
Outil e	Démarrage Comm. Consignation	F2
Outil e	Arrêt Comm. Consignation	F3
Outil e	Clavier Console On	Ctrl+F5
Outil e	Clavier Console Désactivé	Ctrl+F6
Outil e	Documentation/réduire/récupérer la zone de script	F12
Outil e	Réduction/restauration des listings de séquences	Maj+F 12
Scripting	Exécuter le script	Maj+F5
Scripting	Arrêter le script	Maj+F6
Scripting	Pause Script	Maj+F7
Scripting	Continuer le script	Maj+F8
Scripting	Enregistrer le script	Ctrl+T

Fenêtre Communication

Fonction	Combinaison de touches
Trouver une séquence	Ctrl+F
Effacer toutes les fenêtres de communication	Ctrl+W
Commutation entre les représentations ASCII, HEX, décimales et binaires	Ctrl+Tab

Liste des séquences d'émission / de réception

Fonction	Combinaison de touches
Supprimer cette séquence	Del
Modifier cette séquence	Ctrl+E
Envoyer cette séquence *- Séquences d'émission uniquement Liste -*	Espace

Dialogue Modifier la séquence d'envoi / Dialogue Modifier la séquence de réception

Fonction	Combinaison de touches
Annuler	Esc
Caractère de remplacement ' ? (correspond à un caractère)	F7
Caractère de remplacement '#' (correspond à un ou à zéro caractère)	F8

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

Fonction Caractère '&' (délai pour *x* * 0,01 sec.)	`F9`
Symbole de fonction '%' - (état d'interruption)	`F10`
Fonction Signes ' ! (signaux Handshake)	`F11`

Domaine de la documentation

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

Fonction	Combinaison de touches
Police de caractères par défaut	`Ctrl+D`

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

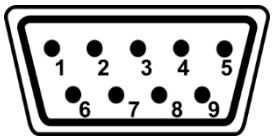
12.3 Connexions RS232 / Pinout

Les connexions les plus courantes pour la communication RS232 sont les suivantes

SUB D9 à 9 pôles (norme EIA/TIA 574). Introduit par IBM et largement répandu. Voir ci-dessous.
SUB D25 à 25 broches (RS232-C). Il s'agit du connecteur d'origine, introduit pour la norme RS232. Il offre un canal de communication secondaire.
RJ45 à 8 broches (différentes configurations de broches pour le câblage Cisco/Yost, EIA/TIA-561 et autres configurations de broches spécifiques aux fabricants).

RS232 SUB D9 (D-Sub DB9) Affectation des broches

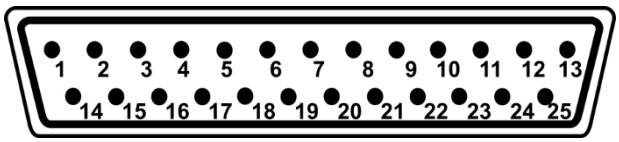
Vue : vue dans le connecteur.
Affectation : Du point de vue du DTE (le DTE envoie des données sur la ligne TX Transmit Data, tandis que le DTE reçoit des données sur cette ligne)



N° de broche	Nom du signal	Description	DTE on/off
1	DCD	Reconnaissance des supports de données	Entrée
2	RX	Réception de données	Entrée
3	TX	Transmettre des données	Édition
4	DTR	Terminal de données prêt	Édition
5	SGND	Masse du signal	-
6	DSR	Ensemble de données prêt	Entrée
7	RTS	Demande d'envoi	Édition
8	CTS	Autoriser l'envoi	Entrée
9	RI	Indicateur de bague	Entrée

RS232 SUB D25 (D-Sub DB25) Affectation des broches

Vue : vue sur le connecteur.
Affectation des broches : Vue d'un DTE.



N° de broche	Nom du signal	Description
1	-	Terre de protection/de blindage
2	TX	Transmettre des données
3	RX	Réception de données
4	RTS	Demande d'envoi
5	CTS	Autoriser l'envoi
6	DSR	Ensemble de données prêt

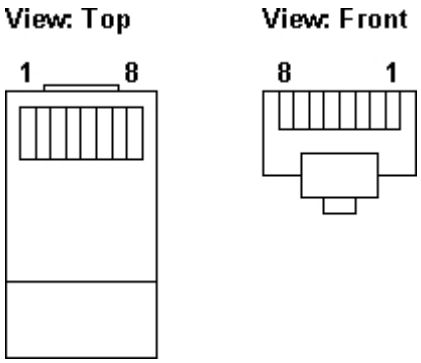
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

7	SGND	Masse du signal
8	DCD	Reconnaissance des supports de données

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

9	-	Réservé
10	-	Réservé
11	-	Non attribué
12	SDCD	Reconnaissance secondaire des disques
13	SCTS	Libération secondaire vers Envoyer
14	STx	Envoi secondaire de données
15	TxCLK	Envoyer l'horloge
16	SRx	Données secondaires reçues
17	RxCLK	Horloge de réception
18	LL	Loopback local
19	SRTS	Demande secondaire d'envoi
20	DTR	Terminal de données prêt
21	RL/SQ	Détecteur de bouclage à distance / de qualification du signal
22	RI	Indicateur de bague
23	CH/CI	Sélecteur de débit de signal
24	ACLK	Horloge auxiliaire
25	-	Non attribué

RJ45 Affectation des broches à 8 pôles



Pour les connecteurs RJ45 dans la communication RS232, il existe plusieurs affectations de broches contradictoires, qui sont également utilisées :

Console Cisco / Câble Yost / Câble Rollover Applications

Affectation : Du point de vue de la DEE (la DEE transmet des données sur la ligne TX Transmit Data)

N° de broche	Nom du signal	Description
1	CTS	Autoriser l'envoi
2	DCD	Reconnaissance des supports de données
3	RX	Réception de données
4	SGND	Masse du signal
5	SGND	Masse du signal
6	TX	Transmettre des données
7	DTR	Terminal de données prêt
8	RTS	Demande d'envoi

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

REMARQUE : l'affectation des broches Cisco/Yost est utilisée pour les câbles qui sont câblés "en miroir" à une extrémité, comme un câble null-modem avec handshaking. Chaque appareil a le même RJ45

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

prise et transfère les données sur la même broche. Voir aussi le Yost Serial Device Wiring Standard .

Norme EIA/TIA-561 pour connecteurs modulaires RJ45/8P8C

N° de broche	Nom du signal	Description
1	DSR / RI	Indicateur de disponibilité de l'ensemble de données/anneau
2	DCD	Reconnaissance des supports de données
3	DTR	Terminal de données prêt
4	SGND	Masse du signal
5	RX	Réception de données
6	TX	Transmettre des données
7	CTS	Autoriser l'envoi
8	RTS	Demande d'envoi

REMARQUE : Bien qu'il s'agisse d'une norme officielle, il est plus probable que vous trouviez des produits RS232 RJ45 avec un autre brochage, soit la variante Cisco/Yost ci-dessus, soit des brochages spécifiques au fabricant, par exemple MOXA Nport.

12.4 Câble RS232 standard

Connexions RS232 classiques

Lors de la connexion de deux appareils série, différents types de câbles doivent être utilisés en fonction des caractéristiques de l'appareil série et du type de communication utilisé.

Aperçu des connexions RS232 SUB D9 (D-Sub DB9)

Périphérique série 1	Périphérique série 2	Contrôle de flux (handshaking)	câble recommandé
DTE (Fin des données (appareil) équipement)	DTE	pas de signaux Handshake	câble null-modem simple
DTE	DTE	DTE/DCE contrôle de flux matériel compatible	Câble null-modem avec handshaking partiel
	DCE (données de communication) de l'établissement)	pas de signaux Handshake	simple câble droit
DTE	DCE	Contrôle du flux de matériel	câble droit plein
DCE	DCE	pas de signaux Handshake	câble null-modem simple mais avec des connecteurs SUB-D9 aux deux extrémités
DCE	DCE	Contrôle de flux matériel	Câble null-modem avec handshaking partiel, mais avec SUB-D9- des connecteurs aux deux

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

			extrémités

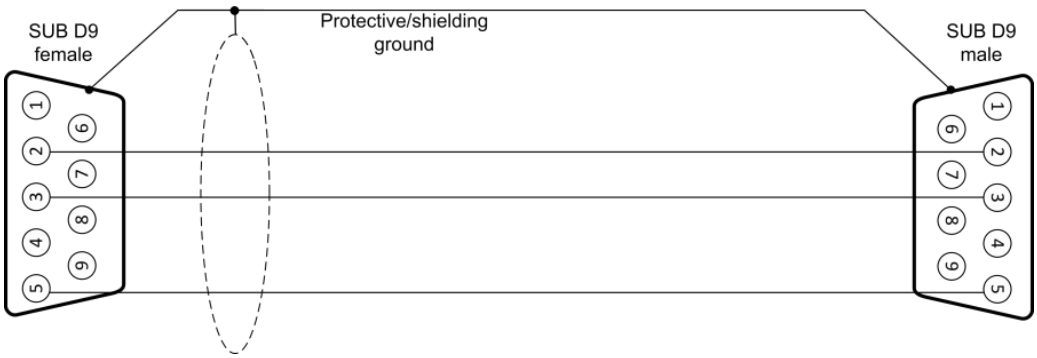
REMARQUE : une bonne alternative pour établir une connexion correcte entre différents périphériques DTE et DCE est d'utiliser le Yost Serial Device Wiring Standard de Dave Yost.

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

SUB D9 Câble droit simple

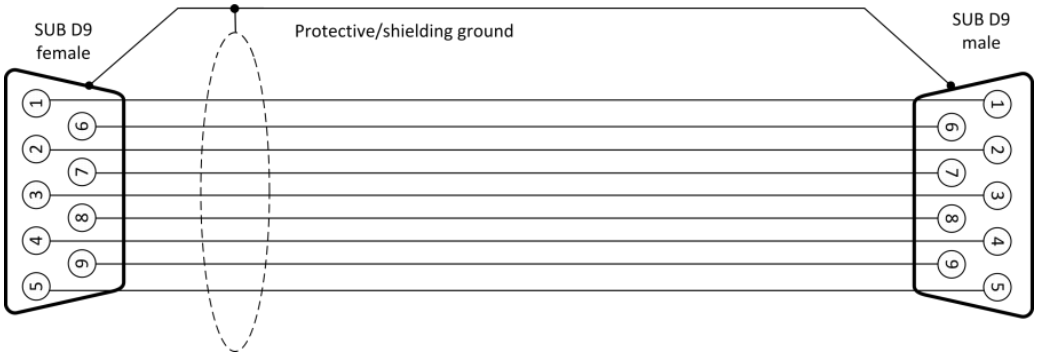
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de

Domaine d'application : communication DTE-DCE pour laquelle aucun signal de poignée de main supplémentaire n'est utilisé.



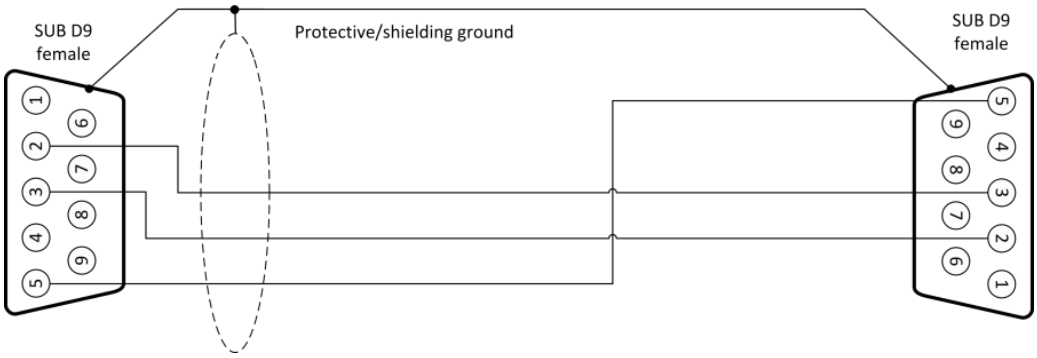
SUB D9 Câble droit complet

Domaine d'application : communication DTE-DCE avec contrôle de flux matériel par des signaux Handshake supplémentaires.



SUB D9 Câble null-modem simple sans handshaking

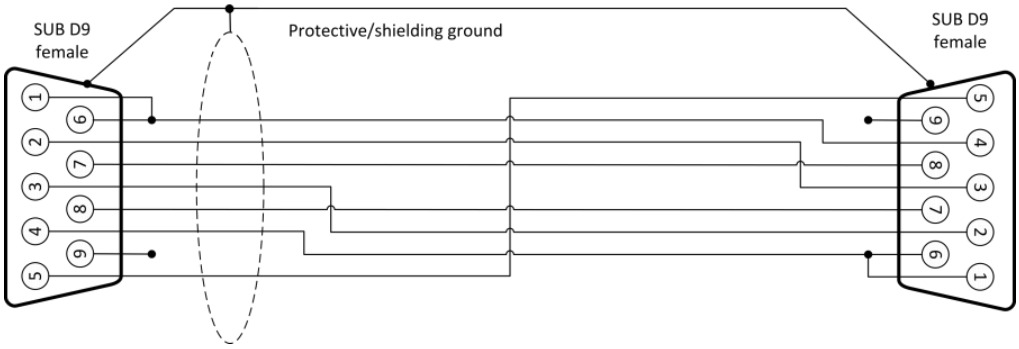
Domaine d'application : communication DTE-DTE qui n'utilise pas de signaux de handshake supplémentaires.



SUB D9 Câble null-modem avec full handshaking

Domaine d'application : communication DTE-DTE avec contrôle de flux matériel compatible DTE/DCE. Fonctionne également lorsqu'aucun signal de handshake n'est utilisé.

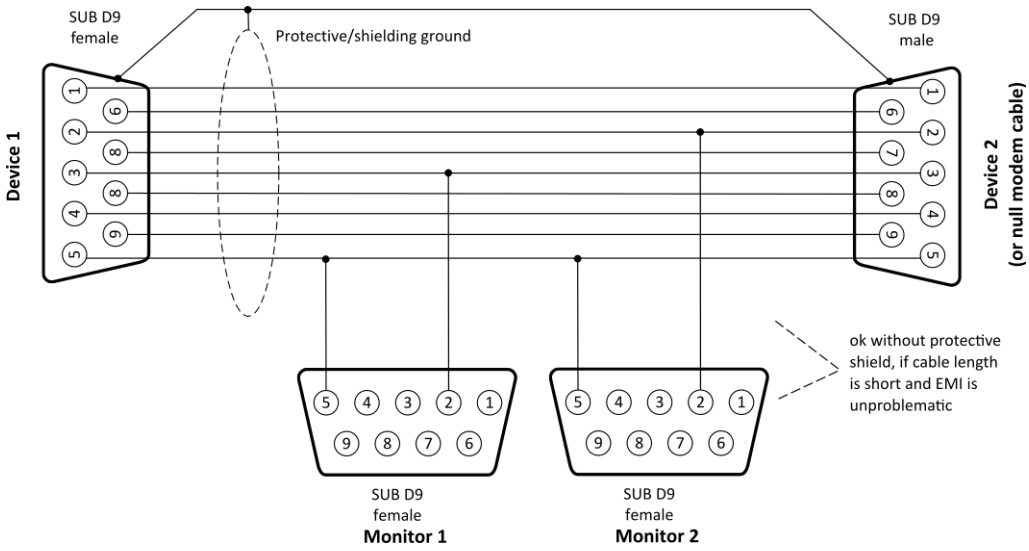
Copyright 2002-2023 www.fuh-edv.de / www.kickdrive.de



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

12.5 Docklight câble de surveillance RS232 SUB D9

Docklight Monitoring Cable est un câble de surveillance RS232 full duplex conçu pour surveiller la communication sérielle entre deux appareils.



Nous proposons un câble de surveillance RS232 robuste et entièrement blindé comme accessoire. Vous trouverez plus de détails sur nos pages d'aperçu des produits et sur la fiche technique du câble de surveillance Docklight.

REMARQUE : nos prises de données Docklight Tap ou Tap Pro / Tap RS485 offrent des caractéristiques de surveillance supérieures et ne nécessitent pas deux ports COM RS232 libres sur votre PC. Ce n'est que dans les applications rares ou anciennes que le câble de surveillance Docklight reste le choix préféré aujourd'hui.

CONSEIL : Une solution économique et rapide pour une surveillance simple consiste à fabriquer son propre câble de surveillance à partir d'un câble plat et de connecteurs autodénudants SUB D9, disponibles dans tous les magasins d'électronique.

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

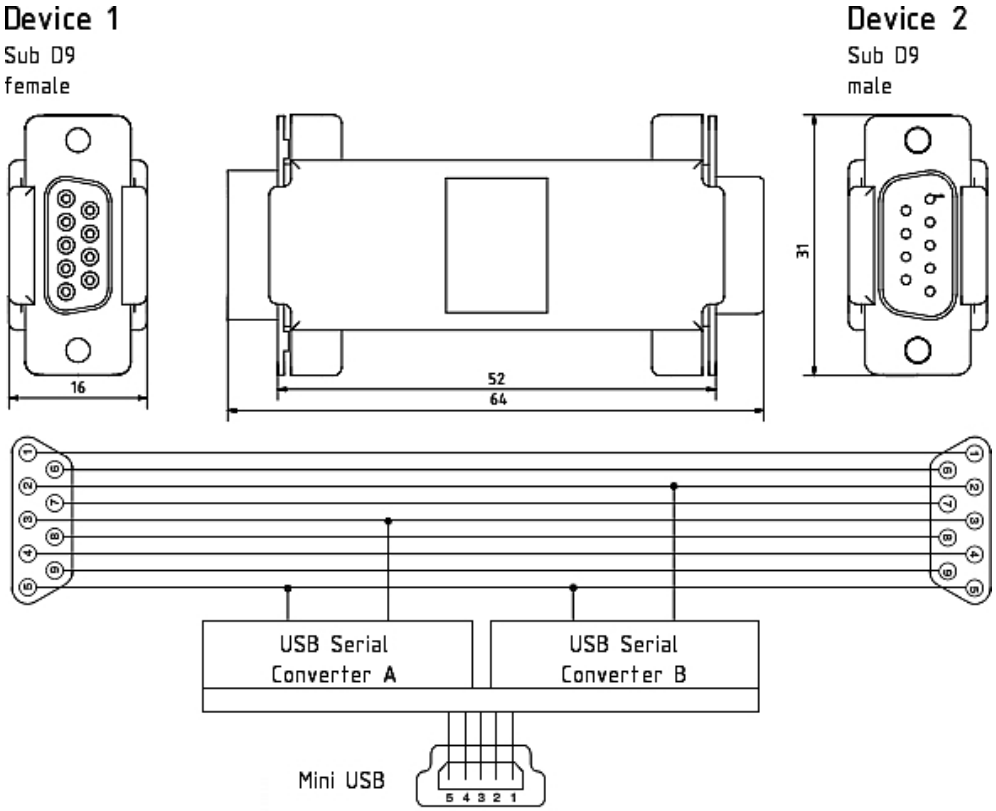
12.6 Robinet Docklight

Docklight Tap est une solution de surveillance des communications RS232 en duplex intégral pour le port USB.

Domaine d'application : surveillance de la communication sérielle entre deux appareils

Docklight a un support intégré pour le Docklight Tap. Il reconnaît le convertisseur USB série à double port et offre un accès à grande vitesse et à faible latence aux données de surveillance. Utilisez le **mode de surveillance Docklight** et les paramètres de canal de réception **TAP0** / **TAP1**.
Pour plus de détails, consultez les paramètres du projet Docklight et le guide pour obtenir la meilleure précision de timing.

Vous trouverez également de plus amples informations sur le Docklight Tap dans nos pages de présentation des produits.



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

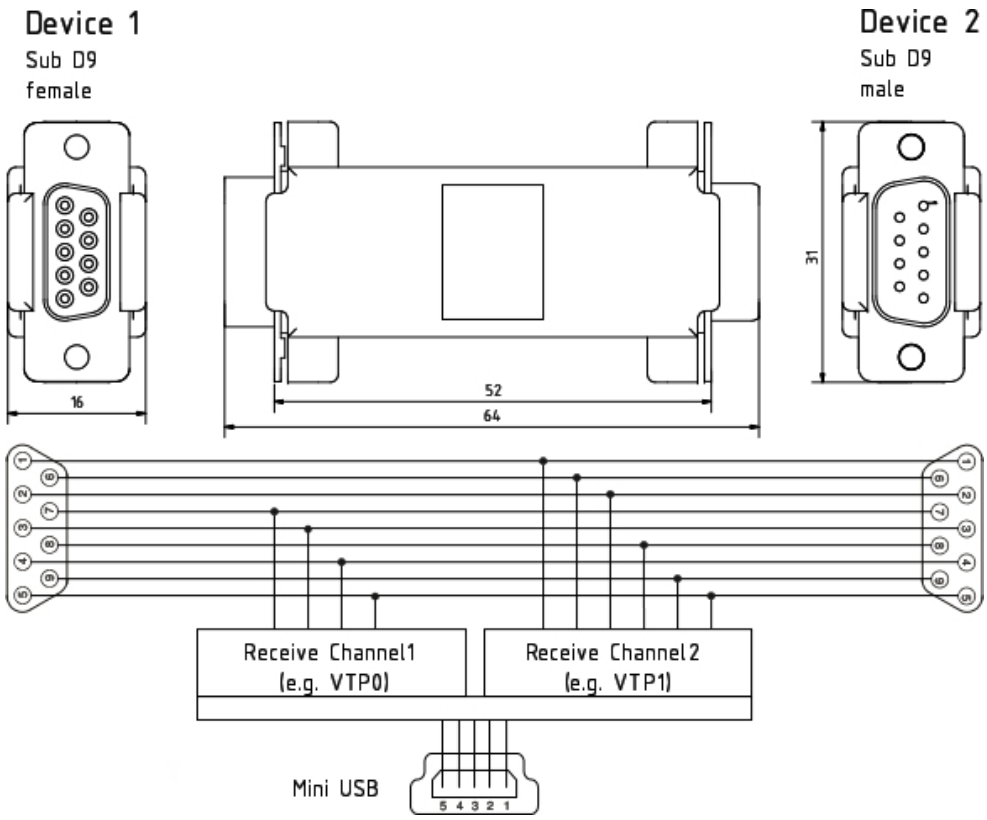
12.7 Robinet Docklight Pro / Robinet 485

Docklight Tap Pro et Docklight Tap 485 sont des solutions de surveillance avancées à haute résolution pour la connexion USB. Ils permettent de réaliser de véritables mesures temporelles de l'ordre de la milliseconde et de surveiller les connexions de données à haut débit, y compris les lignes d'état/d'accrochage RS232. Elles sont prises en charge par Docklight de la même manière que Docklight Tap.

Pour les applications Docklight Tap Pro et Tap 485, utilisez les paramètres Docklight **Monitoring Mode** et Receive Channel **VTP0** / **VTP1**. Pour plus de détails, consultez les paramètres du projet Docklight.

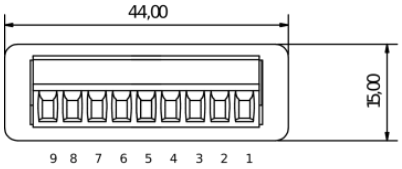
Pour plus d'informations sur le Docklight Tap Pro et le Docklight Tap 485, consultez également nos pages de présentation des produits.

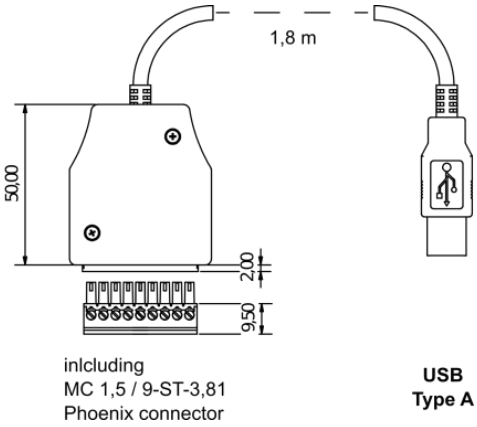
Docklight Tap Pro



Robinet Docklight RS485

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





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

Glossaire / Termes utilisés

13 Glossaire / Termes utilisés

13.1 Action

Pour une séquence de réception, l'utilisateur peut définir une action qui sera exécutée après la réception de la séquence indiquée. Les actions possibles sont

Envoi d'une séquence d'envoi
Seules les séquences d'envoi sans caractères de remplacement peuvent être utilisées Insertion d'un commentaire
Un texte personnalisé ou un horodatage supplémentaire est ajouté au message Fenêtre de données et fichier journal

- Déclenchement d'un instantané

- Interruption de la communication

13.2 Pause

Un état d'interruption dans une connexion RS232 se caractérise par le fait que la ligne de transmission passe à l'espace (logiquement 0) pendant une durée supérieure à la longueur maximale de la trame de caractères, y compris les bits de départ et d'arrêt. Certains protocoles d'application, par exemple LIN, utilisent cette situation à des fins de synchronisation.

13.3 Signes

Un caractère est l'unité de base des informations traitées par Docklight. Docklight utilise toujours des caractères de 8 bits. Toutefois, les paramètres de communication permettent également de transmettre des données avec 7 bits ou moins. Dans ce cas, seul un sous-ensemble des 256 caractères de 8 bits possibles est utilisé, mais les caractères sont toujours stockés et traités dans un format de 8 bits.

13.4 CRC

Code de redondance cyclique. Un CRC est une méthode permettant de détecter si une séquence/un message reçu a été altéré(e), par exemple par des erreurs de transmission. Pour ce faire, une valeur de somme de contrôle supplémentaire est construite, qui dépend des données utiles du message, et cette valeur est ensuite ajoutée au message original. Le destinataire calcule la somme de contrôle à partir des données reçues et la compare à la valeur CRC transmise afin de déterminer si le message est inchangé. Les CRC sont souvent utilisés car ils permettent de détecter les erreurs de transmission typiques (erreurs de bits, erreurs de rafale) avec une très grande précision.

Les algorithmes CRC sont basés sur l'arithmétique polynomiale et sont disponibles dans de nombreuses versions différentes. Les algorithmes courants sont CRC-CCITT, CRC-16 et CRC-32. Un exemple de protocole d'application utilisant un CRC est Modbus over Serial Line.

Un article populaire sur les CRC est "CRC Implementation Code in C" par Michael Barr, précédemment publié sous le titre "Slow and Steady Never Lost the Race" et "Easier Said Than Done" :
https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculation-C-Code

La fonctionnalité CRC de Docklight Scripting (DL.CalcChecksum) a été inspirée par

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

l'article ci-dessus et la bibliothèque CRC Boost proposée : http://www.boost.org/libs/crc/index.html

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

Enfin, si vous êtes vraiment fasciné par l'alchimie CRC, vous tomberez tôt ou tard sur l'article classique suivant, publié en 1993 : "A Painless Guide to CRC Error Detection Algorithms" par Ross N. Williams : http://ross.net/crc/crcpaper.html / http://ross.net/crc/download/crc_v3.txt

13.5 DCE

Périphériques de communication de données. Les termes DCE et DTE font référence aux périphériques série situés de part et d'autre d'une connexion RS232. Un modem est un exemple typique d'appareil DCE. Les DCE sont généralement équipés d'une prise **SUB** D9 ou SUB D25. Voir aussi DTE.

13.6 DTE

les terminaux de données. Les termes DCE et DTE font référence aux périphériques série situés de part et d'autre d'une connexion RS232. Un PC ou un terminal sont des exemples d'appareils DTE typiques. Les DTE sont généralement équipés d'un connecteur **SUB** D9 ou SUB D25. Toutes les spécifications de brochage sont écrites dans la perspective DTE. Voir aussi DCE.

13.7 Contrôle du débit

Le contrôle de flux fournit un mécanisme permettant d'interrompre la transmission lorsqu'un appareil est occupé ou ne peut pas continuer à communiquer pour une raison quelconque. Le DEE et le DEE doivent se mettre d'accord sur le mécanisme de contrôle de flux utilisé pour une session de communication. Il existe deux types de contrôle de flux : matériel et logiciel.

Contrôle de flux matériel
Utilise des signaux de tension sur les lignes d'état RS232 RTS / DTR (définies par le DEE) et CTS / DSR (définies par le DÜE) pour contrôler l'envoi et la réception de données. Voir aussi l'affectation des broches RS232.

Contrôle de flux logiciel
Utilise des caractères de contrôle ASCII spéciaux (XON / XOFF) pour contrôler la transmission des données. Le contrôle de flux logiciel nécessite des données de communication textuelles ou d'autres données qui ne contiennent pas de caractères XON ou XOFF.

13.8 HID

HID (Human Interface Device) est une classe de périphériques et une API utilisée pour les périphériques USB et Bluetooth.

Docklight Scripting prend en charge l'accès HID via VID / PID (ID fabricant / ID produit) ou le chemin d'accès complet du périphérique Windows USB. Docklight Scripting permet des transferts de données binaires et orientées texte via des rapports d'entrée et de sortie HID.

Une application fréquente (en plus de l'intégration clavier/souris Windows standard) pour la classe des appareils HID sont les appareils intégrés avec leur propre protocole pour une transmission simple des données, par exemple les appareils de mesure ou les capteurs.

ASTUCE : pour plus d'informations sur l'accès aux appareils HID via Docklight Scripting, vous pouvez également consulter nos ressources d'assistance en ligne sur www.docklight.de/support/.

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

REMARQUE : Docklight Scripting ne prend actuellement en charge que les rapports d'entrée/sortie
Les rapports ne sont *pas* pris en charge.

HID. Feature-

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

13.9 LIN

Réseau local d'interconnexion (Local Interconnect Network). Un bus de communication série économique pour les systèmes électroniques distribués dans les véhicules, en particulier pour les composants simples tels que les moteurs de porte, les commandes au volant, les capteurs de climatisation, etc. Voir aussi l'entrée Wikipedia sur LIN .

13.10 Modbus

Modbus est un protocole de message de la couche application qui permet une communication client/serveur entre des appareils connectés à différents types de bus ou de réseaux. Il est généralement utilisé comme "Modbus over Serial Line" dans les réseaux RS422/485, mais peut également être mis en œuvre avec TCP over Ethernet ("Modbus TCP").

Deux modes de transmission série différents sont définis pour Modbus : Le "mode RTU" pour les transmissions binaires 8 bits et le "mode ASCII". Le "mode RTU" est le mode standard qui doit être implémenté par tous les appareils.

Vous trouverez une spécification complète du protocole Modbus sur le site www.modbus.org.

13.11 Bus multipoint (MDB)

Le bus multipoint (MDB) est une application RS232/RS485 plus exotique, utilisée par exemple dans les commandes d'automates, qui nécessite un UART compatible avec 9 bits. Le 9e bit de données est utilisé pour choisir entre un mode ADDRESS et un mode DATA.

Un moyen de surveiller et de simuler de telles connexions de communication en utilisant des UART 8 bits standard,
c'est-à-dire les convertisseurs RS232 vers USB standard, est d'utiliser des changements de parité temporaires.

Pour plus d'informations et de détails, consultez Wikipedia sur MDB et la spécification MDB 3.0 originale.

13.12 Tube désigné

Un tube nommé est un mécanisme de mémoire partagée qui peut être utilisé pour la communication entre deux processus sur un PC Windows.

Docklight Scripting peut ouvrir une connexion client à un serveur de tuyaux nommés et envoyer ou recevoir des données ASCII 8 bits ou des octets.

Vous trouverez des détails sur les Named Pipes dans le Windows Development Center.

13.13 Ordre de réception

Une Receive Sequence est une séquence qui peut être reconnue par Docklight dans les données série entrantes. Une séquence de réception est spécifiée par
1. un nom unique (par exemple "Modem Answer OK"),
2. une chaîne de caractères (par exemple "6F 6B 13 10" au format HEX),
3. une action qui sera déclenchée lorsque Docklight recevra la séquence définie.

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

13.14 RS232

La norme RS232 est définie par l'EIA/TIA (Electronic Industries Alliance / Telecommunications Industry Associations). La norme définit un mécanisme de transmission de données en série asynchrone ainsi que les caractéristiques physiques et électriques de l'interface.

RS232 utilise des flux de bits sériels qui sont transmis à une vitesse de transmission prédéfinie. Les informations sont divisées en caractères de 5 à 8 bits.
Des bits de départ et d'arrêt supplémentaires sont utilisés pour la synchronisation et un bit de parité peut être inclus comme simple mécanisme de détection d'erreur.

L'interface électrique comprend des pilotes de ligne asymétriques, c'est-à-dire que tous les signaux sont représentés par une tension par rapport à une masse de signal commune. RS232 définit deux états pour les signaux de données : Mark State (ou 1 logique) et Space State (ou 0 logique). La plage de tension pour la représentation de ces états est définie comme suit :

État du signal	**Plage de tension de l'émetteur**	**Plage de tension du récepteur**
Mark (logique 1)	-15V à -5V	-25V à -3V
Espace (0 logique)	+5V à +15V	+3V à +25V
Indéterminé	-5V à +5V	-3V à +3V

Les caractéristiques physiques de la norme RS232 sont décrites dans la section Connexions RS232 / Brochage.

13.15 RS422

Une liaison de communication RS422 est une liaison à quatre fils avec des drivers de ligne symétriques. Dans un système différentiel symétrique, un signal est transmis par deux fils (A et B). L'état du signal est représenté par la tension aux bornes des deux fils. Bien qu'une connexion de masse de signal commune soit nécessaire, elle n'est pas utilisée pour déterminer l'état du signal au niveau du récepteur. Il en résulte une immunité élevée contre les EMI (interférences électromagnétiques) et permet des longueurs de câble de plus de 1000 m, selon le type de câble et le débit en bauds.

La norme EIA RS422-A "Electrical characteristics of balanced voltage digital interface circuits" définit les caractéristiques d'une interface RS422.

Les caractéristiques de l'émetteur et du récepteur selon RS422-A sont les suivantes :

État du signal	**Tension différentielle du transmetteur V_{AB}**	**Tension différentielle du récepteur V_{AB}**
Marquage (ou logique 1)	-6V à -2V	-6V à -200mV
Espace (ou 0 logique)	+2V à +6V	+200mV à 6V
Indéterminé	-2V à +2V	-200mV à +200mV

Tension de mode commun admissible V_{cm} (tension moyenne des bornes A et B par rapport à la masse du signal) : -7V à +7V

13.16 RS485

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

RS485 définit une ligne de transmission bifilaire symétrique qui peut être utilisé en commun comme ligne de bus par jusqu'à 32 paires de pilotes-récepteurs. De nombreuses caractéristiques de la

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

L'émetteur et le récepteur sont les mêmes que pour RS422. Les principales différences entre RS422 et RS485 sont les suivantes

Transmission bifilaire (semi-duplex) au lieu de la transmission quadrifilaire Pilote de ligne symétrique avec possibilité de tristate. Le driver de ligne RS485 dispose d'un "signal de validation" supplémentaire qui est utilisé pour connecter et déconnecter le driver de sa connexion de sortie. Le terme "tristate" fait référence aux trois états différents possibles sur la borne de sortie : Mark (logique 1), Space (logique 0) ou "disconnected".

- Plage de tension de mode commun étendue (V_{cm}) de -7V à +12V.

La norme EIA RS485 "Standard for electrical characteristics of generators and receivers for use in balanced digital multipoint systems" définit les caractéristiques d'un système RS485.

13.17 Envoyer une séquence

Une séquence d'envoi est une séquence qui peut être envoyée par Docklight. Une séquence d'envoi est indiquée par
1. un nom unique (par exemple "Régler le volume du modem"),
2. une chaîne de caractères (par exemple "41 54 4C 0D 0A" au format HEX).

Il existe deux possibilités pour que Docklight envoie une séquence :

L'envoi d'une séquence peut être déclenché manuellement en appuyant sur le bouton "Envoyer" dans la liste "Séquences d'envoi".
(voir la fenêtre principale).

L'envoi d'une séquence peut être une réaction possible si Docklight détecte une séquence de réception spécifique dans les données entrantes (voir Action).

13.18 Séquence

Une séquence se compose d'un ou de plusieurs caractères de 8 bits. Une séquence peut être n'importe quelle partie de la communication série que vous analysez. Elle peut être constituée de caractères ASCII imprimables, mais peut également contenir tout caractère non imprimable compris entre 0 et 255 décimales.
exemple :
ATL2 (format ASCII)
41 54 4C 0D 0A (format HEX)
Cette séquence est une commande de modem permettant de régler le volume du haut-parleur sur les modems compatibles AT. Elle contient un caractère Carriage Return (0D) et un caractère Line Feed (0A) à la fin de la ligne.

La taille maximale des séquences dans Docklight est de 1024 caractères.

13.19 Indice de séquence

L'index de séquence est le numéro d'élément d'une séquence d'envoi dans la liste des séquences d'envoi ou d'une séquence de réception dans la liste des séquences de réception. L'index de séquence est affiché dans le coin supérieur gauche de la boîte de dialogue "Editer la séquence d'envoi" ou "Editer la séquence de réception".

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

13.20 Serveur de périphériques série

Un Serial Device Server est un périphérique réseau qui offre une ou plusieurs interfaces COM sérielles (RS232, RS422/485) et qui envoie/reçoit les données sérielles via un réseau Ethernet. Les Serial Device Servers sont une méthode courante pour mettre à niveau les appareils existants contrôlés par une interface série et les rendre "compatibles réseau".

13.21 Instantané

Créer un instantané dans Docklight signifie créer une représentation de la communication série juste avant et après la détection d'une séquence de déclenchement. Cela est utile lorsque vous testez pour une erreur rare caractérisée par une séquence spécifique. Pour plus d'informations, voir Interception d'une séquence spécifique et création d'un instantané...

13.22 TCP

Protocole de contrôle de transmission. TCP est, avec UDP, le principal protocole de la couche transport utilisé dans les réseaux IP. TCP est orienté connexion - avant que deux hôtes de réseau puissent communiquer avec TCP, ils doivent d'abord établir une connexion. TCP est un protocole de flux d'octets qui garantit la livraison. TCP garantit que les paquets de données sont transmis sans erreur et dans le bon ordre, même si le réseau sous-jacent n'est pas fiable.

TCP utilise les numéros de port 1-65535 pour identifier les points finaux des applications. Des exemples d'applications TCP connues et de numéros de port sont FTP (21), TELNET (23), SMTP (25), HTTP (80) et POP3 (110).

13.23 Déclencheur

Un déclencheur est une séquence de réception dont l'option "Trigger" est activée (voir Dialogue : Editer la séquence de réception). Si la fonction d'instantané est activée, Docklight n'émettra pas de sortie tant qu'une séquence de déclenchement n'aura pas été détectée dans les données de communication série. Pour plus d'informations, voir Interception d'une séquence spécifique et création d'un snapshot...

13.24 UART

Récepteur / émetteur asynchrone universel. L'UART est le composant matériel qui exécute les principales tâches de communication série :
- Conversion de caractères en un flux binaire sériel
- Ajout de bits de départ/d'arrêt/de parité et vérification des erreurs de parité du côté du récepteur
- toutes les tâches liées au timing, aux vitesses de transmission et à la synchronisation

Les UART traditionnels sont compatibles avec l'UART 16550A. Ils contiennent une mémoire tampon de 16 octets pour les données entrantes (RX FiFo) et une mémoire tampon de 16 octets pour les données sortantes (TX FiFo).
Normalement, ces tampons peuvent être désactivés/activés via le gestionnaire de périphériques Windows et l'ouverture de la page de propriétés pour le port COM correspondant (par ex. COM1).

13.25 UDP

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

datagramme . UDP est un protocole de la couche transport utilisé dans les réseaux IP. UDP est un protocole sans connexion - les partenaires de communication n'établissent pas de connexion avant la transmission des données. UDP n'offre pas de transmissions fiables ou ordonnées.

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

Les datagrammes peuvent arriver dans le mauvais ordre, arriver en double ou être perdus pendant la transmission. Les applications qui nécessitent une livraison ordonnée et fiable des flux de données devraient plutôt utiliser TCP.

UDP est plus rapide que TCP et présente des avantages pour de nombreuses applications de réseau légères ou sensibles au facteur temps. UDP est utilisé pour le système de noms de domaine sur Internet, pour les applications de streaming média telles que Voice Over IP et pour la diffusion sur les réseaux IP.

UDP utilise les numéros de port 1-65535 pour identifier les points d'extrémité des applications. Des exemples de services UDP et de numéros de port connus sont DNS (53), TIME (37) et SNMP (161 et 162).

13.26 Modem zéro virtuel

Un null-modem virtuel est un pilote logiciel pour PC qui émule deux ports COM série reliés par un câble null-modem. Lorsqu'une application PC envoie des données via un port COM virtuel, une deuxième application PC peut recevoir ces données via le deuxième port COM virtuel et vice versa.

En utilisant un pilote null-modem virtuel sur votre PC, vous pouvez facilement déboguer et simuler des connexions de données série sans utiliser de véritables ports et câbles RS232.

Les connexions COM virtuelles n'offrent pas le même timing que les connexions RS232 réelles et n'émulent généralement pas la transmission réelle bit à bit à une vitesse de transmission prédéfinie. Chaque paquet de données envoyé via le premier port COM apparaît presque immédiatement dans le tampon de réception du deuxième port COM. Pour la plupart des débogages et des simulations, cette limitation peut être facilement tolérée. Certains pilotes virtuels null-modem proposent un mode d'émulation de débit en bauds supplémentaire, dans lequel la transmission des données est retardée afin d'émuler une véritable connexion RS232 et son débit limité.

Une solution Windows open source qui a été testée avec succès avec Docklight est disponible à l'adresse suivante
https://sourceforge.net/projects/com0com/
Nous recommandons la version x64 signée de com0com v2.2.2.0, que nous avons testée avec succès *sous Windows 10* et *Windows 11* :
https://sourceforge.net/projects/com0com/files/com0com/2.2.2.0/com0com-2.2.2.0-x64-fre-signed.zip/download

13.27 Espace réservé

Un caractère de remplacement est un caractère spécial qui sert d'espace réservé dans une séquence. Il peut être utilisé pour des séquences de réception lorsque des parties des données reçues ne sont pas spécifiées, par exemple des valeurs de mesure rapportées par un appareil sériel. Les caractères joker peuvent également être utilisés pour prendre en charge des paramètres dans une séquence d'envoi.

Les types de caractères génériques suivants sont disponibles dans Docklight :
Caractère de remplacement ' ? (F7) : Correspond exactement à un caractère quelconque (n'importe quel code ASCII entre 0 et 255)
Caractère de remplacement '#' (F8) : Correspond à zéro ou à un caractère. Ceci est utile pour la prise en charge des arguments de commande de longueur variable (par exemple un mot d'état) dans les séquences d'envoi/réception. Pour des exemples et plus d'informations, voir Vérification des séquences avec caractères aléatoires ou Envoi de commandes a v e c

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

chemin.

Autres caractères génériques autorisant des données aléatoires :

Fonction Caractère ' ! (F12) : Comparaison bit par bit. Ceci est utile lorsqu'il y a un ou plusieurs bits dans un caractère qui doivent être testés pour une valeur spécifique. Voir le caractère de fonction '^'. (F12) - comparaison bit à bit pour plus de détails et un exemple.

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