DMX 2018

1.1

Généré par Doxygen 1.7.6.1

Jeudi Juin 7 2018 20 :37 :25

Table des matières

1 Page principale du projet DMX 2018

1.1 Introduction

Système permettant de commander un ensemble de projecteurs (lyres, scanners, lasers etc.) compatibles avec le bus DMX512, le bus standard du spectacle. Il peut être utilisé dans le cadre d'animations de soirée, de spectacles, ...

1.2 Table des matières

Dépôt SVN : `https://svn.riouxsvn.com/dmx-2018`

2 Changelog

---

r98 | treynier | 2018-06-07 17 :55 :54 +0200 (jeu. 07 juin 2018) | 1 ligne

release tag 1.1

---

r97 | treynier | 2018-06-07 17 :51 :39 +0200 (jeu. 07 juin 2018) | 1 ligne

documentation doxygen mise à jour

---

r96 | treynier | 2018-06-07 17 :18 :43 +0200 (jeu. 07 juin 2018) | 1 ligne

execution de spectacle operationelle

---

r95 | treynier | 2018-06-07 16 :45 :31 +0200 (jeu. 07 juin 2018) | 1 ligne

ajout et suppression de spectacle operationnels

---

r94 | treynier | 2018-06-07 15 :12 :40 +0200 (jeu. 07 juin 2018) | 1 ligne

creation de la partie IHM des spectacles

---

r93 | tdemont | 2018-06-07 12 :03 :16 +0200 (jeu. 07 juin 2018) | 1 ligne

Fusion de la version de la branche portage windows avec le trunk

---

r92 | treynier | 2018-06-06 16 :38 :55 +0200 (mer. 06 juin 2018) | 1 ligne

execution de sequence operationelle

---

r91 | tdemont | 2018-06-06 16 :10 :45 +0200 (mer. 06 juin 2018) | 1 ligne

Fermeture des fichiers 'appareils.xml' et 'consoles.xml' et test ajouté lors de la détection d'interface

---

r90 | treynier | 2018-06-05 16 :21 :11 +0200 (mar. 05 juin 2018) | 1 ligne

creation et execution des sequences implémentée, en attente de tests

---

r89 | tdemont | 2018-06-05 12 :19 :13 +0200 (mar. 05 juin 2018) | 1 ligne

Correction de bug lors de la selection du pilotage depuis l'IHM, ajout d'une fonction de détection de l'interface active et mise en commentaire de messages qDebug

---

r88 | treynier | 2018-06-05 11 :13 :26 +0200 (mar. 05 juin 2018) | 1 ligne

transfert du travail sur les scènes et séquences dans le trunk

---

r87 | tdemont | 2018-06-04 14 :16 :28 +0200 (lun. 04 juin 2018) | 1 ligne

Suppression d'un affichage redondant lors de la detection d'interface

---

r86 | treynier | 2018-06-04 09 :34 :19 +0200 (lun. 04 juin 2018) | 1 ligne

branche 1.1.2 : supression des scenes operationnelle

---

r85 | tdemont | 2018-06-04 09 :25 :33 +0200 (lun. 04 juin 2018) | 1 ligne

Terminaison du portage Windows et correction du bug obligeant à désélectionner et reséléctionner l'interface au démarrage

---

r84 | treynier | 2018-06-04 09 :02 :41 +0200 (lun. 04 juin 2018) | 1 ligne

branche 1.1 : execution des scens operationnelle

---

r83 | tvaira | 2018-06-01 09 :48 :34 +0200 (ven. 01 juin 2018) | 1 ligne

Remplacement de la classe CRS232 par QextSerialPort pour le portage Windows

---

r82 | tdemont | 2018-05-31 17 :19 :56 +0200 (jeu. 31 mai 2018) | 1 ligne

Import de la QextPort

---

r81 | treynier | 2018-05-31 16 :44 :05 +0200 (jeu. 31 mai 2018) | 1 ligne

Ajout de scène fonctionnel

---

r80 | treynier | 2018-05-31 14 :07 :34 +0200 (jeu. 31 mai 2018) | 1 ligne

Création de la branche 1.1.2

---

r79 | tdemont | 2018-05-30 17 :36 :22 +0200 (mer. 30 mai 2018) | 1 ligne

Correction d'un bug empechant le changement de pilotage depuis l'IHM

---

r78 | tvaira | 2018-05-26 08 :25 :08 +0200 (sam. 26 mai 2018) | 1 ligne

Modification de la méthode executerCommandeSysteme pour le portage Windows

---

r77 | tvaira | 2018-05-26 08 :12 :38 +0200 (sam. 26 mai 2018) | 1 ligne

Ajout de la bibliothèque QextSerialPort pour le portage Windows

---

r76 | tvaira | 2018-05-26 07 :55 :54 +0200 (sam. 26 mai 2018) | 1 ligne

Création branche 1.1 pour Windows

---

r75 | treynier | 2018-05-25 15 :08 :31 +0200 (ven. 25 mai 2018) | 1 ligne

correction du doxygen du tag 1.0

---

r74 | treynier | 2018-05-25 14 :42 :04 +0200 (ven. 25 mai 2018) | 1 ligne

---

r73 | tdemont | 2018-05-25 14 :33 :04 +0200 (ven. 25 mai 2018) | 1 ligne

Ajout de documentation doxygene oubliée

---

r72 | treynier | 2018-05-25 12 :21 :56 +0200 (ven. 25 mai 2018) | 1 ligne

tag de la version 1.0

---

r71 | tdemont | 2018-05-24 16 :37 :39 +0200 (jeu. 24 mai 2018) | 1 ligne

Correction d'une erreur de code où les ports des consoles n'étaient pas utiles

---

r70 | tdemont | 2018-05-23 14 :26 :12 +0200 (mer. 23 mai 2018) | 1 ligne

Correction d'un bug sur la liste de choix de modification de console

---

r69 | tdemont | 2018-05-22 16 :24 :57 +0200 (mar. 22 mai 2018) | 1 ligne

Suppression du mot Pilotage redondant lors du choix de pilotage

---

r68 | tdemont | 2018-04-26 15 :36 :26 +0200 (jeu. 26 avril 2018) | 1 ligne

L'utilisateur ne peut désormais plus écrire dans la zone de texte de la détection d'interface

---

r67 | tdemont | 2018-04-26 15 :16 :02 +0200 (jeu. 26 avril 2018) | 1 ligne

Correction du fonctionnement de la modification des paramètres de console, si un champ est vide, l'information à laquelle il correspond n'est pas modifiée

---

r66 | tdemont | 2018-04-26 14 :54 :58 +0200 (jeu. 26 avril 2018) | 1 ligne

Ajout de la modification des paramètres des consoles

---

r65 | tdemont | 2018-04-26 13 :30 :19 +0200 (jeu. 26 avril 2018) | 1 ligne

Toutes les consoles recoivent désormais le chiffre des dizaines du canal de départ actif

---

r64 | tdemont | 2018-04-26 12 :37 :09 +0200 (jeu. 26 avril 2018) | 1 ligne

Ajout de la possibilité de changer de projecteur depuis la console en appuyant sur la combinaison de touche Back + Up / Down

---

r63 | treynier | 2018-04-26 11 :46 :10 +0200 (jeu. 26 avril 2018) | 1 ligne

modification de scene.h et suppression de appareilScene.h , désormais, la structure scene gère directement le conteneur de canaux et valeurs

---

r62 | treynier | 2018-04-26 11 :44 :00 +0200 (jeu. 26 avril 2018) | 1 ligne

creation du contrôle de doublons d'appareils lors de la creation d'une scene

---

r61 | treynier | 2018-04-25 08 :08 :39 +0200 (mer. 25 avril 2018) | 1 ligne

Tag version 0.9 disponible

---

r60 | treynier | 2018-04-23 15 :25 :37 +0200 (lun. 23 avril 2018) | 1 ligne

Minimums de ChoixNBCanaux et ChoixCanalDepart paramétrés à 1

---

r59 | tdemont | 2018-04-19 21 :31 :41 +0200 (jeu. 19 avril 2018) | 1 ligne

Ajout du laser et des ses canaux dans le fichier appareils.xml

---

r58 | tvaira | 2018-04-19 20 :05 :37 +0200 (jeu. 19 avril 2018) | 1 ligne

Mise à jour Doxygen pour les classe CRS232 et EnttecDMXUSB

---

r57 | tdemont | 2018-04-19 19 :11 :42 +0200 (jeu. 19 avril 2018) | 1 ligne

Ajout de commentaire doxygene sur la classe myslider

---

r56 | tdemont | 2018-04-19 19 :03 :37 +0200 (jeu. 19 avril 2018) | 1 ligne

Ajout de l'affichage du chiffre des dizaines du numéro du premier canal actuel sur les L-ED de la console Wing, ajout de commentaires doxygene sur les classes XMLUtilitaire, PlaybackWing et IHM

---

r55 | treynier | 2018-04-19 18 :47 :45 +0200 (jeu. 19 avril 2018) | 1 ligne

mise à jour des commentaires doxygen de myslider

---

r54 | treynier | 2018-04-19 18 :33 :44 +0200 (jeu. 19 avril 2018) | 1 ligne

mise à jour des commentaires doxygen de DMXProjecteur

---

r53 | treynier | 2018-04-19 18 :16 :11 +0200 (jeu. 19 avril 2018) | 1 ligne

mise à jour de masse des commentaires doxygen aux classes IHM, IDProjecteurs-Sauvegarde et XMLUtilitaire

---

r52 | treynier | 2018-04-19 14 :42 :50 +0200 (jeu. 19 avril 2018) | 1 ligne

ajout de la mise à jour auto de ListeProjecteursPilotage lors de l'ajout/suppression d'un projecteur

---

r51 | tdemont | 2018-04-19 11 :38 :00 +0200 (jeu. 19 avril 2018) | 1 ligne

Ajout des projecteurs dans le fichier appareils.xml

---

r50 | treynier | 2018-04-19 10 :55 :00 +0200 (jeu. 19 avril 2018) | 1 ligne

correction d'un bug de decalage de canal de depart lors de l'enregistrement des projecteurs

---

r49 | treynier | 2018-04-19 10 :18 :48 +0200 (jeu. 19 avril 2018) | 1 ligne

correction d'un bug concernant la supression simultanée de tous les projecteurs enregistres

---

r48 | treynier | 2018-04-18 17 :53 :12 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout d'une securité lors de la saisie des noms de canaux durant la creation/modification d'un apparei

---

r47 | tdemont | 2018-04-18 17 :06 :28 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout de la gestion des paramètres des consoles, du fichier console.h et du fichier consoles.xml

---

r46 | treynier | 2018-04-18 15 :01 :38 +0200 (mer. 18 avril 2018) | 1 ligne

Ajout d'une securité lors de la saisie des noms de canaux durant la creation/modification d'un apparei

---

r45 | treynier | 2018-04-18 14 :30 :17 +0200 (mer. 18 avril 2018) | 1 ligne

Creation de l'option permettant de modifier un appareil DMX dans la liste d'appareils

enregistrés depuis le fichier XML et restructuration des methodes basées sur la gestion des appareils en les identifiant à partir de l'uuid plutot que du nom

---

r44 | tdemont | 2018-04-18 09 :57 :02 +0200 (mer. 18 avril 2018) | 1 ligne

Correction de la récupération de la derniere interface utilisée, le type d'interface est maintenant aussi récupéré

---

r43 | tdemont | 2018-04-18 09 :50 :14 +0200 (mer. 18 avril 2018) | 1 ligne

Suppression de la méthode testerInterface() (remplacée par detecterInterface())

---

r42 | tdemont | 2018-04-18 09 :47 :31 +0200 (mer. 18 avril 2018) | 1 ligne

Fonctionnalité de détéction des Interfaces et affichage de leurs informations ajoutée

---

r41 | tvaira | 2018-04-16 12 :35 :56 +0200 (lun. 16 avril 2018) | 1 ligne

Maj Doxyfile

---

r40 | tvaira | 2018-04-16 12 :27 :42 +0200 (lun. 16 avril 2018) | 1 ligne

Verification des TODO

---

r39 | tvaira | 2018-04-13 11 :32 :23 +0200 (ven. 13 avril 2018) | 1 ligne

Ajout méthode executerCommandeSysteme()

---

r38 | treynier | 2018-04-12 22 :06 :27 +0200 (jeu. 12 avril 2018) | 1 ligne

creation de nouvelles structures pour la recuperation des données des scenes depuis le fichier XML + récuperation des scènes du fichier xml dans l'IHM

---

r37 | tdemont | 2018-04-12 18 :58 :31 +0200 (jeu. 12 avril 2018) | 1 ligne

Correction de la suppression d'interface, la liste de choix de l'interface est maintenant mise à jour automatiquement

---

r36 | tdemont | 2018-04-12 17 :18 :21 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout de la fonctionnalité permettant l'enregistrement de la derniere interface DMX utilisée et changement du nom du fichier interfaces.xml en adaptateurs.xml

---

r35 | tdemont | 2018-04-12 15 :17 :53 +0200 (jeu. 12 avril 2018) | 1 ligne

Ajout et suppression d'interfaces

---

r34 | treynier | 2018-04-12 14 :25 :35 +0200 (jeu. 12 avril 2018) | 1 ligne

création de toute la partie Interface pour la création de scènes

---

r33 | tdemont | 2018-04-11 12 :00 :38 +0200 (mer. 11 avril 2018) | 1 ligne

Correction de la fonctionnalité d'ajout d'interface, la liste du choix de l'interface se met automatiquement à jour

---

r32 | tdemont | 2018-04-11 11 :48 :29 +0200 (mer. 11 avril 2018) | 1 ligne

Ajout de la fonctionnalité d'ajout d'interface

---

r31 | tdemont | 2018-04-09 16 :09 :35 +0200 (lun. 09 avril 2018) | 1 ligne

Modification test interface Enttec

---

r30 | tdemont | 2018-04-09 15 :26 :19 +0200 (lun. 09 avril 2018) | 1 ligne

Modification message interface DMX absente

---

r29 | treynier | 2018-04-09 15 :24 :49 +0200 (lun. 09 avril 2018) | 1 ligne

Ajout de l'option de suppression d'un appareil DMX ainsi que creation de la fenetre IHM pour la modification d'appareil DMX

---

r28 | tdemont | 2018-04-09 15 :10 :26 +0200 (lun. 09 avril 2018) | 1 ligne

Gestion interface DMX

---

r27 | treynier | 2018-04-06 11 :56 :48 +0200 (ven. 06 avril 2018) | 1 ligne

Finalisation de l'option d'enregistrement d'un nouveau projecteur via le fichier xml

---

r26 | treynier | 2018-04-05 14 :14 :26 +0200 (jeu. 05 avril 2018) | 1 ligne

Creation de la partie IHM de la fenetre d'options d'enregistrement d'un nouvel appareil DMX

---

r25 | treynier | 2018-04-05 11 :42 :16 +0200 (jeu. 05 avril 2018) | 1 ligne

Option pour supprimmer tous les projecteurs enregistrés depuis les parametres operationnelle

---

r24 | treynier | 2018-04-04 17 :49 :14 +0200 (mer. 04 avril 2018) | 1 ligne

creation de l'affichage des projecteurs enregistrés depuis appareils.xml

---

r23 | tdemont | 2018-04-03 15 :45 :07 +0200 (mar. 03 avril 2018) | 1 ligne

Ajout du fichier interfaces.xml

---

r22 | tvaira | 2018-04-03 15 :44 :27 +0200 (mar. 03 avril 2018) | 1 ligne

Retour sur la revue 2

---

r21 | treynier | 2018-03-30 12 :33 :23 +0200 (ven. 30 mars 2018) | 1 ligne

Rectification de l'ajout de idProjecteurSauvegarde dans le repertoire svn

---

r20 | treynier | 2018-03-30 12 :31 :54 +0200 (ven. 30 mars 2018) | 1 ligne

Creation de la partie IHM de l'ajout des projecteurs + creation de la classe IDProjecteur-Sauvegarde

---

r19 | treynier | 2018-03-29 16 :09 :01 +0200 (jeu. 29 mars 2018) | 1 ligne

ajout de la classe xmlutilitaire et des fichiers spectacle.xml, scenes.xml, sequences.xml et appareils.xml + correction des noms d'attributs de la classe IHM

---

r17 | tdemont | 2018-03-28 18 :00 :40 +0200 (mer. 28 mars 2018) | 1 ligne

Suppression des widgets superflus dans l'IHM

---

r16 | tdemont | 2018-03-28 09 :00 :26 +0200 (mer. 28 mars 2018) | 1 ligne

Ajout des commentaires doxygene pour les classes IHM et PlaybackWing

---

r15 | tvaira | 2018-03-24 10 :23 :07 +0100 (sam. 24 mars 2018) | 1 ligne

Ajout parametrage Doxygen

---

r14 | tdemont | 2018-03-22 17 :40 :37 +0100 (jeu. 22 mars 2018) | 1 ligne

Ajout de la fonctionnalité permettant le controle des projecteurs depuis soit l'application, soit la console.

---

r13 | treynier | 2018-03-22 16 :14 :00 +0100 (jeu. 22 mars 2018) | 1 ligne

Remplacement du code source fonctionnel, communication entre la chaîne DMX et l'application fonctionnelle

---

r12 | tdemont | 2018-03-22 15 :25 :06 +0100 (jeu. 22 mars 2018) | 1 ligne

Suppression de tout les fichiers en prévision de l'ajout des fichiers homologues indépendants de l'ui

---

r11 | treynier | 2018-03-17 09 :50 :12 +0100 (sam. 17 mars 2018) | 1 ligne

Deplacement des fichiers de documentation et revue dans le nas

---

r10 | tdemont | 2018-03-15 13 :52 :06 +0100 (jeu. 15 mars 2018) | 1 ligne

Ajout du fichier ui_IHM.h

---

r9 | tdemont | 2018-03-15 13 :43 :08 +0100 (jeu. 15 mars 2018) | 1 ligne

Ajout de la methode permettant l'extraction des donnees concernant les faders dans la classe PlaybackWing et ajout d'un raccourci permettant de fermer l'application.

---

r8 | tdemont | 2018-02-22 19 :04 :41 +0100 (jeu. 22 févr. 2018) | 1 ligne

Ajout de méthodes permettant la réception de données dans la classe PlaybackWing

---

r7 | treynier | 2018-02-21 10 :19 :23 +0100 (mer. 21 févr. 2018) | 1 ligne

Ajout du compte rendu et du journal de bord

---

r6 | treynier | 2018-02-21 10 :16 :44 +0100 (mer. 21 févr. 2018) | 1 ligne

Ajout du dossier de diagramme de classes

---

r5 | treynier | 2018-02-21 09 :53 :56 +0100 (mer. 21 févr. 2018) | 1 ligne

Création des classes et du code source dans le dossier src du projet

---

r4 | tdemont | 2018-02-08 12 :23 :38 +0100 (jeu. 08 févr. 2018) | 1 ligne

Ajout de l'arborescence src du trunk

---

r3 | tdemont | 2018-02-08 11 :35 :06 +0100 (jeu. 08 févr. 2018) | 1 ligne

Ajout du gantt, du diagramme de cas d'utilisation et de l'arborescence

---

r2 | tvaira | 2018-02-03 11 :11 :19 +0100 (sam. 03 févr. 2018) | 1 ligne

Ajout initial (tv)

---

r1 | www-data | 2018-02-03 11 :02 :12 +0100 (sam. 03 févr. 2018) | 1 ligne

Creating initial repository structure

---

3 Recette IR

Étudiant 3 : Reynier Tony

- La création d'un nouveau spectacle est possible
- La création d'une scène est opérationnelle
- L'exécution d'une scène est fonctionnelle
- La création d'une séquence est opérationnelle
- L'exécution d'une séquence est fonctionnelle
- L'ajout d'un projecteur est réalisable à partir de l'IHM
- Les fichiers XML existent et sont correctement renseignés

Étudiant 4 : Demont Thomas

- L'interface de communication est paramétrée et fonctionnelle
- Le fichier adaptateurs.xml existe et correctement renseigné
- La commande d'un projecteur est possible à partir de l'IHM
- Le fichier consoles.xml existe et correctement renseigné
- Une communication avec la console est possible
- La commande d'un projecteur est possible à partir de la console

4 Fichiers XML

Étudiant 4 : Demont Thomas

- consoles.xml

A faire définir le fichier consoles.xml

- adpatateurs.xml

< ?xml version="1.0" encoding="UTF-8" ?>

<interfaces>

<interface utilisee="0" id="1">

<port>/dev/dmx</port>

<type>1</type>

</interface>

<interface utilisee="1" id="2">

<port>/dev/ttyUSB0</port>

<type>0</type>

</interface>

<interface utilisee="0" id="3">

<port>/dev/ttyUSB1</port>

<type>1</type>

</interface>

</interfaces>

Étudiant 3 : Reynier Tony

– appareils.xml

< ?xml version='1.0' encoding='UTF-8' ?>

<appareils nbappareils="4">

<appareil nom="Par LED 1" nbcanal="4" type="PAR LED" uuid="{f163b05a-58bf-46d9-887b-a4e9a0ad8f11}">

<canal id="1">RED</canal>

<canal id="2">GREEN</canal>

<canal id="3">BLUE</canal>

<canal id="4">DIMSTRO</canal>

</appareil>

<appareil nom="Scanner 1" nbcanal="3" type="SCANNER" uuid="{83fe644d-acbb-4b31-a9ee-06882b204556}">

<canal id="32">PAN</canal>

<canal id="33">TILT</canal>

<canal id="34">GLOBOS</canal>

</appareil>

<appareil nom="Lyre au sol" nbcanal="5" type="LYRE" uuid="{16dc3967-5ea3-42a4-a29d-b16131eb62c4}">

<canal id="132">PAN</canal>

<canal id="133">TILT</canal>

<canal id="129">GLOBOS</canal>

<canal id="131">COULEUR</canal>

<canal id="128">STROBE</canal>

</appareil>

<appareil nom="Par LED 2" nbcanal="3" type="PAR LED" uuid="{be786ddb-9af0-

4b94-bf2c-7b175b9e56d6}">

<canal id="10">RED</canal>

<canal id="11">GREEN</canal>

<canal id="12">BLUE</canal>

</appareil>

</appareils>

– scenes.xml

< ?xml version='1.0' encoding='UTF-8' ?>

<scenes>

<scene nom="Scene Intro">

<appareil uuid="{f163b05a-58bf-46d9-887b-a4e9a0ad8f11}">

<canal id="1">127</canal>

<canal id="2">127</canal>

<canal id="3">127</canal>

<canal id="4">0</canal>

</appareil>

<appareil uuid="{be786ddb-9af0-4b94-bf2c-7b175b9e56d6}">

<canal id="10">255</canal>

<canal id="11">255</canal>

<canal id="12">255</canal>

</appareil>

</scene>

<scene nom="Scene Personnage">

<appareil uuid="{83fe644d-acbb-4b31-a9ee-06882b204556}">

<canal id="32">127</canal>

<canal id="33">0</canal>

<canal id="34">127</canal>

</appareil>

</scene>

</scenes>

– sequences.xml

< ?xml version='1.0' encoding='UTF-8' ?>

<sequences nbsequences="2">

<sequence nom="Sequence Test" nbscenes="3">

<scene nom="Scene Test" temps="10">

<scene nom="Scene Test Bis" temps="5">

<scene nom="Scene Test" temps="30">

</sequence>

<sequence nom="Sequence Test Bis" nbscenes="2">

<scene nom="Scene Test Bis" temps="15">

<scene nom="Scene Test" temps="5">

</sequence>

</sequences>

– spectacles.xml

< ?xml version='1.0' encoding='UTF-8' ?>

<spectacles nbspectacles="1">

<spectacle nom="Spectacle Test" nbsequences="2">

<sequence nom="Sequence Test">

<sequence nom="Sequence Test Bis">

</spectacle>

</spectacles>

5 A propos

Auteur

Demont Thomas <thomasdemont2@gmail.com>
Reynier Tony <tonyreynier@gmail.com>

Version

1.1

Date

2018

6 Licence GPL

PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program ; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

7 Liste des choses à faire

Membre DMXProjecteur : :setNom (QString nom)
définir les accesseurs set

Page Fichiers XML
définir le fichier consoles.xml

8 Documentation des classes

8.1 Référence de la structure Console

Structure comprenant les diférents paramètre d'une interface.

`#include <console.h>`

Attributs publics

– QString adresseIP
– quint16 port

8.1.1 Description détaillée

Version

1.0

8.1.2 Documentation des données membres

8.1.2.1 QString Console : :adresseIP

Référencé par PlaybackWing : :ajouterConsoleListe(), IHM : :envoyerModification-Console(), et XMLUtilitaire : :lireConsoles().

8.1.2.2 quint16 Console : :port

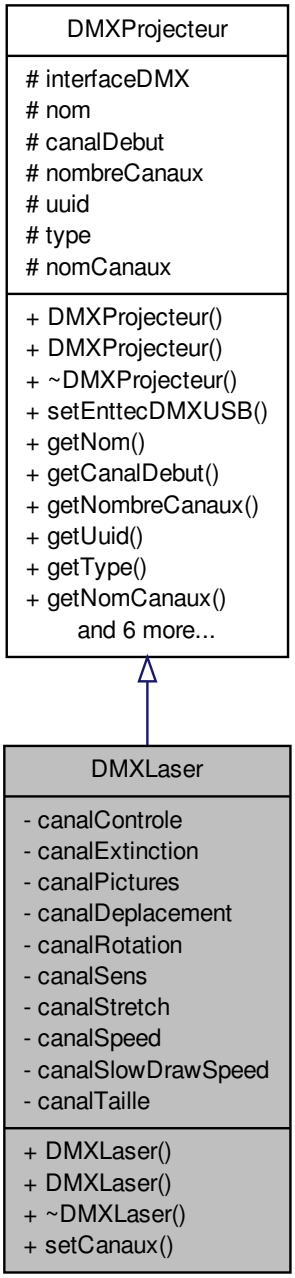
Référencé par PlaybackWing : :ajouterConsoleListe(), IHM : :envoyerModification-Console(), et XMLUtilitaire : :lireConsoles().

La documentation de cette structure a été générée à partir du fichier suivant :
– console.h

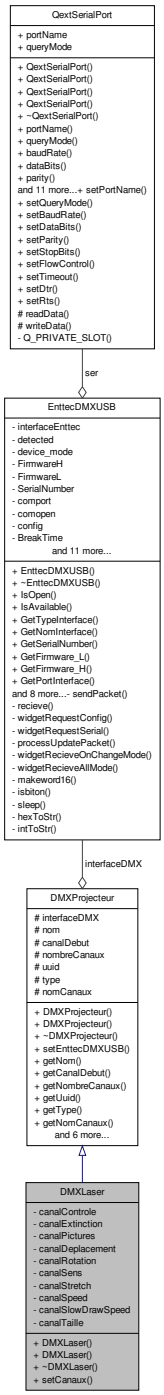
8.2 Référence de la classe DMXLaser

```
#include <DMXLaser.h>
```

Graphe d'héritage de DMXLaser :



Graphe de collaboration de DMXLaser :



Fonctions membres publiques

- DMXLaser ()
- DMXLaser (QString nom, int canalDebut, int nombreCanaux, QString uuid, QString type="LASER")
- virtual ~DMXLaser ()
- void setCanaux (int controle, int extinction, int pictures, int deplacement, int rotation, int sens, int stretch, int speed, int slowDrawSpeed, int taille)

Attributs privés

- int canalControle
- int canalExtinction
- int canalPictures
- int canalDeplacement
- int canalRotation
- int canalSens
- int canalStretch
- int canalSpeed
- int canalSlowDrawSpeed
- int canalTaille

8.2.1 Documentation des constructeurs et destructeur

8.2.1.1 DMXLaser : :DMXLaser ( )

```
                    : DMXProjecteur()
{

}
```

8.2.1.2 DMXLaser : :DMXLaser (QString *nom,* int *canalDebut,* int *nombreCanaux,* QString *uuid,* QString *type* = "LASER")

```
                    : DMXProjecteur(nom, canalDebut, nombreCanaux, uuid, type)
{

}
```

8.2.1.3 DMXLaser : :~DMXLaser ( ) [virtual]

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.2.2 Documentation des fonctions membres

8.2.2.1 void DMXLaser : :setCanaux (int *controle,* int *extinction,* int *pictures,* int *deplacement,* int *rotation,* int *sens,* int *stretch,* int *speed,* int *slowDrawSpeed,* int *taille*)

Références canalControle, canalDeplacement, canalExtinction, canalPictures, canalRotation, canalSens, canalSlowDrawSpeed, canalSpeed, canalStretch, et canalTaille.

```
{
    canalControle = controle;
    canalExtinction = extinction;
    canalPictures = pictures;
    canalDeplacement = deplacement;
    canalRotation = rotation;
    canalSens = sens;
    canalStretch = stretch;
    canalSpeed = speed;
    canalSlowDrawSpeed = slowDrawSpeed;
    canalTaille = taille;
}
```

8.2.3 Documentation des données membres

8.2.3.1 int DMXLaser : :canalControle [private]

Référencé par setCanaux().

8.2.3.2 int DMXLaser : :canalDeplacement [private]

Référencé par setCanaux().

8.2.3.3 int DMXLaser : :canalExtinction [private]

Référencé par setCanaux().

8.2.3.4 int DMXLaser : :canalPictures [private]

Référencé par setCanaux().

8.2.3.5 int DMXLaser : :canalRotation [private]

Référencé par setCanaux().

8.2.3.6 int DMXLaser : :canalSens [private]

Référencé par setCanaux().

8.2.3.7 int DMXLaser : :canalSlowDrawSpeed [private]

Référencé par setCanaux().

8.2.3.8 int DMXLaser : :canalSpeed [private]

Référencé par setCanaux().

8.2.3.9 int DMXLaser : :canalStretch [private]

Référencé par setCanaux().

8.2.3.10 int DMXLaser : :canalTaille [private]

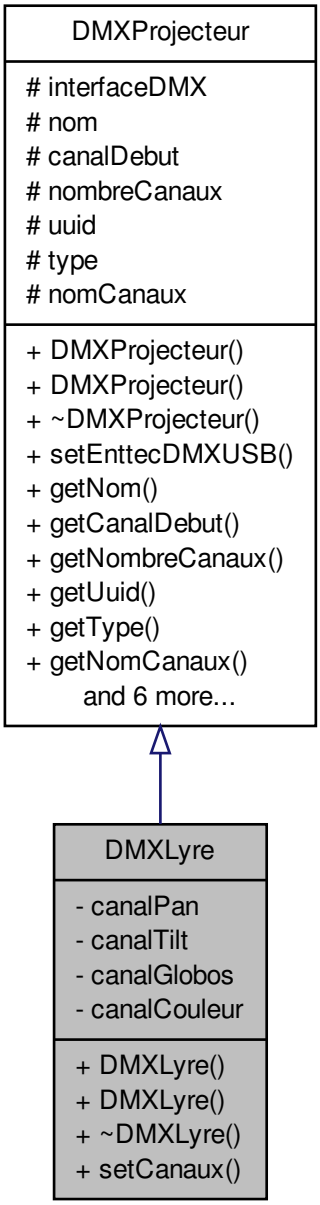
Référencé par setCanaux().

La documentation de cette classe a été générée à partir des fichiers suivants :
– DMXLaser.h
– DMXLaser.cpp

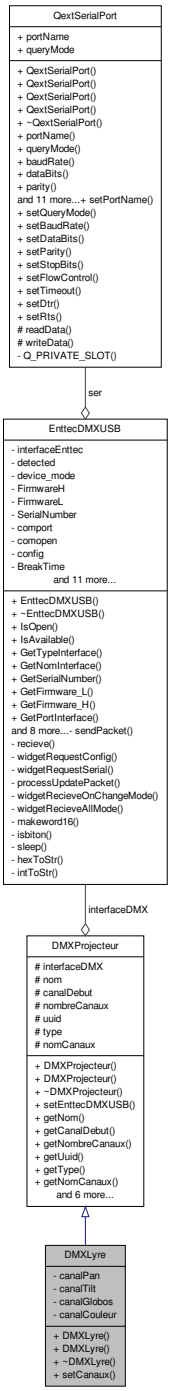
8.3 Référence de la classe DMXLyre

```
#include <DMXLyre.h>
```

Graphe d'héritage de DMXLyre :



Graphe de collaboration de DMXLyre :



Fonctions membres publiques

- DMXLyre ()
- DMXLyre (QString nom, int canalDebut, int nombreCanaux, QString uuid, QString type="LYRE")
- virtual ~DMXLyre ()
- void setCanaux (int pan, int tilt, int globos, int couleur)

Attributs privés

- int canalPan
- int canalTilt
- int canalGlobos
- int canalCouleur

8.3.1 Documentation des constructeurs et destructeur

8.3.1.1 DMXLyre : :DMXLyre ( )

```
{

}
```

8.3.1.2 DMXLyre : :DMXLyre (QString *nom,* int *canalDebut,* int *nombreCanaux,* QString *uuid,* QString *type* = "LYRE")

```
                : DMXProjecteur(nom, canalDebut, nombreCanaux, uuid, type)
{

}
```

8.3.1.3 DMXLyre : :~DMXLyre ( ) [virtual]

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.3.2 Documentation des fonctions membres

8.3.2.1 void DMXLyre : :setCanaux (int *pan,* int *tilt,* int *globos,* int *couleur*)

Références canalCouleur, canalGlobos, canalPan, et canalTilt.

```
{
    canalPan = pan;
    canalTilt = tilt;
    canalGlobos = globos;
    canalCouleur = couleur;
}
```

8.3.3 Documentation des données membres

8.3.3.1 int DMXLyre : :canalCouleur [private]

Référencé par setCanaux().

8.3.3.2 int DMXLyre : :canalGlobos [private]

Référencé par setCanaux().

8.3.3.3 int DMXLyre : :canalPan [private]

Référencé par setCanaux().

8.3.3.4 int DMXLyre : :canalTilt [private]

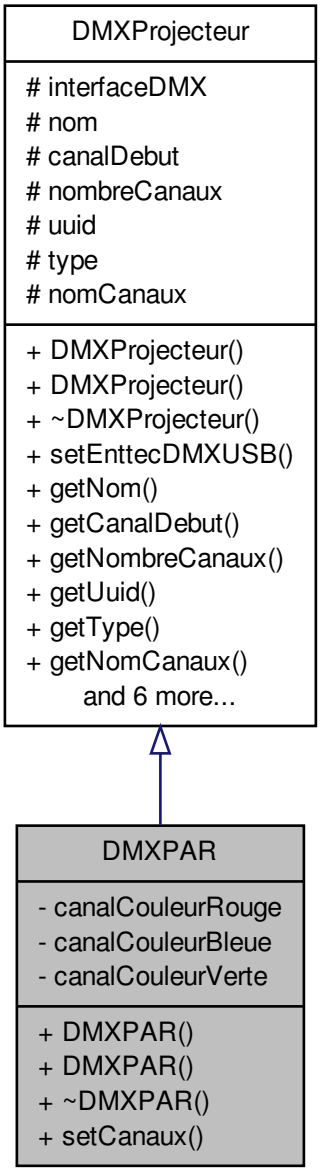
Référencé par setCanaux().

La documentation de cette classe a été générée à partir des fichiers suivants :
– DMXLyre.h
– DMXLyre.cpp

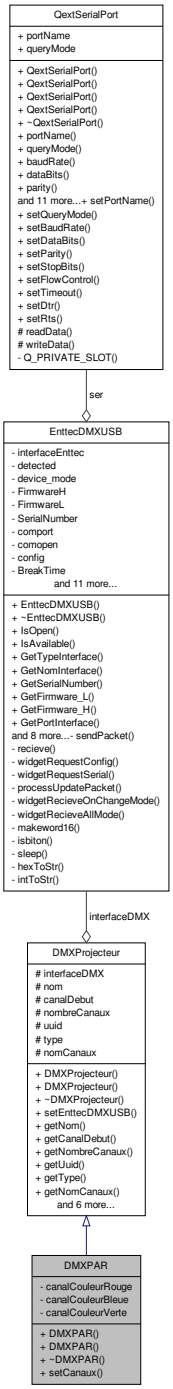
8.4 Référence de la classe DMXPAR

```
#include <DMXPAR.h>
```

Graphe d'héritage de DMXPAR :



Graphe de collaboration de DMXPAR :



Fonctions membres publiques

- DMXPAR ()
- DMXPAR (QString nom, int canalDebut, int nombreCanaux, QString uuid, QString type="PAR LED")
- virtual ~DMXPAR ()
- void setCanaux (int rouge, int bleu, int vert)

Attributs privés

- int canalCouleurRouge
- int canalCouleurBleue
- int canalCouleurVerte

8.4.1 Documentation des constructeurs et destructeur

8.4.1.1 DMXPAR : :DMXPAR ( )

```
{

}
```

8.4.1.2 DMXPAR : :DMXPAR (QString *nom,* int *canalDebut,* int *nombreCanaux,* QString *uuid,* QString *type* = "PAR LED")

```
                : DMXProjecteur(nom, canalDebut, nombreCanaux, uuid, type)
{

}
```

8.4.1.3 DMXPAR : :~DMXPAR ( ) [virtual]

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.4.2 Documentation des fonctions membres

8.4.2.1 void DMXPAR : :setCanaux (int *rouge,* int *bleu,* int *vert*)

Références canalCouleurBleue, canalCouleurRouge, et canalCouleurVerte.

```
{
    canalCouleurRouge = rouge;
    canalCouleurBleue = bleu;
    canalCouleurVerte = vert;
}
```

8.4.3 Documentation des données membres

8.4.3.1 int DMXPAR : :canalCouleurBleue [private]

Référencé par setCanaux().

8.4.3.2 int DMXPAR : :canalCouleurRouge [private]

Référencé par setCanaux().

8.4.3.3 int DMXPAR : :canalCouleurVerte [private]

Référencé par setCanaux().

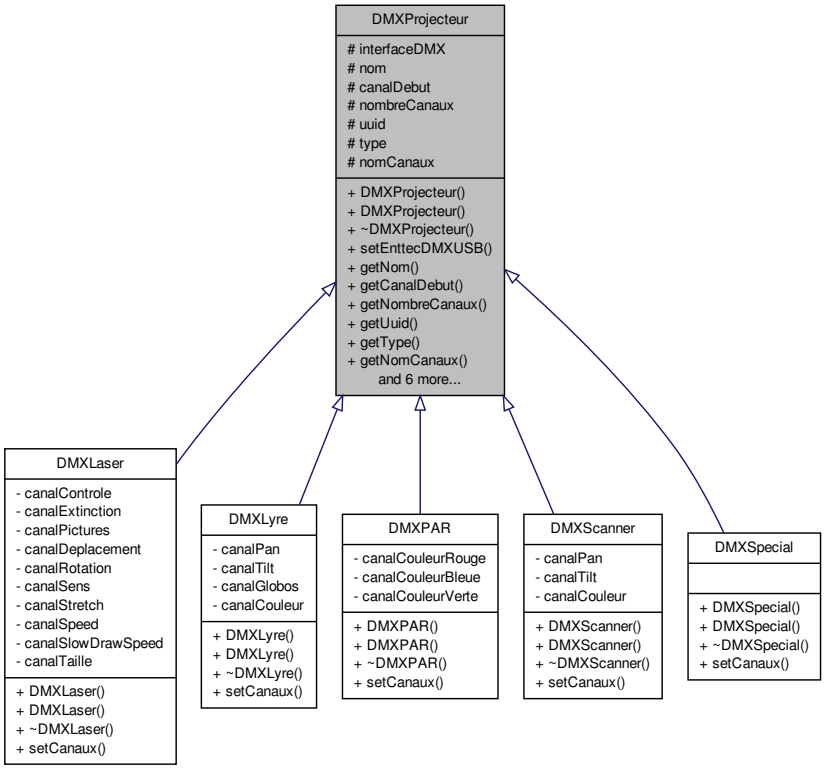
La documentation de cette classe a été générée à partir des fichiers suivants :
– DMXPAR.h
– DMXPAR.cpp

8.5 Référence de la classe DMXProjecteur

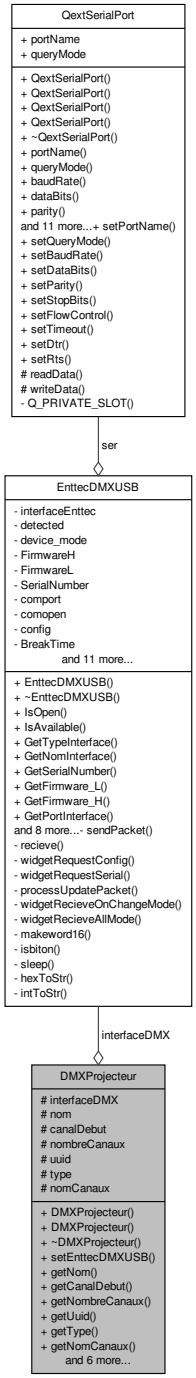
Definition de la classe DMXProjecteur.

`#include <DMXProjecteur.h>`

Graphe d'héritage de DMXProjecteur :



Graphe de collaboration de DMXProjecteur :



Fonctions membres publiques

- DMXProjecteur ()
- DMXProjecteur (QString nom, int canalDebut, int nombreCanaux, QString uuid, QString type="")
- virtual ~DMXProjecteur ()
- void setEnttecDMXUSB (EnttecDMXUSB *interfaceDMX)
  Mutateur de l'attribut interfaceDMX de la classe DMXProjecteur.
- QString getNom () const
  Accesseur de l'attribut nom de la classe DMXProjecteur.
- int getCanalDebut () const
  Accesseur de l'attribut canalDebut de la classe DMXProjecteur.
- int getNombreCanaux () const
  Accesseur de l'attribut nombreCanaux de la classe DMXProjecteur.
- QString getUuid () const
  Accesseur de l'attribut uuid de la classe DMXProjecteur.
- QString getType () const
  Accesseur de l'attribut type de la classe DMXProjecteur.
- QStringList getNomCanaux () const
  Accesseur de l'attribut nomCanaux[] de la classe DMXProjecteur.
- QString getNomCanal (int canal) const
  Accesseur de la case canal-canalDebut de l'attribut nomCanaux[] de la classe DMXProjecteur.
- void setCanalDebut (int debut)
  Mutateur de l'attribut canalDebut de la classe DMXProjecteur.
- void setNomCanaux (QStringList nomCanaux)
  Mutateur de l'attribut nomCanaux[] de la classe DMXProjecteur.
- void setNom (QString nom)
  Mutateur de l'attribut nom de la classe DMXProjecteur.
- void setNombreCanaux (int nombre)
  Mutateur de l'attribut nombreCanaux de la classe DMXProjecteur.
- void setUuid (QString uuid)
  Mutateur de l'attribut uuid de la classe DMXProjecteur.
- void setType (QString type)
  Mutateur de l'attribut type de la classe DMXProjecteur.

Attributs protégés

- EnttecDMXUSB * interfaceDMX
  Association vers la classe EnttecDMXUSB.
- QString nom
- int canalDebut
- int nombreCanaux
- QString uuid
- QString type
- QStringList nomCanaux

8.5.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

8.5.2 Documentation des constructeurs et destructeur

8.5.2.1 DMXProjecteur : :DMXProjecteur ( )

```
                              : interfaceDMX(NULL), nom(""), canalDebut(1),
      nombreCanaux(0), uuid(""), type("")
{
}
```

8.5.2.2 DMXProjecteur : :DMXProjecteur (QString *nom*, int *canalDebut*, int *nombreCanaux*, QString *uuid*, QString *type* = " ")

```
                              : interfaceDMX(NULL), nom(nom), canalDebut(canalDebut),
      nombreCanaux(nombreCanaux), uuid(uuid), type(type)
{

}
```

8.5.2.3 DMXProjecteur : :~DMXProjecteur ( ) [virtual]

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.5.3 Documentation des fonctions membres

8.5.3.1 int DMXProjecteur : :getCanalDebut ( ) const

Renvoie

int

Références canalDebut.

Référencé par IHM : :genererFenetreCanauxScene(), IHM : :ouvrirFenetreInformations-Projecteur(), et IHM : :selectionnerProjecteursPilotage().

```
{
    return canalDebut;
}
```

8.5.3.2 QString DMXProjecteur : :getNom ( ) const

Renvoie

QString

Références nom.

```
{
    return nom;
}
```

8.5.3.3 int DMXProjecteur : :getNombreCanaux ( ) const

Renvoie

int

Références nombreCanaux.

```
{
    return nombreCanaux;
}
```

8.5.3.4 QString DMXProjecteur : :getNomCanal (int *canal*) const

Paramètres

canal	

Renvoie

QString

Références canalDebut, et nomCanaux.

Référencé par IHM : :genererFenetreCanauxScene(), et IHM : :selectionnerProjecteurs-Pilotage().

```
{
    return nomCanaux.at(canal - canalDebut);
}
```

8.5.3.5 QStringList DMXProjecteur : :getNomCanaux ( ) const

Renvoie

QStringList

Références nomCanaux.

Référencé par IHM : :genererFenetreCanauxScene(), IHM : :ouvrirFenetreInformations-Projecteur(), IHM : :ouvrirFenetreModifierProjecteur(), et IHM : :selectionner-ProjecteursPilotage().

```
{
    return nomCanaux;
}
```

8.5.3.6 QString DMXProjecteur : :getType ( ) const

Renvoie

QString

Références type.

```
{
    return type;
}
```

8.5.3.7 QString DMXProjecteur : :getUuid ( ) const

Renvoie

QString

Références uuid.

```
{
    return uuid;
}
```

8.5.3.8 void DMXProjecteur : :setCanalDebut (int *debut*)

Paramètres

debut	

Renvoie

void

Références canalDebut.

Référencé par XMLUtilitaire : :lireAppareils().

```
{
    this->canalDebut = debut;
}
```

8.5.3.9 void DMXProjecteur : :setEnttecDMXUSB (EnttecDMXUSB * *interfaceDMX*)

Paramètres

*interfaceDMX	

Renvoie

void

Références interfaceDMX.

```
{
    this->interfaceDMX = interfaceDMX;
}
```

8.5.3.10 void DMXProjecteur : :setNom (QString *nom*)

A faire définir les accesseurs set

Paramètres

nom	

Renvoie

void

Références nom.

```
{
    this->nom = nom;
}
```

8.5.3.11 void DMXProjecteur : :setNombreCanaux (int *nombre*)

Paramètres

nombre	

Renvoie

void

Références nombreCanaux.

```
{
    this->nombreCanaux = nombre;
}
```

8.5.3.12 void DMXProjecteur : :setNomCanaux (QStringList *nomCanaux*)

Paramètres

nomCanaux	

Renvoie

void

Références nomCanaux.

Référencé par XMLUtilitaire : :lireAppareils().

```
{
    this->nomCanaux = nomCanaux;
}
```

8.5.3.13 void DMXProjecteur : :setType (QString *type*)

Paramètres

type	

Renvoie

void

Références type.

```
{
    this->type = type;
}
```

8.5.3.14 void DMXProjecteur : :setUuid (QString *uuid*)

Paramètres

uuid	

Renvoie

void

Références uuid.

```
{
    this->uuid = uuid;
}
```

8.5.4 Documentation des données membres

8.5.4.1 int DMXProjecteur : :canalDebut [protected]

Référencé par getCanalDebut(), getNomCanal(), et setCanalDebut().

8.5.4.2 EnttecDMXUSB* DMXProjecteur : :interfaceDMX [protected]

Référencé par setEnttecDMXUSB().

8.5.4.3 QString DMXProjecteur : :nom [protected]

Référencé par getNom(), et setNom().

8.5.4.4 int DMXProjecteur : :nombreCanaux [protected]

Référencé par getNombreCanaux(), et setNombreCanaux().

8.5.4.5 QStringList DMXProjecteur : :nomCanaux [protected]

Référencé par getNomCanal(), getNomCanaux(), et setNomCanaux().

8.5.4.6 QString DMXProjecteur : :type [protected]

Référencé par getType(), et setType().

8.5.4.7 QString DMXProjecteur : :uuid [protected]

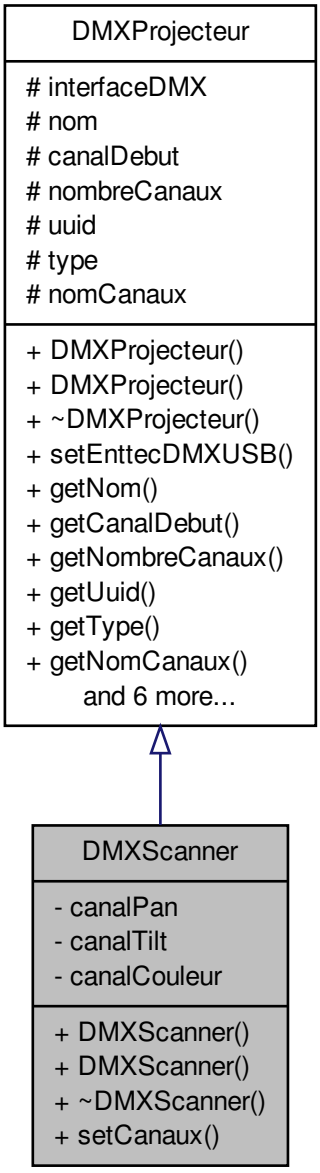
Référencé par getUuid(), et setUuid().

La documentation de cette classe a été générée à partir des fichiers suivants :
- DMXProjecteur.h
- DMXProjecteur.cpp

8.6 Référence de la classe DMXScanner

`#include <DMXScanner.h>`

Graphe d'héritage de DMXScanner :



Graphe de collaboration de DMXScanner :

Fonctions membres publiques

- DMXScanner ()
- DMXScanner (QString nom, int canalDebut, int nombreCanaux, QString uuid, QString type="SCANNER")
- virtual ~DMXScanner ()
- void setCanaux (int pan, int tilt, int couleur)

Attributs privés

- int canalPan
- int canalTilt
- int canalCouleur

8.6.1 Documentation des constructeurs et destructeur

8.6.1.1 DMXScanner : :DMXScanner ( )

```
{

}
```

8.6.1.2 DMXScanner : :DMXScanner (QString *nom,* int *canalDebut,* int *nombreCanaux,* QString *uuid,* QString *type* = "SCANNER")

```
                        : DMXProjecteur(nom, canalDebut, nombreCanaux, uuid, type)
{

}
```

8.6.1.3 DMXScanner : :~DMXScanner ( ) [virtual]

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.6.2 Documentation des fonctions membres

8.6.2.1 void DMXScanner : :setCanaux (int *pan,* int *tilt,* int *couleur*)

Références canalCouleur, canalPan, et canalTilt.

```
{
    canalPan = pan;
    canalTilt = tilt;
    canalCouleur = couleur;
}
```

8.6.3 Documentation des données membres

8.6.3.1 int DMXScanner : :canalCouleur [private]

Référencé par setCanaux().

8.6.3.2 int DMXScanner : :canalPan [private]

Référencé par setCanaux().

8.6.3.3 int DMXScanner : :canalTilt [private]

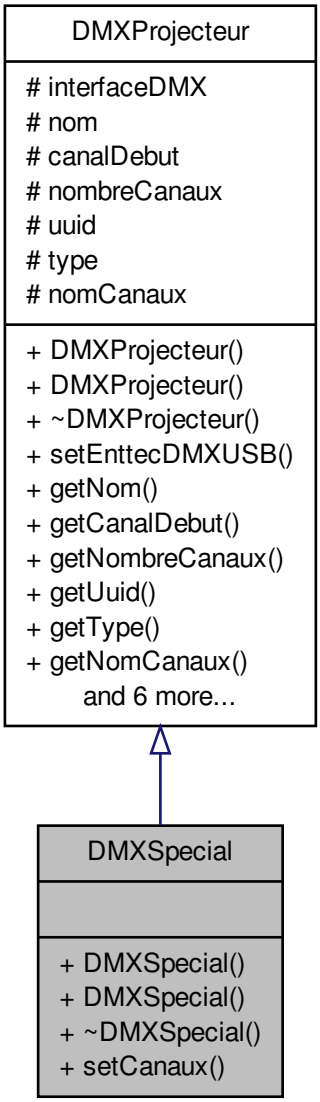
Référencé par setCanaux().

La documentation de cette classe a été générée à partir des fichiers suivants :
– DMXScanner.h
– DMXScanner.cpp

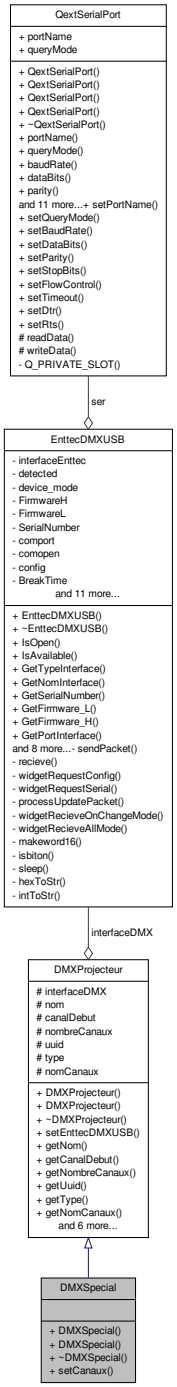
8.7 Référence de la classe DMXSpecial

```
#include <DMXSpecial.h>
```

Graphe d'héritage de DMXSpecial :



Graphe de collaboration de DMXSpecial :



Fonctions membres publiques

- DMXSpecial ()
- DMXSpecial (QString nom, int canalDebut, int nombreCanaux, QString uuid, QString type="SPECIAL")
- virtual ~DMXSpecial ()
- void setCanaux ()

8.7.1 Documentation des constructeurs et destructeur

8.7.1.1 DMXSpecial : :DMXSpecial ( )

```
{

}
```

8.7.1.2 DMXSpecial : :DMXSpecial (QString *nom,* int *canalDebut,* int *nombreCanaux,* QString *uuid,* QString *type* = "SPECIAL")

```
                    : DMXProjecteur(nom, canalDebut, nombreCanaux, uuid, type)
{

}
```

8.7.1.3 DMXSpecial : :~DMXSpecial ( ) [virtual]

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.7.2 Documentation des fonctions membres

8.7.2.1 void DMXSpecial : :setCanaux ( )

```
{

}
```

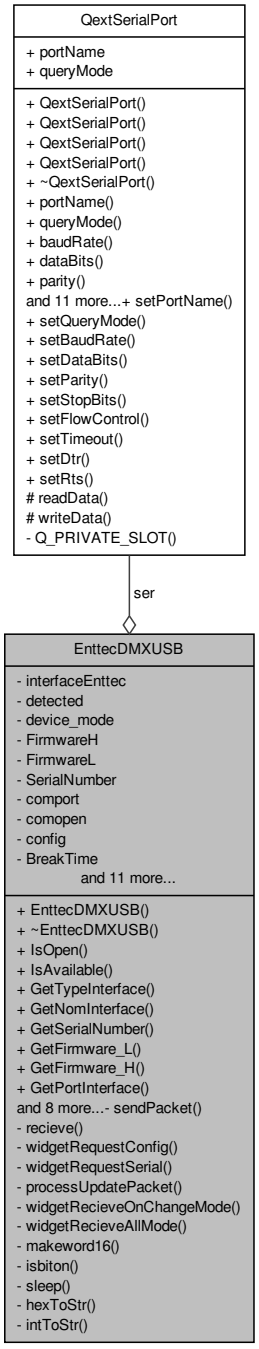
La documentation de cette classe a été générée à partir des fichiers suivants :
- DMXSpecial.h
- DMXSpecial.cpp

8.8 Référence de la classe EnttecDMXUSB

Classe permettant la communication avec l'interface Enttec.

`#include <enttecdmxusb.h>`

Graphe de collaboration de EnttecDMXUSB :



Fonctions membres publiques

- EnttecDMXUSB (EnttecInterfaces typeInterface=DMX_USB_PRO, string portInterface="/dev/ttyUSB0")
- ~EnttecDMXUSB ()
- bool IsOpen ()
- bool IsAvailable ()
- EnttecInterfaces GetTypeInterface ()
- string GetNomInterface ()
- string GetSerialNumber ()
- byte GetFirmware_L ()
- byte GetFirmware_H ()
- string GetPortInterface ()
- string GetConfiguration ()
- bool SetCanalDMX (int canal, byte valeur)
- bool SetNbCanauxDMX (int start=1, int length=NB_CANAUX_MAX)
- bool ResetCanauxDMX (int start=1, int end=NB_CANAUX_MAX)
- void SendDMX ()
- bool SendDatasDMX (byte *datas, int start=1, int length=NB_CANAUX_MAX)
- void DisplayConfig ()
- void closePort ()
- bool openPort (string port_use)

Fonctions membres privées

- int sendPacket (int pkt_type, char *data, int length)
- bool recieve ()
- void widgetRequestConfig ()
- void widgetRequestSerial ()
- void processUpdatePacket ()
- void widgetRecieveOnChangeMode ()
- void widgetRecieveAllMode ()
- int makeword16 (byte lsb, byte msb)
- bool isbiton (int value, byte bit)
- void sleep (int usec)
- void hexToStr (int i, int nb, char *s)
- void intToStr (int i, char *s)

Attributs privés

- EnttecInterfaces interfaceEnttec
- bool detected
- byte device_mode
- byte FirmwareH
- byte FirmwareL
- string SerialNumber
- string comport
- bool comopen
- bool config
- byte BreakTime
- byte MABTime
- byte FrameRate
- char buffer [4096]
- QextSerialPort ser
- TDmxArray dmxout
- int dmxout_length
- bool dmx_available
- TDmxArray dmxin
- int dmxin_length
- int dmxin_quality
- byte dmxin_mode
- bool dmxin_filter

8.8.1 Description détaillée

Auteur

Thierry Vaira

8.8.2 Documentation des constructeurs et destructeur

8.8.2.1 EnttecDMXUSB : :EnttecDMXUSB (EnttecInterfaces *typeInterface* = DMX_USB_PRO, string *portInterface* = "/dev/ttyUSB0")

Références BreakTime, buffer, comopen, comport, config, detected, dmx_available, dmxin, dmxin_filter, dmxin_length, dmxin_quality, dmxout, dmxout_length, FirmwareH, FirmwareL, FrameRate, interfaceEnttec, MABTime, NB_CANAUX_MAX, openPort(), et SerialNumber.

```
{
    interfaceEnttec = typeInterface;
    comport = portInterface;
    detected = false;
    FirmwareH = 0;
    FirmwareL = 0;/* Firmware Version */
    SerialNumber = "";/* Serial Number */
    comopen = false;
    config = false;
    BreakTime = 0;
    MABTime = 0;
    FrameRate = 0; /* tranmission frame rate 1..40 */

    memset(buffer, 0x00, 4096);

    memset(dmxout, 0x00, NB_CANAUX_MAX+1);//+ le startcode
    memset(dmxin, 0x00, NB_CANAUX_MAX+1);//+ le startcode
    dmxout_length = NB_CANAUX_MAX+1;//+ le startcode
    dmx_available = false;
    dmxin_length = 0;
    dmxin_quality = -1;
    dmxin_filter = false;

    openPort(portInterface);
}
```

8.8.2.2 EnttecDMXUSB : :~EnttecDMXUSB ( )

Références closePort().

```
{
    closePort();
}
```

8.8.3 Documentation des fonctions membres

8.8.3.1 void EnttecDMXUSB : :closePort ( )

Références QextSerialPort : :close(), comopen, detected, et ser.

Référencé par IHM : :detecterInterface(), openPort(), et ~EnttecDMXUSB().

```
{
     detected = false;
     if (comopen)
     {
         ser.close();
         //ser.CloseSerial();
         comopen = false;
     }
}
```

8.8.3.2 void EnttecDMXUSB : :DisplayConfig ( )

Références BreakTime, config, DMX_USB_PRO, FirmwareH, FirmwareL, FrameRate, interfaceEnttec, MABTime, et SerialNumber.

Référencé par openPort().

```
{
    if(interfaceEnttec == DMX_USB_PRO)
    {
        if(config == true)
        {
            printf("Interface DMX USB PRO\n");
            printf("Firmware: %d.%d\n", FirmwareH, FirmwareL);
            printf("BreakTime: %.2f ms\n", (BreakTime*10.67));
            printf("MABTime: %.2f ms\n", (MABTime*10.67));
            printf("Frames Per Second: %d\n", FrameRate);
            if (FrameRate > 40) printf("warning: invalid frame rate\n");
            else if (FrameRate < 1) printf("warning: invalid frame rate\n");
            if(SerialNumber.length() != 0)
                printf("Serial Number: %s\n", SerialNumber.c_str());
        }
        else printf("Interface DMX USB PRO : no data config for this interface
       !\n");
    }
    else printf("Interface OPEN DMX USB : no data config for this interface !\n
      ");
}
```

8.8.3.3 string EnttecDMXUSB : :GetConfiguration ( )

Références BreakTime, config, DMX_USB_PRO, FirmwareH, FirmwareL, FrameRate, interfaceEnttec, MABTime, recieve(), SerialNumber, sleep(), widgetRequestConfig(), et widgetRequestSerial().

```
{
    char c[256] = "";
    string configuration;

    if(interfaceEnttec == DMX_USB_PRO)
    {
        widgetRequestConfig();
        this->sleep(100);
        recieve();
        widgetRequestSerial();
        this->sleep(100);
        recieve();

        if(config == true)
        {
            sprintf(c, "Firmware: %d.%d\n", FirmwareH, FirmwareL);
            configuration += string(c);
            sprintf(c, "BreakTime: %.2f ms\n", (BreakTime*10.67));
            configuration += string(c);
            sprintf(c, "MABTime: %.2f ms\n", (MABTime*10.67));
```

```
            configuration += string(c);
            sprintf(c, "Frames Per Second: %d\n", FrameRate);
            configuration += string(c);
            if(SerialNumber.length() != 0)
            {
                sprintf(c, "Serial Number: %s\n", SerialNumber.c_str());
                configuration += string(c);
            }
        }
    }

    return configuration;
}
```

8.8.3.4 byte EnttecDMXUSB : :GetFirmware_H ( )

Référencé par IHM : :detecterInterface().

```
{ return FirmwareH; }
```

8.8.3.5 byte EnttecDMXUSB : :GetFirmware_L ( )

Référencé par IHM : :detecterInterface().

```
{ return FirmwareL; }
```

8.8.3.6 string EnttecDMXUSB : :GetNomInterface ( )

Références nomInterfaces.

```
{ return string(nomInterfaces[interfaceEnttec]); }
```

8.8.3.7 string EnttecDMXUSB : :GetPortInterface ( )

Référencé par IHM : :detecterInterface().

```
{ return comport; }
```

8.8.3.8 string EnttecDMXUSB : :GetSerialNumber ( )

```
{ return SerialNumber; }
```

8.8.3.9 EnttecInterfaces EnttecDMXUSB : :GetTypeInterface ( )

```
{ return interfaceEnttec; }
```

8.8.3.10 void EnttecDMXUSB : :hexToStr (int *i*, int *nb*, char * *s*) [private]

Référencé par recieve().

```
{
   int k = nb-1, l;
   unsigned int num = (unsigned int)i;

   for(;k>=0;k--)
   {
      l=(num>>(k*4)) & 0xF;
      if(l>9) l+=7;
      l+='0';
      *(s++)=l;
   }
   *s=0;
}
```

8.8.3.11 void EnttecDMXUSB : :intToStr (int *i,* char * *s*) [private]

```
{
   int k;

   if(i<0)
   {
      *(s++)='-';
      i=-i;
   }
   for(k=1;(i/k)>9;k=k*10);
   while(k)
   {
      *(s++)='0' + (i/k);
      i-=(i/k)*k;
      k/=10;
   }
   *s=0;
}
```

8.8.3.12 bool EnttecDMXUSB : :IsAvailable ( )

Référencé par IHM : :detecterInterface().

```
{ return detected; }
```

8.8.3.13 bool EnttecDMXUSB : :isbiton (int *value,* byte *bit*) [private]

Référencé par processUpdatePacket().

```
{
    int result = (value && (1 << bit));
    if(result != 0) return true;
    return false;
}
```

8.8.3.14 bool EnttecDMXUSB : :IsOpen ( )

Référencé par IHM : :detecterInterface(), et IHM : :selectionnerInterface().

```
{ return comopen; }
```

8.8.3.15 int EnttecDMXUSB : :makeword16 (byte *lsb,* byte *msb*) [private]

Référencé par recieve().

```
{
    int newnum;
    newnum = msb; newnum = newnum << 8; newnum = newnum + lsb;
    return(newnum);
}
```

8.8.3.16 bool EnttecDMXUSB : :openPort (string *port_use*)

Références BAUD9600, closePort(), comopen, comport, config, DATA_8, detected, device_mode, DisplayConfig(), DMX_USB_PRO, dmxin_filter, dmxin_length, dmxout_length, QextSerialPort : :EventDriven, FLOW_OFF, QextSerialPort : :flush(), interfaceEnttec, NB_CANAUX_MAX, QextSerialPort : :open(), OPEN_DMX_USB, PAR_NONE, PKT_DMXIN, recieve(), SendDMX(), ser, QextSerialPort : :setBaudRate(), QextSerialPort : :setDataBits(), QextSerialPort : :setFlowControl(), QextSerialPort : :setParity(), QextSerialPort : :setPortName(), QextSerialPort : :setQueryMode(), QextSerialPort : :setRts(), QextSerialPort : :setStopBits(), sleep(), STOP_1, STOP_2, widgetRequestConfig(), et widgetRequestSerial().

Référencé par IHM : :detecterInterface(), et EnttecDMXUSB().

```
{
    //int flags;

    dmxin_filter = false;
    dmxin_length = 0;

    //pas de port ?
    if (port_use.length() == 0)
    {
        return(false);
    }

    dmxout_length = NB_CANAUX_MAX+1; //+ le startcode
    comport = port_use;
    device_mode = PKT_DMXIN; /* par dut */

    //qDebug() << Q_FUNC_INFO << QString::fromStdString(port_use);

    //ouverture du port
    if(interfaceEnttec == DMX_USB_PRO)
    {
        //flags = O_RDWR | O_NONBLOCK;
        ser.setPortName(QString::fromStdString(port_use));
        ser.setQueryMode(QextSerialPort::EventDriven);
        ser.open(QIODevice::ReadWrite);
        //if(ser.OpenSerial(port_use, flags) == -1)
        if(!ser.isOpen())
            comopen = false;
        else    comopen = true;
        config = true;
    }
    else if(interfaceEnttec == OPEN_DMX_USB)
    {
        //flags = O_RDWR | O_NONBLOCK;
        ser.setPortName(QString::fromStdString(port_use));
        ser.setQueryMode(QextSerialPort::EventDriven);
        ser.open(QIODevice::ReadWrite);
        //if(ser.OpenSerial(port_use, flags) == -1)
        if(!ser.isOpen())
            comopen = false;
        else    comopen = true;
        config = false;
    }
    else
    {
        //if(!ser.isOpen())
        //    comopen = false;
```

```
        //if(ser.OpenSerial(port_use) == -1)
            comopen = false;
    }

    //configuration du port
    if(comopen == true)
    {
        if(interfaceEnttec == DMX_USB_PRO)
        {
            //ser.setBaudRate(BAUD4000000);
            ser.setBaudRate(BAUD9600);
            ser.setDataBits(DATA_8);
            ser.setParity(PAR_NONE);
            ser.setStopBits(STOP_1);
            ser.setFlowControl(FLOW_OFF);
            //ser.SetSerialParams(4000000, 8, 'N', 1, 0); /* for DMX_USB_PRO */
        }
        else
        {
            //ser.setBaudRate(BAUD2500000);
            ser.setBaudRate(BAUD9600);
            ser.setDataBits(DATA_8);
            ser.setParity(PAR_NONE);
            ser.setStopBits(STOP_2);
            ser.setFlowControl(FLOW_OFF);
            ser.setRts(0);
            //ser.SetCustomBaudRate(250000);
            //ser.SetSerialParams(38400, 8, 'N', 2, 0); /* for OPEN_DMX_USB */
            //ser.SetSerialRTS(0);//clear RTS
        }
        detected = true;
        qDebug() << detected;
        ser.flush();
        //ser.Purge();
    }

    return(true);

    //dction de l'interface
    if(comopen == true)
    {
        if(interfaceEnttec == DMX_USB_PRO)
        {
            SendDMX(); /* set dmx out with 0 */
            this->sleep(100);
            ser.flush(); /* clear the buffer */
            //ser.Purge(); /* clear the buffer */

            widgetRequestConfig();
            this->sleep(100);
            recieve();
        }
        else detected = true; /* default for OPEN_DMX_USB */
    }

    //recupere la configuration de l'interface dctet termine
    if (detected)
    {
        if(interfaceEnttec == DMX_USB_PRO)
        {
            widgetRequestSerial();
            this->sleep(100);
            recieve();
        }
        //else /* for OPEN_DMX_USB */
        #ifdef DEBUG_DMX_USB

        std::cerr << " EnttecDMXUSB::OpenPort() " << port_use << std::endl;
        DisplayConfig();
        #endif

        return(true); /* ok */
    }
    closePort(); /* not ok */
```

```
    #ifdef DEBUG_DMX_USB

    std::cerr << " EnttecDMXUSB::OpenPort() " << port_use << " : failed !" <<
      std::endl;
    #endif


    return(false);
}
```

8.8.3.17 void EnttecDMXUSB : :processUpdatePacket ( ) [private]

Références buffer, dmxin, et isbiton().

Référencé par recieve().

```
{
  int cbi, bai, tmp, bi;
  cbi = 0; tmp = 0;
  for (bi = 0;bi<4;bi++)
  {
   for (bai = 0;bai<7;bai++)
   {
     if (isbiton(buffer[1+bi], bai))
     {
        dmxin[(buffer[0] * 8) + cbi] = buffer[6+tmp];
        #ifdef DEBUG_DMX_USB

        std::cerr << " EnttecDMXUSB::ProcessUpdatePacket() ";
        fprintf(stderr, "<RX> update set channel %d to 0x%02X\n", ((buffer[0] *
       8) + cbi), buffer[6+tmp]);
        #endif

        tmp++;
     }
     cbi++;
    }
  }
}
```

8.8.3.18 bool EnttecDMXUSB : :recieve ( ) [private]

Références BreakTime, buffer, comopen, config, detected, dmx_available, dmxin, dmxin_filter, dmxin_length, dmxin_quality, FirmwareH, FirmwareL, FrameRate, hexToStr(), MABTime, makeword16(), PKT_DMXIN, PKT_DMXIN_UPDATE, PKT_EOM, PKT_GETCFG, PKT_GETSERIAL, PKT_SOM, processUpdatePacket(), ser, et SerialNumber.

Référencé par GetConfiguration(), et openPort().

```
{
  int a; /* length of data in buffer */
  int err;
  bool d_update;
  char ts[12]; /* serial number */
  bool valid = false;
  int nb = 0;
  char c;

    if(comopen == false && detected == false)
        return false;

    while ((valid == false) && (nb < 3)) /* check for waiting data */
    {
        if(ser.waitForReadyRead(250))
```

```
 {
     if(ser.getChar(&c))
     {
         if (c != (char)PKT_SOM) /* check to see if is a valid packet */
         {
               /* not a valid packet */
               #ifdef DEBUG_DMX_USB

               fprintf(stderr, "<RX> not SOM, lost sync\n");
               #endif

               valid = false;
         }
     }
     else
     {
         err = ser.read((char *)&buffer[0],3); /* get the header data */
         switch(buffer[0])    /* determine the packet type */
         {
         case PKT_DMXIN : /* 5 */
              {
                  d_update = false;
                  dmxin_length =  makeword16(buffer[1],buffer[2]);
                  if(ser.getChar(&c))
                      dmxin_quality = c; /* next byte is data quality */
                  err = ser.read((char *)&buffer[0],dmxin_length); /* get
 the dmx data + PKT_EOM */
                  //if (err == ErrTimeout) { printf("<RX> timeout");
 return false; }

                  for (a = 0;a<dmxin_length;a++)
                      /* check if dmx is different to last frame */
                      if (dmxin[a] != buffer[a])
                      {
                          dmxin[a] = buffer[a];
                          d_update = true; /* frame is different */
                      }
                  printf("<RX> length : %d\n",dmxin_length);
                  if (dmxin_quality != 0) printf("<RX> data invalid !\n")
;
                  if (dmxin[0] != 0) printf("<RX> non-zero startcode
 recieved, possible RDM frame ?\n");
                      /* only send valid data, if frame is different from
 last */
                      if (d_update)
                      {
                          if (dmxin_filter)
                          {
                              if ((dmxin_quality = 0) && (dmxin[0] = 0))
                                  dmx_available = true; /* dmx passed
 filter */
                          }
                          else dmx_available = true; /* filter not
 enabled */
                      }
                  valid = true;
              }
             break;
         case PKT_DMXIN_UPDATE : /* 9 */ /* recieve a dmx update packet
*/
              {
                  dmxin_length =  makeword16(buffer[1],buffer[2]);
                  err=ser.read((char *)&buffer[0],dmxin_length+1);
                  //if (err == ErrTimeout) { printf("timeout"); return
 false; }

                  processUpdatePacket();
                  dmx_available = true; /* flag new dmx packet */
                  valid = true;
              }
             break;
         case PKT_GETCFG : /* 3 */ /* recieve the current config from
 the widget */
              {
```

```
                    if(ser.getChar(&c))
                        FirmwareL = c;
                    if(ser.getChar(&c))
                        FirmwareH = c;
                    if(ser.getChar(&c))
                        BreakTime = c;
                    if(ser.getChar(&c))
                        MABTime   = c;
                    if(ser.getChar(&c))
                        FrameRate = c;

                    valid = true;
                    detected = true;
                    config = true;
                }
                break;
            case PKT_GETSERIAL : /* 10 */ /* recieve serial number from
    device */
                {
                    if(ser.getChar(&c))
                        hexToStr(c, 2, &ts[0]);
                    if(ser.getChar(&c))
                        hexToStr(c, 2, &ts[2]);
                    if(ser.getChar(&c))
                        hexToStr(c, 2, &ts[4]);
                    if(ser.getChar(&c))
                        hexToStr(c, 2, &ts[6]);

                    if(ser.getChar(&c))
                        if ((byte)c == (byte)PKT_EOM)
                        {
                            SerialNumber = ts;
                            valid = true;
                        }
                }
                break;

            } /* case packet_type */
        } /* if valid packet */
      }
      else
          fprintf(stderr, "WaitingData\n");
      ++nb;
    } /* if buffer has data */

    if(nb == 3)
        return false;

     return true;
}
```

8.8.3.19 bool EnttecDMXUSB : :ResetCanauxDMX (int *start* = 1, int *end* = NB_CANAUX_MAX)

Références dmxout, et NB_CANAUX_MAX.

```
{
    int canal;

    #ifdef DEBUG_DMX_USB

    fprintf(stderr, " EnttecDMXUSB::ResetCanauxDMX() mise  de %d d\n", start,
       end);
    #endif

    /* 1 12 */
    if((start > 0 && start <= NB_CANAUX_MAX) && (end > 0 && end <= NB_CANAUX_MAX
      ))
        for(canal=start;canal<end;canal++)
```

```
            dmxout[canal] = 0x00;
    else    return false;

    return true;
}
```

8.8.3.20 bool EnttecDMXUSB : :SendDatasDMX (byte * *datas,* int *start* = 1, int *length* = NB_CANAUX_MAX)

Références device_mode, dmxout, dmxout_length, NB_CANAUX_MAX, PKT_DMXOUT, et sendPacket().

```
{
    #ifdef DEBUG_DMX_USB

    fprintf(stderr, " EnttecDMXUSB::SendDatasDMX() de %d d\n", start, length);
    for(int i=0;i<length;i++)
    {
        fprintf(stderr, " canal %d -> 0x%02X\n", i, *(datas+i));
    }
    #endif

    if(datas != NULL)
    {
        if((start > 0 && start <= NB_CANAUX_MAX && length > 0) && ((start+
      length) <= NB_CANAUX_MAX))
        {
            memcpy((&dmxout[0]+start), datas, length);
            dmxout_length = start+length;
            sendPacket(PKT_DMXOUT, (char *)&dmxout[0], dmxout_length);
            device_mode = PKT_DMXOUT;
            return true;
        }
    }
    return false;
}
```

8.8.3.21 void EnttecDMXUSB : :SendDMX ( )

Références device_mode, dmxout, dmxout_length, PKT_DMXOUT, et sendPacket().

Référencé par IHM : :envoyerTrameDMX(), IHM : :executerScene(), et openPort().

```
{
  qDebug() << Q_FUNC_INFO;
  qDebug() << dmxout[0] << "et " << dmxout_length;
  sendPacket(PKT_DMXOUT, (char *)&dmxout[0], dmxout_length);
  device_mode = PKT_DMXOUT;
}
```

8.8.3.22 int EnttecDMXUSB : :sendPacket (int *pkt_type,* char * *data,* int *length*) [private]

Références comopen, comport, detected, DMX_USB_PRO, interfaceEnttec, nomInterfaces, PKT_EOM, PKT_SOM, ser, et sleep().

Référencé par SendDatasDMX(), SendDMX(), widgetRecieveAllMode(), widgetRecieveOnChangeMode(), widgetRequestConfig(), et widgetRequestSerial().

```
{
```

```
        short int len = (short int)length;
    char c;
    int i;

    #ifdef DEBUG_DMX_USB

    fprintf(stderr, " EnttecDMXUSB::sendPacket() type : %d (%d octets)\n",
      pkt_type, length);
    if(comopen == false)
            fprintf(stderr, " Port %s non ouvert\n", comport.c_str());
    else    fprintf(stderr, " Port %s ouvert\n", comport.c_str());
    if(detected == false)
            fprintf(stderr, " Interface %s non dct       nomInterfaces[interfaceEnttec]);
    else    fprintf(stderr, " Interface %s dct\n",
       nomInterfaces[interfaceEnttec]);
    #endif
    if(comopen == false && detected == false) return 0;
    if(interfaceEnttec == DMX_USB_PRO)
    {
        c = (char)PKT_SOM;
        ser.write((char *)&c, 1);
        //ser.SendByte(PKT_SOM);  /* start of message */
        //fprintf(stderr, "0x%02x ", c);

        c = pkt_type;
        ser.write((char *)&c, 1);
        //ser.SendByte(pkt_type); /* the packet label, type of packet */
        //fprintf(stderr, "0x%02x ", c);

        ser.write((char *)&len, sizeof(len)); /* 16-bit length of packet
       LSB-MSB */
        //ser.SendBuffer((byte *)&len, sizeof(len)); /* 16-bit length of packet
       LSB-MSB */

        /*for(i=0;i<sizeof(len);i++)
            fprintf(stderr, "0x%02x ", *(((char *)&len)+i));*/

        if (len > 0)
        {
            ser.write((char *)data, len); /* packet data */
            //ser.SendBuffer((byte *)data, len); /* packet data */
            /*for(i=0;i<len;i++)
            {
                    fprintf(stderr, "0x%02x ", *(data+i));
            }*/
        }
        c = PKT_EOM;
        ser.write((char *)&c, 1);/* end of message */
        //ser.SendByte(PKT_EOM); /* end of message */
        //fprintf(stderr, "0x%02x\n", (unsigned char)c);
    }
    else /* OPEN_DMX_USB */
    {
        // 88us break
        //ser.SetSerialBreak(1);
        this->sleep(88);
        //ser.SetSerialBreak(0);
        this->sleep(8);
        if (len > 0)
        {
            ser.write((char *)data, sizeof(len));/* packet data */
            //ser.SendBuffer((byte *)data, len); /* packet data */
        }
    }
    return 1;
}
\n", nomInterfaces[interfaceEnttec]);
    else    fprintf(stderr, " Interface %s dct       nomInterfaces[interfaceEnttec]);
    #endif
    if(comopen == false && detected == false) return 0;
    if(interfaceEnttec == DMX_USB_PRO)
    {
        c = (char)PKT_SOM;
        ser.write((char *)&c, 1);
```

```
        //ser.SendByte(PKT_SOM);  /* start of message */
        //fprintf(stderr, "0x%02x ", c);

        c = pkt_type;
        ser.write((char *)&c, 1);
        //ser.SendByte(pkt_type); /* the packet label, type of packet */
        //fprintf(stderr, "0x%02x ", c);

        ser.write((char *)&len, sizeof(len)); /* 16-bit length of packet
       LSB-MSB */
        //ser.SendBuffer((byte *)&len, sizeof(len)); /* 16-bit length of packet
       LSB-MSB */

        /*for(i=0;i<sizeof(len);i++)
            fprintf(stderr, "0x%02x ", *(((char *)&len)+i));*/

        if (len > 0)
        {
            ser.write((char *)data, len); /* packet data */
            //ser.SendBuffer((byte *)data, len); /* packet data */
            /*for(i=0;i<len;i++)
            {
                    fprintf(stderr, "0x%02x ", *(data+i));
            }*/
        }
        c = PKT_EOM;
        ser.write((char *)&c, 1);/* end of message */
        //ser.SendByte(PKT_EOM); /* end of message */
        //fprintf(stderr, "0x%02x\n", (unsigned char)c);
    }
    else /* OPEN_DMX_USB */
    {
        // 88us break
        //ser.SetSerialBreak(1);
        this->sleep(88);
        //ser.SetSerialBreak(0);
        this->sleep(8);
        if (len > 0)
        {
            ser.write((char *)data, sizeof(len));/* packet data */
            //ser.SendBuffer((byte *)data, len); /* packet data */
        }
    }
    return 1;
}
\n", nomInterfaces[interfaceEnttec]);
    #endif

    if(comopen == false && detected == false) return 0;
    if(interfaceEnttec == DMX_USB_PRO)
    {
        c = (char)PKT_SOM;
        ser.write((char *)&c, 1);
        //ser.SendByte(PKT_SOM);  /* start of message */
        //fprintf(stderr, "0x%02x ", c);

        c = pkt_type;
        ser.write((char *)&c, 1);
        //ser.SendByte(pkt_type); /* the packet label, type of packet */
        //fprintf(stderr, "0x%02x ", c);

        ser.write((char *)&len, sizeof(len)); /* 16-bit length of packet
       LSB-MSB */
        //ser.SendBuffer((byte *)&len, sizeof(len)); /* 16-bit length of packet
       LSB-MSB */

        /*for(i=0;i<sizeof(len);i++)

            fprintf(stderr, "0x%02x ", *(((char *)&len)+i));*/

        if (len > 0)
        {
            ser.write((char *)data, len); /* packet data */
            //ser.SendBuffer((byte *)data, len); /* packet data */
```

```
            /*for(i=0;i<len;i++)

            {

                    fprintf(stderr, "0x%02x ", *(data+i));

            }*/
        }
        c = PKT_EOM;
        ser.write((char *)&c, 1);/* end of message */
        //ser.SendByte(PKT_EOM); /* end of message */
        //fprintf(stderr, "0x%02x\n", (unsigned char)c);
    }
    else /* OPEN_DMX_USB */
    {
        // 88us break
        //ser.SetSerialBreak(1);
        this->sleep(88);
        //ser.SetSerialBreak(0);
        this->sleep(8);
        if (len > 0)
        {
            ser.write((char *)data, sizeof(len));/* packet data */
            //ser.SendBuffer((byte *)data, len); /* packet data */
        }
    }
    return 1;
}
```

8.8.3.23 bool EnttecDMXUSB : :SetCanalDMX (int *canal*, byte *valeur*)

Références dmxout, et NB_CANAUX_MAX.

Référencé par IHM : :envoyerTrameDMX(), et IHM : :executerScene().

```
{
    if(canal > 0 && canal <= NB_CANAUX_MAX) /* 1 12 */
    {
        dmxout[canal] = valeur;
        #ifdef DEBUG_DMX_USB

        //fprintf(stderr, " EnttecDMXUSB::SetCanalDMX() canal %d -> 0x%02X\n",
       canal, valeur);
        #endif

    }
    else    return false;
    return true;
}
```

8.8.3.24 bool EnttecDMXUSB : :SetNbCanauxDMX (int *start* = 1, int *length* = NB_CANAUX_MAX)

Références dmxout_length, et NB_CANAUX_MAX.

```
{
    if((start > 0 && start <= NB_CANAUX_MAX && length > 0) && ((start+length)
      <= NB_CANAUX_MAX)) /* 1 12 */
    {
        dmxout_length = start+length;
        #ifdef DEBUG_DMX_USB

        fprintf(stderr, " EnttecDMXUSB::SetNbCanauxDMX() de %d d = %d\n",
      start, length, dmxout_length);
        #endif
```

```
    }
    else    return false;
    return true;
}
```

8.8.3.25 void EnttecDMXUSB : :sleep (int *usec*) [private]

Référencé par GetConfiguration(), openPort(), et sendPacket().

```
{
#ifdef HAVE_NANOSLEEP

        struct timespec req;

        req.tv_sec = usec / 1000000;
        usec -= req.tv_sec * 1000000;
        req.tv_nsec = usec * 1000;

        nanosleep(&req, NULL);
#else

    usleep(usec);
#endif

}
```

8.8.3.26 void EnttecDMXUSB : :widgetRecieveAllMode ( ) [private]

Références device_mode, PKT_DMXIN, PKT_DMXIN_MODE, et sendPacket().

```
{
   byte mode;
   mode = 0;
   sendPacket(PKT_DMXIN_MODE, (char *)&mode, 1);
   device_mode = PKT_DMXIN;
}
```

8.8.3.27 void EnttecDMXUSB : :widgetRecieveOnChangeMode ( ) [private]

Références device_mode, PKT_DMXIN, PKT_DMXIN_MODE, et sendPacket().

```
{
   byte mode;
   mode = 1;
   sendPacket(PKT_DMXIN_MODE, (char *)&mode, 1);
   device_mode = PKT_DMXIN;
}
```

8.8.3.28 void EnttecDMXUSB : :widgetRequestConfig ( ) [private]

Références PKT_GETCFG, et sendPacket().

Référencé par GetConfiguration(), et openPort().

```
{
  sendPacket(PKT_GETCFG, (char *)NULL, 0);
}
```

8.8.3.29 void EnttecDMXUSB : :widgetRequestSerial ( ) [private]

Références PKT_GETSERIAL, et sendPacket().

Référencé par GetConfiguration(), et openPort().

```
{
  sendPacket(PKT_GETSERIAL, (char *)NULL, 0);
}
```

8.8.4 Documentation des données membres

8.8.4.1 byte EnttecDMXUSB : :BreakTime [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), et recieve().

8.8.4.2 char EnttecDMXUSB : :buffer[4096] [private]

Référencé par EnttecDMXUSB(), processUpdatePacket(), et recieve().

8.8.4.3 bool EnttecDMXUSB : :comopen [private]

Référencé par closePort(), EnttecDMXUSB(), openPort(), recieve(), et sendPacket().

8.8.4.4 string EnttecDMXUSB : :comport [private]

Référencé par EnttecDMXUSB(), openPort(), et sendPacket().

8.8.4.5 bool EnttecDMXUSB : :config [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), openPort(), et recieve().

8.8.4.6 bool EnttecDMXUSB : :detected [private]

Référencé par closePort(), EnttecDMXUSB(), openPort(), recieve(), et sendPacket().

8.8.4.7 byte EnttecDMXUSB : :device_mode [private]

Référencé par openPort(), SendDatasDMX(), SendDMX(), widgetRecieveAllMode(), et widgetRecieveOnChangeMode().

8.8.4.8 bool EnttecDMXUSB : :dmx_available [private]

Référencé par EnttecDMXUSB(), et recieve().

8.8.4.9 TDmxArray EnttecDMXUSB : :dmxin [private]

Référencé par EnttecDMXUSB(), processUpdatePacket(), et recieve().

8.8.4.10 bool EnttecDMXUSB : :dmxin_filter [private]

Référencé par EnttecDMXUSB(), openPort(), et recieve().

8.8.4.11 int EnttecDMXUSB : :dmxin_length [private]

Référencé par EnttecDMXUSB(), openPort(), et recieve().

8.8.4.12 byte EnttecDMXUSB : :dmxin_mode [private]

8.8.4.13 int EnttecDMXUSB : :dmxin_quality [private]

Référencé par EnttecDMXUSB(), et recieve().

8.8.4.14 TDmxArray EnttecDMXUSB : :dmxout [private]

Référencé par EnttecDMXUSB(), ResetCanauxDMX(), SendDatasDMX(), SendDMX(), et SetCanalDMX().

8.8.4.15 int EnttecDMXUSB : :dmxout_length [private]

Référencé par EnttecDMXUSB(), openPort(), SendDatasDMX(), SendDMX(), et SetNbCanauxDMX().

8.8.4.16 byte EnttecDMXUSB : :FirmwareH [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), et recieve().

8.8.4.17 byte EnttecDMXUSB : :FirmwareL [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), et recieve().

8.8.4.18 byte EnttecDMXUSB : :FrameRate [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), et recieve().

8.8.4.19 EnttecInterfaces EnttecDMXUSB : :interfaceEnttec [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), openPort(), et sendPacket().

8.8.4.20 byte EnttecDMXUSB : :MABTime [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), et recieve().

8.8.4.21 QextSerialPort EnttecDMXUSB : :ser [private]

Référencé par closePort(), openPort(), recieve(), et sendPacket().

8.8.4.22 string EnttecDMXUSB : :SerialNumber [private]

Référencé par DisplayConfig(), EnttecDMXUSB(), GetConfiguration(), et recieve().

La documentation de cette classe a été générée à partir des fichiers suivants :
- enttecdmxusb.h
- enttecdmxusb.cpp

8.9 Référence de la structure EtatFaders

Structure permettant de stocker les valeurs des faders.

`#include <PlaybackWing.h>`

Attributs publics

- int fader0
- int fader1
- int fader2
- int fader3
- int fader4
- int fader5
- int fader6
- int fader7
- int fader8
- int fader9

8.9.1 Description détaillée

Auteur

Demont Thomas

8.9.2 Documentation des données membres

8.9.2.1 int EtatFaders : :fader0

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.2 int EtatFaders : :fader1

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.3 int EtatFaders : :fader2

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.4 int EtatFaders : :fader3

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.5 int EtatFaders : :fader4

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.6 int EtatFaders : :fader5

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.7 int EtatFaders : :fader6

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.8 int EtatFaders : :fader7

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.9 int EtatFaders : :fader8

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

8.9.2.10 int EtatFaders : :fader9

Référencé par IHM : :afficherEtatFaders(), et PlaybackWing : :ExtraireFaders().

La documentation de cette structure a été générée à partir du fichier suivant :
– PlaybackWing.h

8.10 Référence de la structure EtatTouchesControle

Structure permettant de stocker les valeurs des touches de contrôle (Page Up, Page Back, Back, Go)

`#include <PlaybackWing.h>`

Attributs publics

– int touchePageUp
– int touchePageDown
– int toucheBack
– int toucheGo

8.10.1 Description détaillée

Auteur

Demont Thomas

8.10.2 Documentation des données membres

8.10.2.1 int EtatTouchesControle : :toucheBack

Référencé par IHM : :afficherEtatTouchesControle(), et PlaybackWing : :Decoder-TouchesControle().

8.10.2.2 int EtatTouchesControle : :toucheGo

Référencé par IHM : :afficherEtatTouchesControle(), et PlaybackWing : :Decoder-TouchesControle().

8.10.2.3 int EtatTouchesControle : :touchePageDown

Référencé par IHM : :afficherEtatTouchesControle(), et PlaybackWing : :DecoderTouchesControle().

8.10.2.4 int EtatTouchesControle : :touchePageUp

Référencé par IHM : :afficherEtatTouchesControle(), et PlaybackWing : :DecoderTouchesControle().

La documentation de cette structure a été générée à partir du fichier suivant :
– PlaybackWing.h

8.11 Référence de la classe IDProjecteurSauvegarde

`#include <idProjecteurSauvegarde.h>`

Signaux

– void modification (QString nomAppareilModifie, QString uuidAppareilModifie)
– void suppression (QString uuidAppareilModifie)
– void informations (QString nomAppareilModifie, QString uuidAppareilModifie)

Fonctions membres publiques

– IDProjecteurSauvegarde (QWidget ∗parent=0)
– void setLabel (QString nom)
Mutateur du contenu texte du Label nomProjecteur.
– void setUuid (QString uuid)
Mutateur de l'attribut uuid de la classe IDProjecteurSauvegarde.
– QString getLabel ()
Accesseur du contenu texte du Label nomProjecteur.
– QString getUuid ()
Accesseur de l'attribut uuid de la classe IDProjecteurSauvegarde.

Connecteurs privés

– void modifier ()
Emet le signal modification.
– void supprimer ()
Emet le signal suppression.
– void informer ()
Emet le signal informations.

Attributs privés

– QLabel ∗ nomProjecteur
– QPushButton ∗ boutonModifier
– QPushButton ∗ boutonSupprimer
– QPushButton ∗ boutonInformations
– QString uuid

8.11.1 Documentation des constructeurs et destructeur

8.11.1.1 IDProjecteurSauvegarde : :IDProjecteurSauvegarde (QWidget * *parent* = 0) [explicit]

Références boutonInformations, boutonModifier, boutonSupprimer, informer(), modifier(), nomProjecteur, et supprimer().

```
                                                                    :
    QWidget(parent)
{
    QHBoxLayout *hLayout = new QHBoxLayout;
    nomProjecteur = new QLabel();
    nomProjecteur->setText("");
    boutonModifier = new QPushButton(QString::fromUtf8("Modifier"), this);
    boutonSupprimer = new QPushButton(QString::fromUtf8("Supprimer"), this);
    boutonInformations = new QPushButton(QString::fromUtf8("Informations"),
      this);



    hLayout->addWidget(nomProjecteur);
    hLayout->addWidget(boutonModifier);
    hLayout->addWidget(boutonSupprimer);
    hLayout->addWidget(boutonInformations);

    setLayout(hLayout);
    connect(boutonModifier, SIGNAL(clicked()), this, SLOT(modifier()));
    connect(boutonSupprimer, SIGNAL(clicked()), this, SLOT(supprimer()));
    connect(boutonInformations, SIGNAL(clicked()), this, SLOT(informer()));
}
```

8.11.2 Documentation des fonctions membres

8.11.2.1 QString IDProjecteurSauvegarde : :getLabel ( )

Renvoie

QString

Références nomProjecteur.

Référencé par informer(), et modifier().

```
{
    return nomProjecteur->text();
}
```

8.11.2.2 QString IDProjecteurSauvegarde : :getUuid ( )

Renvoie

QString

Références uuid.

Référencé par informer(), modifier(), et supprimer().

```
{
    return uuid;
}
```

8.11.2.3 void IDProjecteurSauvegarde : :informations (QString *nomAppareilModifie*, QString *uuidAppareilModifie*) [signal]

Référencé par informer().

8.11.2.4 void IDProjecteurSauvegarde : :informer ( ) [private, slot]

Renvoie

void

Références getLabel(), getUuid(), et informations().

Référencé par IDProjecteurSauvegarde().

```
{
    emit informations(this->getLabel(), this->getUuid());
}
```

8.11.2.5 void IDProjecteurSauvegarde : :modification (QString *nomAppareilModifie*, QString *uuidAppareilModifie*) [signal]

Référencé par modifier().

8.11.2.6 void IDProjecteurSauvegarde : :modifier ( ) [private, slot]

Renvoie

void

Références getLabel(), getUuid(), et modification().

Référencé par IDProjecteurSauvegarde().

```
{
    emit modification(this->getLabel(), this->getUuid());
}
```

8.11.2.7 void IDProjecteurSauvegarde : :setLabel (QString *nom*)

Paramètres

nom	

Renvoie

void

Références nomProjecteur.

Référencé par XMLUtilitaire : :afficherProjecteursEnregistres().

```
{
    nomProjecteur->setText(nom);
    //qDebug() << Q_FUNC_INFO << "nom donne:" << nom;
}
```

8.11.2.8 void IDProjecteurSauvegarde : :setUuid (QString *uuid*)

Paramètres

uuid	

Renvoie

void

Références uuid.

Référencé par XMLUtilitaire : :afficherProjecteursEnregistres().

```
{
    this->uuid = uuid;
}
```

8.11.2.9 void IDProjecteurSauvegarde : :suppression (QString *uuidAppareilModifie*) [signal]

Référencé par supprimer().

8.11.2.10 void IDProjecteurSauvegarde : :supprimer ( ) [private, slot]

Renvoie

void

Références getUuid(), et suppression().

Référencé par IDProjecteurSauvegarde().

```
{
    emit suppression(/*this->getLabel(),*/ this->getUuid());
}
```

8.11.3 Documentation des données membres

8.11.3.1 QPushButton* IDProjecteurSauvegarde : :boutonInformations [private]

Référencé par IDProjecteurSauvegarde().

8.11.3.2 QPushButton* IDProjecteurSauvegarde : :boutonModifier [private]

Référencé par IDProjecteurSauvegarde().

8.11.3.3 QPushButton* IDProjecteurSauvegarde : :boutonSupprimer [private]

Référencé par IDProjecteurSauvegarde().

8.11.3.4 QLabel* IDProjecteurSauvegarde : :nomProjecteur [private]

Référencé par getLabel(), IDProjecteurSauvegarde(), et setLabel().

8.11.3.5 QString IDProjecteurSauvegarde : :uuid [private]

Référencé par getUuid(), et setUuid().

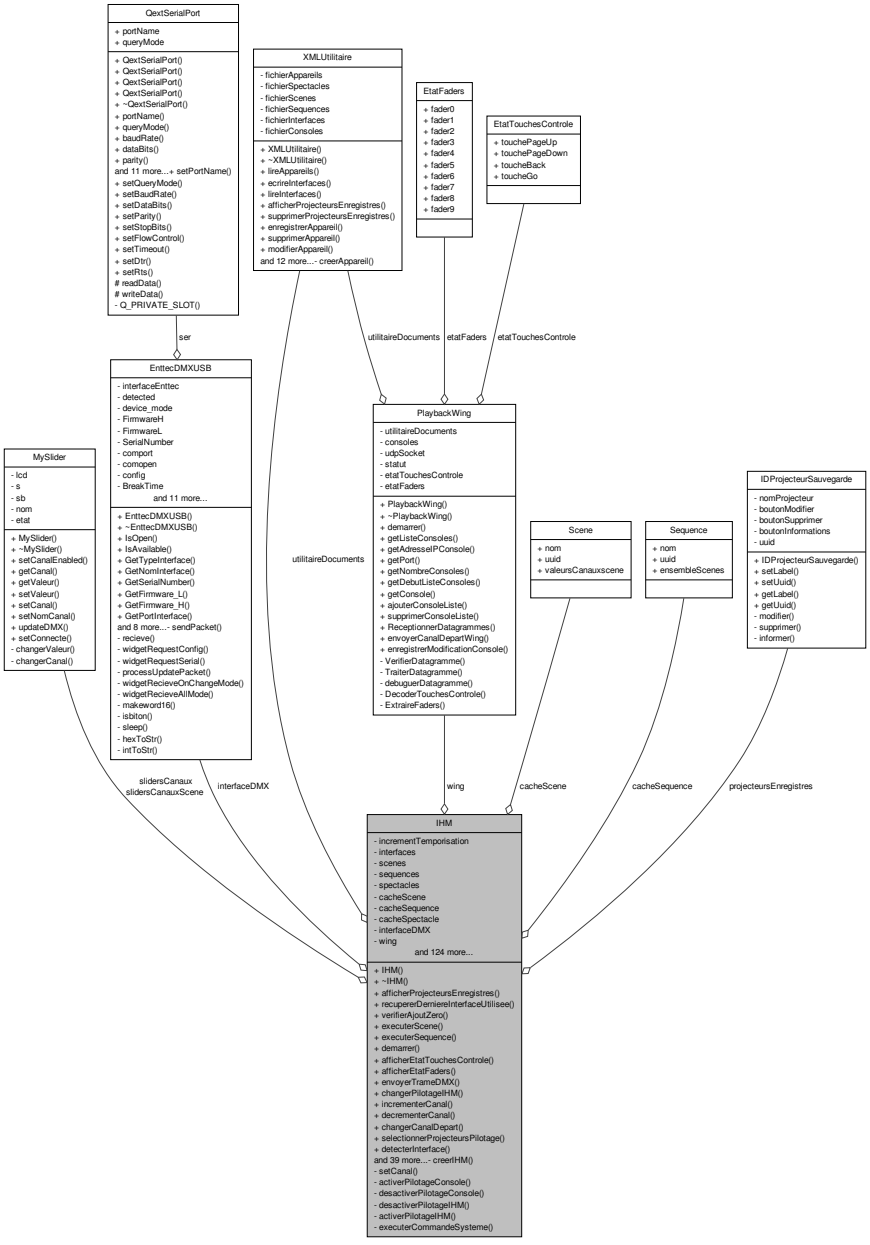
La documentation de cette classe a été générée à partir des fichiers suivants :
– idProjecteurSauvegarde.h
– idProjecteurSauvegarde.cpp

8.12 Référence de la classe IHM

Définition de la classe IHM.

```
#include <IHM.h>
```

Graphe de collaboration de IHM :



Connecteurs publics

– void demarrer ()

Démarre la réception des données en provenance de la console WING.

– void afficherEtatTouchesControle (EtatTouchesControle etatTouchesControle)

Méthode régissant l'appui des touches de contrôles (Page Up, PAge Down, Back, Go) de la console wing.

- void afficherEtatFaders (EtatFaders etatFaders)
  Affiche les valeurs des faders de la console WING sur les sliders de l'IHM.
- void envoyerTrameDMX (int valeurFader, int valeurCanal)
  Envoie une valeur sur un canal DMX.
- void changerPilotageIHM (QString choix)
  Active ou désactive le pilotage des projecteurs depuis l'IHM par rapport à l'état du QComboBox listeChoixPilotage.
- void incrementerCanal ()
  Méthode permettant d'incrémenter la valeur de départ du canal.
- void decrementerCanal ()
  Méthode permettant de décrémenter la valeur de départ du canal.
- void changerCanalDepart (int canal)
- void selectionnerProjecteursPilotage (int index)
  Méthode gérant le type de projecteurs séléctionné (sur la liste proposée), l'affichage ou non des sliders selon le nombre de canaux du projecteur séléctionné et l'incrémentation du canal de départ par rapport au numéro indiqué dans le type de projecteur.
- void detecterInterface ()
  Detecte les interfaces séries connectées au PC, les affiches sur l'IHM, et si l'interface sélectionnée (dans la liste du choix de l'interface) est une interface Enttec Pro, affiche ses caractéristiques.
- void selectionnerInterface (int index)
- void enregistrerProjecteur ()
  Méthode permettant l'enregistrement de projecteurs à partir de l'utilitaire XML.
- void genererFenetreNouveauProjecteur ()
  Création de la fenetre d'interface pour créer un nouveau Projecteur.
- void bloquerChoixNomCanaux (int valeur)
  Bloquage des LineEdit de noms de canaux en trop.
- void genererFenetreNouvelleScene ()
  Création de la fenetre d'interface pour créer une nouvelle Scène.
- void genererFenetreNouvelleSequence ()
  Création de la fenetre d'interface pour créer une nouvelle Séquence.
- void genererFenetreInformationsAppareil ()
  Création de la fenetre contenant les information d'un projecteur au choix.
- void genererFenetreCanauxScene ()
  Création de la fenetre d'options des canaux d'un apparail à ajouter pour une scène.
- void ouvrirFenetreNouveauProjecteur ()
  Ouverture de la fenetre permettant l'ajout d'un projecteur dans le fichier XML.
- void ouvrirFenetreModifierProjecteur (QString projecteur, QString uuid)
  Ouverture de la fenetre permettant la modification d'un projecteur dans le fichier XML.
- void ouvrirFenetreInformationsProjecteur (QString projecteur, QString uuid)
  Affichage des informations d'un projecteur depuis l'IHM.
- void ouvrirFenetreNouvelleScene ()
  Affichage de la fenêtre d'options de creation d'une scène.
- void ouvrirFenetreCanauxScene ()
  Affichage de la fenêtre d'options d'ajustements des canaux de l'appareil à implémenter.
- void ouvrirFenetreNouvelleSequence ()
  Ouverture de la fenetre permettant l'ajout d'une sequence dans le fichier XML.
- void enregistrerScene ()
  Enregistrement d'une scène depuis l'IHM.
- void ajouterAppareilScene ()
  Ajout d'un appareil à la scène en cours de création.
- void enregistrerSequence ()
- void ajouterSceneSequence ()
  ajoute une scène dans le cache de la séquence à enregistrer
- void executerSceneDefault ()
- void supprimerScene ()
  Suppression d'une scène depuis l'IHM.

- void executerSequenceDefault ()
- void supprimerSequence ()
- void supprimerProjecteur (QString uuid)
  Supression d'un projecteur du fichier XML depuis l'IHM.
- void supprimerTousProjecteurs ()
  Supression simultanée de tous les projecteurs depuis l'IHM.
- void recupererDonneesNouvelleInterface ()
  Récupere les données entrées dans les LineEdit de l'ajout d'interface et les place dans la liste interfaces.
- void quitter ()
  Ferme l'application.
- void supprimerInterface ()
  Supprime l'interface DMX du fichier adaptateurs.xml et de la QListe interfaces.
- void ajouterConsole ()
  Ajoute une console à la QList consoles dont les paramètres seront définis par les QLineEdit champAdresseIPConsole, champPortConsole.
- void genererFenetreModifierConsole ()
  Genere une nouvelle fenetre de dialogue qui contient les informations de la console à modifier.
- void envoyerModificationConsole ()
- void supprimerConsole ()
  Supprime la console activement séléctionnée dans le QComboBox listeChoixConsole de la QList consoles.
- void ajouterSequenceSpectacle ()
  ajoute une séquence dans le cache du spectacle à enregistrer
- void enregistrerSpectacle ()
- void executerSpectacle ()
- void supprimerSpectacle ()
- void effacerAffichageProjecteursEnregistres ()
  Efface l'affichage des projecteurs enregistrés dans le menu d'ajout de projecteurs.
- void mettreAJourProjecteursEnregistres ()
  Réaffiche la liste mise à jour des projecteurs enregistrés dans le menu d'ajout de projecteur.
- void effacerAffichageProjecteursPilotage ()
- void mettreAJourProjecteursPilotage ()
  Réaffiche les projecteurs en index mis à jour dans listeProjecteursPilotage.
- void envoyerCanalDepartWing ()

Fonctions membres publiques

- IHM (QWidget *parent=0)
  Constructeur de la fenêtre principale.
- ~IHM ()
  Destructeur.
- bool afficherProjecteursEnregistres ()
- void recupererDerniereInterfaceUtilisee ()
- void verifierAjoutZero (QString &nombre)
  ajoute un 0 à coté d'un nombre si ce dernier est inferieur à 10
- void executerScene (Scene scene)
  Exécution d'une scène sur la chaîne DMX.
- void executerSequence (Sequence sequence)

Fonctions membres privées

- void creerIHM ()
  Crée la fenêtre principale.
- void setCanal ()

Méthode permettant de fixer la valeur des canaux.

- void activerPilotageConsole ()
  Connecte l'envoi de l'etat des faders depuis la console avec leur affichage dans l'éventualité où la console est désactivée.
- void desactiverPilotageConsole ()
  déconnecte l'envoi de l'etat des faders depuis la console avec leur affichage
- void desactiverPilotageIHM ()
  Désactive les sliders de l'IHM.
- void activerPilotageIHM ()
  Active les sliders de l'IHM.
- QStringList executerCommandeSysteme (QString commande)
  Execute la commande Syteme passée en paramètre.

Attributs privés

- int incrementTemporisation
- QList< Interface > interfaces
  Liste d'interfaces Enttex USB.
- QVector< Scene > scenes
  Liste des scenes enregistrées depuis le fichier scenes.xml.
- QVector< Sequence > sequences
  Liste des sequences enregistrées depuis le fichier sequences.xml.
- QVector< Spectacle > spectacles
  Liste des spectacles enregistrés depuis le fichier spectacles.xml.
- Scene cacheScene
  cache d'un objet Scene retenu dans cet espace mémoire avant d'etre implémenté dans un conteneur
- Sequence cacheSequence
  cache d'un objet Sequence retenu dans cet espace mémoire avant d'etre implémenté dans un conteneur
- Spectacle cacheSpectacle
  cache d'un objet Spectacle retenu dans cet espace mémoire avant d'etre implémenté dans un conteneur
- EnttecDMXUSB * interfaceDMX
  Association vers la classe EnttecDMXUSB.
- PlaybackWing * wing
  Association vers la classe PlaybackWing.
- QVector< DMXProjecteur * > projecteursDMX
  rampe de projecteurs DMX
- XMLUtilitaire * utilitaireDocuments
  Utilitaire de gestion des fichiers xml.
- int nbAppareils
- int canalDepartFaders
- bool pilotageIHMActif
- bool pilotageDMXActif
- QTabWidget * ongletsPrincipaux
- QTabWidget * optionsCreations
- QTabWidget * optionsParametres
- QTabWidget * optionsSpectacles
- QWidget * pageAssembler
- QWidget * pageCreer
- QWidget * pageJouer
- QWidget * pagePiloter
- QWidget * parametresInterface
- QWidget * parametresProjecteurs
- QWidget * parametresConsoles
- QWidget * pagesSpectacles [2]
- QDialog * fenetreNouveauProjecteur
- QDialog * fenetreNouvelleScene
- QDialog * fenetreNouvelleSequence

- QDialog * fenetreInformationsAppareil
- QDialog * fenetreCanauxScene
- QDialog * fenetreModifierConsole
- MySlider * slidersCanaux [NB_SLIDERS]
- QScrollArea * listeProjecteursEnregistres
- QPushButton * boutonNouveauProjecteur
- QPushButton * boutonSupprimerTousProjecteurs
- QComboBox * listeChoixPilotage
- QComboBox * listeProjecteursPilotage
- QPushButton * boutonIncrementationCanal
- QPushButton * boutonDecrementationCanal
- QLabel * labelNom
- QLineEdit * choixNom
- QLabel * labelType
- QComboBox * choixTypes
- QLabel * labelNBCanaux
- QSpinBox * choixNBCanaux
- QLabel * labelCanalDepart
- QSpinBox * choixCanalDepart
- QLabel * labelNomCanaux
- QLineEdit * nomsCanaux [10]
- QPushButton * boutonEnregistrer
- QLabel * labelInformationsCanalDepart
- QLabel * labelInformationsCanaux [10]
- QScrollArea * listeInformationsCanaux
- QLabel * labelInterface
- QComboBox * listeChoixInterface
- QComboBox * listeTypeInterface
- QPushButton * boutonDetecterInterface
- QTextEdit * resultatDetectionInterface
- QLabel * labelScenesDisponibles
- QComboBox * choixScenesDisponibles
- QPushButton * boutonexecuterScene
- QPushButton * boutonModifierScene
- QPushButton * boutonSupprimerScene
- QPushButton * boutonNouvelleScene
- QPushButton * boutonSupprimerToutesScenes
- QLabel * labelNomScene
- QLineEdit * choixNomScene
- QLabel * labelAppareilScene
- QComboBox * choixAppareilScene
- QPushButton * boutonParametresCanauxScene
- QScrollArea * listeAppareilsScene
- QList< QLabel * > labelsAppareilsScene
- QPushButton * boutonSauvegarderScene
- MySlider * slidersCanauxScene [NB_SLIDERS]
- QPushButton * boutonSauvegarderCanauxScene
- QLineEdit * numeroIdAjouterInterface
- QLineEdit * portInterfaceAjouterInterface
- QComboBox * listeTypeInterfaceAjouterInterface
- QPushButton * boutonAjouterInterface
- QComboBox * listeInterfacesSupprimables
- QPushButton * boutonSupprimerInterface
- QComboBox * listeChoixConsole
- QLineEdit * champAdresseIPConsole
- QLineEdit * champPortConsole
- QPushButton * boutonAjouterConsole
- QPushButton * boutonSupprimerConsole
- QPushButton * boutonModifierConsole
- QLineEdit * champAdresseIPModifierConsole
- QLineEdit * champPortModifierConsole
- QPushButton * boutonEnregistrerModificationConsole
- QLabel * labelSequencesDisponibles
- QComboBox * choixSequencesDisponibles
- QPushButton * boutonExecuterSequence
- QPushButton * boutonNouvelleSequence

- QPushButton * boutonSupprimmerSequence
- QLabel * labelnomSequence
- QLineEdit * choixnomSequence
- QLabel * labelScenesSequence
- QComboBox * choixScenesSequence
- QLabel * labelTemporisation
- QSpinBox * choixHeures
- QLabel * labelHeures
- QSpinBox * choixMinutes
- QLabel * labelMinutes
- QSpinBox * choixSecondes
- QLabel * labelSecondes
- QPushButton * boutonAjouterScene
- QScrollArea * listeScenesSequenceChoisies
- QList< QLabel * > labelsScenesSequences
- QPushButton * boutonEnregistrerSequence
- QLabel * labelCreationSpectacle
- QLabel * labelnomSpectacle
- QLineEdit * choixNomSpectacle
- QLabel * labelSequencesSpectacle
- QComboBox * choixSequencesSpectacle
- QPushButton * boutonchoixSequenceSpectacle
- QScrollArea * listeSequencesSpectacle
- QList< QLabel * > labelsSequencesSpectacle
- QPushButton * boutonSauvegarderSpectacle
- QLabel * labelJeuSpectacle
- QLabel * labelSpectaclesDisponibles
- QComboBox * choixSpectaclesDisponibles
- QSpacerItem * espace
- QPushButton * boutonExecuterSpectacle
- QPushButton * boutonSupprimerSpectacle
- QAction * actionQuitter
- QVBoxLayout * layoutListeProjecteursParametres
- QVBoxLayout * layoutlisteAppareilsScene
- QVBoxLayout * layoutListeScenesSequences
- QVBoxLayout * layoutListeSequenceSpectacle
- IDProjecteurSauvegarde * projecteursEnregistres [NB_PROJECTEURS]
- QString cacheUuid

8.12.1 Description détaillée

La fenêtre principale de l'application.

Auteur

Demont Thomas, Reynier Tony

Version

1.0

Auteur

Demont Thomas, Reynier Tony

Version

1.1

8.12.2 Documentation des constructeurs et destructeur

8.12.2.1 IHM : :IHM (QWidget * *parent* = 0) [explicit]

Paramètres

parent	QObject Adresse de l'objet Qt parent ici 0 car c'est la fenêtre principale

Références actionQuitter, afficherEtatFaders(), afficherEtatTouchesControle(), ajouterConsole(), ajouterSequenceSpectacle(), boutonAjouterConsole, boutonAjouterInterface, boutonchoixSequenceSpectacle, boutonDecrementationCanal, boutonDetecterInterface, boutonEnregistrer, boutonexecuterScene, boutonExecuterSequence, boutonExecuterSpectacle, boutonIncrementationCanal, boutonModifierConsole, boutonNouveauProjecteur, boutonNouvelleScene, boutonNouvelleSequence, boutonParametresCanauxScene, boutonSauvegarderSpectacle, boutonSupprimerConsole, boutonSupprimerInterface, boutonSupprimerScene, boutonSupprimerSpectacle, boutonSupprimerTousProjecteurs, boutonSupprimmerSequence, canalDepartFaders, changerCanalDepart(), changerPilotageIHM(), creerIHM(), decrementerCanal(), PlaybackWing : :demarrer(), detecterInterface(), enregistrerProjecteur(), enregistrerSpectacle(), envoyerTrameDMX(), executerSceneDefault(), executerSequenceDefault(), executerSpectacle(), genererFenetreInformationsAppareil(), genererFenetreModifierConsole(), genererFenetreNouveauProjecteur(), genererFenetreNouvelleScene(), genererFenetreNouvelleSequence(), incrementerCanal(), interfaceDMX, interfaces, XMLUtilitaire : :lireAppareils(), XMLUtilitaire : :lireInterfaces(), listeChoixInterface, listeChoixPilotage, listeProjecteursPilotage, NB_SLIDERS, ouvrirFenetreCanauxScene(), ouvrirFenetreNouveauProjecteur(), ouvrirFenetreNouvelleScene(), ouvrirFenetreNouvelleSequence(), pilotageDMXActif, pilotageIHMActif, projecteursDMX, quitter(), recupererDerniereInterfaceUtilisee(), recupererDonneesNouvelleInterface(), selectionnerInterface(), selectionnerProjecteursPilotage(), slidersCanaux, supprimerConsole(), supprimerInterface(), supprimerScene(), supprimerSequence(), supprimerSpectacle(), supprimerTousProjecteurs(), utilitaireDocuments, et wing.

```
                        : QWidget(parent)
{
    //qDebug() << Q_FUNC_INFO;

    interfaceDMX = NULL;
    wing = new PlaybackWing(this);
    utilitaireDocuments = new XMLUtilitaire(this);
    utilitaireDocuments->lireAppareils(projecteursDMX);
    for(int i = 0; i < projecteursDMX.count(); i++)
    {
        projecteursDMX.at(i)->setEnttecDMXUSB(interfaceDMX);
    }

    utilitaireDocuments->lireInterfaces(interfaces);
    canalDepartFaders = 1;
    pilotageIHMActif = true;
    pilotageDMXActif = false;

    //Mise en page
    creerIHM();
    genererFenetreNouvelleScene();
    genererFenetreNouvelleSequence();
    genererFenetreNouveauProjecteur();
    genererFenetreInformationsAppareil();
    // Sélectionne l'interface par défaut
    recupererDerniereInterfaceUtilisee();
```

```
connect(wing, SIGNAL(envoyerEtatTouchesControle(EtatTouchesControle)), this
  , SLOT(afficherEtatTouchesControle(EtatTouchesControle)));
connect(wing, SIGNAL(envoyerEtatFaders(EtatFaders)), this, SLOT(
  afficherEtatFaders(EtatFaders)));
for (int i = 0; i < NB_SLIDERS ; ++i)
{
    if(slidersCanaux[i] != NULL)
    {
        if(i == 0)
        {
            connect(slidersCanaux[0], SIGNAL(canalChange(int)), this, SLOT(
  changerCanalDepart(int)));
            connect(slidersCanaux[i], SIGNAL(sliderChange(int,int)), this,
  SLOT(envoyerTrameDMX(int,int)));
        }
        else
        {
            connect(slidersCanaux[i], SIGNAL(sliderChange(int,int)), this,
  SLOT(envoyerTrameDMX(int,int)));
        }
    }
}
connect(listeChoixPilotage, SIGNAL(activated(QString)), this, SLOT(
  changerPilotageIHM(QString)));
connect(listeProjecteursPilotage, SIGNAL(currentIndexChanged(int)), this,
  SLOT(selectionnerProjecteursPilotage(int)));
connect(boutonIncrementationCanal, SIGNAL(clicked()), this, SLOT(
  incrementerCanal()));
connect(boutonDecrementationCanal, SIGNAL(clicked()), this, SLOT(
  decrementerCanal()));
connect(boutonDetecterInterface, SIGNAL(clicked()), this, SLOT(
  detecterInterface()));
connect(listeChoixInterface, SIGNAL(currentIndexChanged(int)), this, SLOT(
  selectionnerInterface(int)));
connect(boutonNouveauProjecteur, SIGNAL(clicked()), this, SLOT(
  ouvrirFenetreNouveauProjecteur()));
connect(boutonEnregistrer, SIGNAL(clicked()), this, SLOT(
  enregistrerProjecteur()));
connect(boutonSupprimerTousProjecteurs, SIGNAL(clicked()), this, SLOT(
  supprimerTousProjecteurs()));
connect(actionQuitter, SIGNAL(triggered()), this, SLOT(quitter()));
connect(boutonAjouterInterface, SIGNAL(clicked()), this, SLOT(
  recupererDonneesNouvelleInterface()));

connect(boutonSupprimerInterface, SIGNAL(clicked()), this, SLOT(
  supprimerInterface()));

connect(boutonNouvelleScene, SIGNAL(clicked()), this, SLOT(
  ouvrirFenetreNouvelleScene()));
connect(boutonParametresCanauxScene, SIGNAL(clicked()), this, SLOT(
  ouvrirFenetreCanauxScene()));

connect(boutonexecuterScene, SIGNAL(clicked()),this,SLOT(
  executerSceneDefault()));
connect(boutonSupprimerScene, SIGNAL(clicked()),this,SLOT(supprimerScene())
  );

connect(boutonNouvelleSequence, SIGNAL(clicked()),this,SLOT(
  ouvrirFenetreNouvelleSequence()));
connect(boutonSupprimmerSequence, SIGNAL(clicked()),this,SLOT(
  supprimerSequence()));
connect(boutonExecuterSequence, SIGNAL(clicked()),this,SLOT(
  executerSequenceDefault()));

connect(boutonchoixSequenceSpectacle, SIGNAL(clicked()),this,SLOT(
  ajouterSequenceSpectacle()));
connect(boutonSauvegarderSpectacle, SIGNAL(clicked()),this,SLOT(
  enregistrerSpectacle()));
connect(boutonExecuterSpectacle, SIGNAL(clicked()),this,SLOT(
  executerSpectacle()));
connect(boutonSupprimerSpectacle, SIGNAL(clicked()),this,SLOT(
  supprimerSpectacle()));
```

```
    connect(boutonAjouterConsole, SIGNAL(clicked()), this, SLOT(ajouterConsole(
      )));
    connect(boutonSupprimerConsole, SIGNAL(clicked()), this, SLOT(
      supprimerConsole()));
    connect(boutonModifierConsole, SIGNAL(clicked()), this ,SLOT(
      genererFenetreModifierConsole()));
    wing->demarrer();

    pilotageDMXActif = true;
}
```

8.12.2.2 IHM : :∼IHM ( )

Références XMLUtilitaire : :ecrireInterfaces(), PlaybackWing : :envoyerCanalDepartWing(), interfaceDMX, interfaces, projecteursDMX, utilitaireDocuments, et wing.

```
{
    wing->envoyerCanalDepartWing(1);
    utilitaireDocuments->ecrireInterfaces(interfaces);
    for(int i = 0; i < projecteursDMX.count(); i++)
    {
        delete projecteursDMX.at(i);
    }
    delete interfaceDMX;
    //qDebug() << Q_FUNC_INFO;
}
```

8.12.3 Documentation des fonctions membres

8.12.3.1 void IHM : :activerPilotageConsole ( ) [private]

Renvoie

void

Références afficherEtatFaders(), et wing.

Référencé par desactiverPilotageIHM().

```
{
    connect(wing, SIGNAL(envoyerEtatFaders(EtatFaders)), this, SLOT(
      afficherEtatFaders(EtatFaders)));
}
```

8.12.3.2 void IHM : :activerPilotageIHM ( ) [private]

Renvoie

void

Références boutonDecrementationCanal, boutonIncrementationCanal, desactiverPilotageConsole(), listeProjecteursPilotage, NB_SLIDERS, pilotageDMXActif, pilotageIHMActif, et slidersCanaux.

Référencé par afficherEtatTouchesControle(), et changerPilotageIHM().

```
{
    pilotageIHMActif = true;
```

```
    for(int i = 0; i < NB_SLIDERS; i++)
    {
        slidersCanaux[i]->setEnabled(true);
    }
    boutonIncrementationCanal->setEnabled(true);
    boutonDecrementationCanal->setEnabled(true);
    listeProjecteursPilotage->setEnabled(true);

    desactiverPilotageConsole();

    pilotageDMXActif = true;
}
```

8.12.3.3 void IHM : :afficherEtatFaders (EtatFaders *etatFaders*) [slot]

Paramètres

etatFaders	EtatFaders structure regroupant l'ensemble des états des faders de la console WING

Renvoie

void

Références EtatFaders : :fader0, EtatFaders : :fader1, EtatFaders : :fader2, EtatFaders : :fader3, EtatFaders : :fader4, EtatFaders : :fader5, EtatFaders : :fader6, EtatFaders : :fader7, EtatFaders : :fader8, EtatFaders : :fader9, setCanal(), MySlider : :setValeur(), et slidersCanaux.

Référencé par activerPilotageConsole(), desactiverPilotageConsole(), et IHM().

```
{
        setCanal();

        slidersCanaux[0]->setValeur(etatFaders.fader0);
        slidersCanaux[1]->setValeur(etatFaders.fader1);
        slidersCanaux[2]->setValeur(etatFaders.fader2);
        slidersCanaux[3]->setValeur(etatFaders.fader3);
        slidersCanaux[4]->setValeur(etatFaders.fader4);
        slidersCanaux[5]->setValeur(etatFaders.fader5);
        slidersCanaux[6]->setValeur(etatFaders.fader6);
        slidersCanaux[7]->setValeur(etatFaders.fader7);
        slidersCanaux[8]->setValeur(etatFaders.fader8);
        slidersCanaux[9]->setValeur(etatFaders.fader9);
}
```

8.12.3.4 void IHM : :afficherEtatTouchesControle (EtatTouchesControle *etatTouchesControle*) [slot]

Paramètres

etatTouches-Controle	structure regroupant l'ensemble des états des touches de la console WING

Renvoie

void

Références activerPilotageIHM(), decrementerCanal(), desactiverPilotageIHM(), incrementerCanal(), listeChoixPilotage, listeProjecteursPilotage, pilotageIHMActif, setCanal(), EtatTouchesControle : :toucheBack, EtatTouchesControle : :toucheGo, EtatTouchesControle : :touchePageDown, et EtatTouchesControle : :touchePageUp.

Référencé par IHM().

```
{
    QString etats = QString::fromUtf8("Etats touches : ") + QString::number(
      etatTouchesControle.touchePageUp) + QString::number(etatTouchesControle.
      touchePageDown) + QString::number(etatTouchesControle.toucheBack) +
      QString::number(etatTouchesControle.toucheGo);

    if(!etatTouchesControle.toucheBack && !etatTouchesControle.touchePageUp)
    {
        int index = listeProjecteursPilotage->currentIndex();
        if(index != listeProjecteursPilotage->count() - 1)
        {
            listeProjecteursPilotage->setCurrentIndex(index + 1);
        }
        else qDebug() << "Dernier projecteur atteint";

    }
    if(!etatTouchesControle.toucheBack && !etatTouchesControle.touchePageDown)
    {
        int index = listeProjecteursPilotage->currentIndex();
        if(index != 0)
        {
            listeProjecteursPilotage->setCurrentIndex(index - 1);
        }
        else qDebug() << "Premier projecteur atteint";

    }

    //qDebug() << Q_FUNC_INFO << etats;

    if(!pilotageIHMActif)
    {
        if(etatTouchesControle.touchePageUp == 0)
        {
            if(etatTouchesControle.toucheBack)
            incrementerCanal();
            //qDebug() << Q_FUNC_INFO << "canal depart = " <<
       canalDepartFaders;
        }
        if(etatTouchesControle.touchePageDown == 0)
        {
            if(etatTouchesControle.toucheBack)
            decrementerCanal();
            //qDebug() << Q_FUNC_INFO << "canal depart = " <<
       canalDepartFaders;
        }
    }
    if(etatTouchesControle.toucheBack == 0)
    {
        desactiverPilotageIHM();
        listeChoixPilotage->setCurrentIndex(1);
    }
    if(etatTouchesControle.toucheGo == 0)
    {
        activerPilotageIHM();
        listeChoixPilotage->setCurrentIndex(0);
    }

    setCanal();
}
```

8.12.3.5 bool IHM : :afficherProjecteursEnregistres ( )

8.12.3.6 void IHM : :ajouterAppareilScene ( ) [slot]

Renvoie

void

Références cacheScene, choixAppareilScene, fenetreCanauxScene, MySlider : :getCanal(), MySlider : :getValeur(), labelsAppareilsScene, layoutlisteAppareilsScene, NB_SLIDERS, slidersCanauxScene, et Scene : :valeursCanauxscene.

Référencé par genererFenetreCanauxScene().

```
{
    QLabel* appareil = new QLabel(choixAppareilScene->currentText(), this);
    labelsAppareilsScene.append(appareil);
    for(int i=0; i < NB_SLIDERS; i++)
    {
        cacheScene.valeursCanauxscene[slidersCanauxScene[i]->getCanal()] =
      slidersCanauxScene[i]->getValeur();
    }

    layoutlisteAppareilsScene->addWidget(labelsAppareilsScene.last());

    fenetreCanauxScene->close();
}
```

8.12.3.7 void IHM : :ajouterConsole ( ) [slot]

Renvoie

void

Références PlaybackWing : :ajouterConsoleListe(), champAdresseIPConsole, champPortConsole, PlaybackWing : :getConsole(), PlaybackWing : :getNombreConsoles(), listeChoixConsole, et wing.

Référencé par IHM().

```
{
    wing->ajouterConsoleListe(champAdresseIPConsole->text(), champPortConsole->
      text());

    listeChoixConsole->addItem(wing->getConsole(wing->getNombreConsoles()-1).
      adresseIP);
}
```

8.12.3.8 void IHM : :ajouterSceneSequence ( ) [slot]

Renvoie

void

Références cacheSequence, choixHeures, choixMinutes, choixScenesSequence, choixSecondes, Sequence : :ensembleScenes, labelsScenesSequences, layoutListeScenesSequences, SceneSequence : :scene, scenes, SceneSequence : :tempo, et verifierAjoutZero().

Référencé par genererFenetreNouvelleSequence().

```
{

    QString h = QString::number(choixHeures->value());
    QString m = QString::number(choixMinutes->value());
    QString s = QString::number(choixSecondes->value());

    verifierAjoutZero(h);
    verifierAjoutZero(m);
    verifierAjoutZero(s);


    QString texteSceneSequence = choixScenesSequence->currentText() + " " + h +
       ":" + m + ":" + s;
    QLabel* labelscene = new QLabel(texteSceneSequence, this);
    labelsScenesSequences.append(labelscene);

    layoutListeScenesSequences->addWidget(labelsScenesSequences.last());

    SceneSequence sceneSequence;
    sceneSequence.tempo = choixHeures->value() * 3600 + choixMinutes->value() *
       60 + choixSecondes->value();
    sceneSequence.scene = scenes[choixScenesSequence->currentIndex()];
    cacheSequence.ensembleScenes.append(sceneSequence);

}
```

8.12.3.9 void IHM : :ajouterSequenceSpectacle ( ) [slot]

Renvoie

void

Références cacheSpectacle, choixSequencesSpectacle, labelsSequencesSpectacle, layoutListeSequenceSpectacle, Sequence : :nom, sequences, et Sequence : :uuid.

Référencé par IHM().

```
{

    QLabel* labelsequence = new QLabel(choixSequencesSpectacle->currentText(),
      this);
    labelsSequencesSpectacle.append(labelsequence);
    layoutListeSequenceSpectacle->addWidget(labelsSequencesSpectacle.last());
    Sequence sequence;
    sequence.nom = choixSequencesSpectacle->currentText();
    sequence.uuid = sequences[choixSequencesSpectacle->currentIndex()].uuid;
    cacheSpectacle.ensembleSequences.append(sequence);

}
```

8.12.3.10 void IHM : :bloquerChoixNomCanaux (int *valeur*) [slot]

Paramètres

valeur	int

Renvoie

void

Références nomsCanaux.

Référencé par genererFenetreNouveauProjecteur(), ouvrirFenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

```
{
    for(int i = 0; i < 10; i++)
    {
        nomsCanaux[i]->setEnabled(false);
    }
    for(int i = 0; i < valeur; i++)
    {
        nomsCanaux[i]->setEnabled(true);
    }
}
```

8.12.3.11 void IHM : :changerCanalDepart (int *canal*) [slot]

Paramètres

canal	

Références canalDepartFaders, envoyerCanalDepartWing(), NB_CANAUX, pilotageDMXActif, et setCanal().

Référencé par IHM().

```
{
    pilotageDMXActif = false;
    canalDepartFaders = canal % NB_CANAUX;
    setCanal();
    pilotageDMXActif = true;
    envoyerCanalDepartWing();

}
```

8.12.3.12 void IHM : :changerPilotageIHM (QString *choix*) [slot]

Paramètres

choix	

Renvoie

void

Références activerPilotageIHM(), et desactiverPilotageIHM().

Référencé par IHM().

```
{
    //qDebug() << Q_FUNC_INFO << choix;

    if(choix == QString::fromUtf8("IHM"))
    {
        activerPilotageIHM();
```

```
    }
    else if(choix == QString::fromUtf8("Console"))
    {
        desactiverPilotageIHM();
    }
}
```

8.12.3.13 void IHM : :creerIHM ( ) [private]

Références actionQuitter, XMLUtilitaire : :afficherProjecteursEnregistres(), boutonAjouterConsole, boutonAjouterInterface, boutonchoixSequenceSpectacle, boutonDecrementationCanal, boutonDetecterInterface, boutonexecuterScene, boutonExecuterSequence, boutonExecuterSpectacle, boutonIncrementationCanal, boutonModifierConsole, boutonModifierScene, boutonNouveauProjecteur, boutonNouvelleScene, boutonNouvelleSequence, boutonSauvegarderSpectacle, boutonSupprimerConsole, boutonSupprimerInterface, boutonSupprimerScene, boutonSupprimerSpectacle, boutonSupprimerTousProjecteurs, boutonSupprimerToutesScenes, boutonSupprimmerSequence, canalDepartFaders, champAdresseIPConsole, champPortConsole, choixNomSpectacle, choixScenesDisponibles, choixSequencesDisponibles, choixSequencesSpectacle, choixSpectaclesDisponibles, DMX_USB_PRO, espace, PlaybackWing : :getAdresseIPConsole(), PlaybackWing : :getNombreConsoles(), interfaces, labelInterface, labelnomSpectacle, labelScenesDisponibles, labelSequencesDisponibles, labelSequencesSpectacle, labelSpectaclesDisponibles, layoutListeProjecteursParametres, layoutListeSequenceSpectacle, XMLUtilitaire : :lireScenes(), XMLUtilitaire : :lireSequences(), XMLUtilitaire : :lireSpectacles(), listeChoixConsole, listeChoixInterface, listeChoixPilotage, listeInterfacesSupprimables, listeProjecteursEnregistres, listeProjecteursPilotage, listeSequencesSpectacle, listeTypeInterface, listeTypeInterfaceAjouterInterface, mettreAJourProjecteursEnregistres(), NB_SLIDERS, ongletsPrincipaux, OPEN_DMX_USB, optionsCreations, optionsParametres, optionsSpectacles, pageAssembler, pageCreer, pageJouer, pagePiloter, pagesSpectacles, parametresConsoles, parametresInterface, parametresProjecteurs, portInterfaceAjouterInterface, projecteursDMX, projecteursEnregistres, resultatDetectionInterface, scenes, sequences, MySlider : :setCanal(), setCanal(), MySlider : :setCanalEnabled(), slidersCanaux, spectacles, utilitaireDocuments, et wing.

Référencé par IHM().

```
{
    // Crée les onglets
    ongletsPrincipaux = new QTabWidget(this);
    optionsCreations= new QTabWidget(this);
    optionsParametres= new QTabWidget(this);
    optionsSpectacles= new QTabWidget(this);

    // Crée les pages
    pageAssembler = new QWidget(this);
    pageCreer = new QWidget(this);
    pageJouer = new QWidget(this); // 3
    pagePiloter = new QWidget(this); // 2
    parametresInterface = new QWidget(this);
    parametresProjecteurs = new QWidget(this);
    parametresConsoles = new QWidget(this);
    pagesSpectacles[0] = new QWidget(this);
    pagesSpectacles[1] = new QWidget(this);

    // Place les pages dans les onglets
    ongletsPrincipaux->addTab(optionsCreations, QString::fromUtf8("Créer un
       spectacle"));
    ongletsPrincipaux->addTab(optionsSpectacles, QString::fromUtf8("Jouer un
```

```
    spectacle"));
ongletsPrincipaux->addTab(pagePiloter, QString::fromUtf8("Piloter les
    projecteurs"));
ongletsPrincipaux->addTab(optionsParametres, QString::fromUtf8("Configurer
    le système"));
optionsCreations->addTab(pageCreer, QString::fromUtf8("Créer les scènes"));
optionsCreations->addTab(pageAssembler, QString::fromUtf8("Assembler les
    scènes/Créer des Séquences"));
optionsParametres->addTab(parametresInterface, QString::fromUtf8("
  Paramétrer l'interface"));
optionsParametres->addTab(parametresProjecteurs, QString::fromUtf8("
  Paramétrer les projecteurs"));
optionsParametres->addTab(parametresConsoles, QString::fromUtf8("Paramétrer
    les consoles"));
optionsSpectacles->addTab(pagesSpectacles[0], QString::fromUtf8("Création
    de Spectacles"));
optionsSpectacles->addTab(pagesSpectacles[1], QString::fromUtf8("Jouer un
    Spectacle"));

// Crée les widgets
listeProjecteursEnregistres = new QScrollArea(this);
boutonNouveauProjecteur = new QPushButton(QString::fromUtf8("Nouveau
    Projecteur"), this);
boutonSupprimerTousProjecteurs = new QPushButton(QString::fromUtf8("Tout
    supprimer"), this);
for (int i = 0; i < NB_SLIDERS ; ++i)
{
    slidersCanaux[i] = new MySlider(this);
    if(i > 0)
        slidersCanaux[i]->setCanalEnabled(false);
}
slidersCanaux[0]->setCanal(canalDepartFaders);
listeChoixPilotage = new QComboBox(this);
listeChoixPilotage->addItem(QString::fromUtf8("IHM"));
listeChoixPilotage->addItem(QString::fromUtf8("Console"));
labelInterface = new QLabel(this);
labelInterface->setText("Interface : ");
listeProjecteursPilotage = new QComboBox(this);
listeProjecteursPilotage->addItem(QString::fromUtf8(""));
for(int i = 0; i < projecteursDMX.count(); i++)
{
     listeProjecteursPilotage->addItem(projecteursDMX.at(i)->getNom() + " ("
    + projecteursDMX.at(i)->getType() + ")");
}
boutonIncrementationCanal = new QPushButton(QString::fromUtf8(">>"), this);
boutonDecrementationCanal = new QPushButton(QString::fromUtf8("<<"), this);
setCanal();
listeChoixInterface = new QComboBox(this);
//listeChoixInterface->addItem(QString::fromUtf8(""));
for(int i = 0; i < interfaces.count(); i++)
{
    if(interfaces.at(i).typeInterface == OPEN_DMX_USB)
        listeChoixInterface->addItem(interfaces.at(i).portInterface);
    else if(interfaces.at(i).typeInterface == DMX_USB_PRO)
        listeChoixInterface->addItem(interfaces.at(i).portInterface);
    else
        listeChoixInterface->addItem(interfaces.at(i).portInterface);
}
listeTypeInterface = new QComboBox(this);
listeTypeInterface->addItem(QString::fromUtf8("OPEN DMX USB"));
listeTypeInterface->addItem(QString::fromUtf8("DMX USB PRO"));
if(interfaces.count() > 0)
{
    if(interfaces.at(0).typeInterface == OPEN_DMX_USB)
        listeTypeInterface->setCurrentIndex((int)OPEN_DMX_USB);
    else if(interfaces.at(0).typeInterface == DMX_USB_PRO)
        listeTypeInterface->setCurrentIndex((int)DMX_USB_PRO);
    else
        listeTypeInterface->setCurrentIndex((int)DMX_USB_PRO);
}
boutonDetecterInterface = new QPushButton(QString::fromUtf8("Detecter
    Interfaces"), this);
boutonDetecterInterface->setFixedSize(boutonDetecterInterface->sizeHint());
resultatDetectionInterface = new QTextEdit(this);
```

```
resultatDetectionInterface->setReadOnly(true);

portInterfaceAjouterInterface = new QLineEdit(this);
portInterfaceAjouterInterface->setPlaceholderText("port Interface");
listeTypeInterfaceAjouterInterface = new QComboBox(this);
listeTypeInterfaceAjouterInterface->addItem(QString::fromUtf8("OPEN DMX USB
  "));
listeTypeInterfaceAjouterInterface->addItem(QString::fromUtf8("DMX USB PRO"
  ));
boutonAjouterInterface = new QPushButton(QString::fromUtf8("Ajouter
   Interface"), this);
listeInterfacesSupprimables = new QComboBox(this);
for(int i = 0; i < interfaces.count(); i++)
{
    listeInterfacesSupprimables->addItem(interfaces[i].portInterface);
}
boutonSupprimerInterface = new QPushButton(QString::fromUtf8("Supprimer
   Interface"), this);


labelScenesDisponibles = new QLabel(QString::fromUtf8("Scènes disponibles :
  "), this);
choixScenesDisponibles = new QComboBox;
int nbScenes = utilitaireDocuments->lireScenes(scenes);
for(int i = 0; i < nbScenes; i++)
{
    choixScenesDisponibles->addItem(scenes.at(i).nom);
}
// TODO : remettre les boutons en enabled quand les methodes seront
   fonctionnelles
boutonexecuterScene = new QPushButton(QString::fromUtf8("Tester"), this);

boutonModifierScene = new QPushButton(QString::fromUtf8("Modifier"), this);
boutonModifierScene->setEnabled(false);
boutonSupprimerScene = new QPushButton(QString::fromUtf8("Supprimer"), this
  );
boutonNouvelleScene = new QPushButton(QString::fromUtf8("Nouvelle scène"),
  this);
boutonSupprimerToutesScenes = new QPushButton(QString::fromUtf8("Supprimer
   toutes les scènes"), this);
boutonSupprimerToutesScenes->setEnabled(false);

labelSequencesDisponibles = new QLabel(QString::fromUtf8("Séquences
   disponibles :"), this);
choixSequencesDisponibles = new QComboBox;
int nbSequences = utilitaireDocuments->lireSequences(sequences);
for(int i = 0; i < nbSequences; i++)
{
    choixSequencesDisponibles->addItem(sequences.at(i).nom);
}
boutonExecuterSequence = new QPushButton(QString::fromUtf8("Tester"), this)
  ;
boutonNouvelleSequence = new QPushButton(QString::fromUtf8("Nouvelle
   Séquence"), this);
boutonSupprimmerSequence = new QPushButton(QString::fromUtf8("Supprimer"),
  this);

labelnomSpectacle = new QLabel(QString::fromUtf8("Nom du nouveau Spectacle
   :"), this);
choixNomSpectacle = new QLineEdit;
labelSequencesSpectacle = new QLabel(QString::fromUtf8("Séquences
   Disponibles :"), this);
choixSequencesSpectacle = new QComboBox;
for(int i = 0; i < nbSequences; i++)
{
    choixSequencesSpectacle->addItem(sequences.at(i).nom);
}
boutonchoixSequenceSpectacle = new QPushButton(QString::fromUtf8("Choisir
   cette Séquence"), this);
listeSequencesSpectacle = new QScrollArea;
boutonSauvegarderSpectacle = new QPushButton(QString::fromUtf8("Sauvegarder
  "), this);
```

```
labelSpectaclesDisponibles = new QLabel(QString::fromUtf8("Spectacles
   disponibles :"), this);
choixSpectaclesDisponibles = new QComboBox();
espace = new QSpacerItem(0,150);
int nbSpectacles = utilitaireDocuments->lireSpectacles(spectacles);
for(int i = 0; i < nbSpectacles; i++)
{
    choixSpectaclesDisponibles->addItem(spectacles.at(i).nom);
}
boutonExecuterSpectacle = new QPushButton(QString::fromUtf8("Jouer"), this)
  ;
boutonSupprimerSpectacle = new QPushButton(QString::fromUtf8("Supprimer"),
  this);


listeChoixConsole = new QComboBox(this);
for(int i = 0; i < wing->getNombreConsoles(); i++)
{
    listeChoixConsole->addItem(wing->getAdresseIPConsole(i));
}

champAdresseIPConsole = new QLineEdit(this);
champAdresseIPConsole->setPlaceholderText("Adresse IP");
champPortConsole = new QLineEdit(this);
champPortConsole->setPlaceholderText("Port");
boutonAjouterConsole = new QPushButton(QString::fromUtf8("Ajouter Console")
  , this);
boutonSupprimerConsole = new QPushButton(QString::fromUtf8("Supprimer
   Console"), this);
boutonModifierConsole = new QPushButton(QString::fromUtf8("Modifier Console
  "), this);

// Crée le positionnement
QHBoxLayout *mainLayout = new QHBoxLayout;
QVBoxLayout *mainLayoutParametres = new QVBoxLayout;
QVBoxLayout *mainLayoutPilotage = new QVBoxLayout;
layoutListeProjecteursParametres = new QVBoxLayout;

QHBoxLayout *layoutOptionsParametres = new QHBoxLayout;

QHBoxLayout *layoutListePilotage = new QHBoxLayout;
QHBoxLayout *layoutListeProjecteursPilotage = new QHBoxLayout;

QHBoxLayout *layoutFadersPilotage = new QHBoxLayout;
QHBoxLayout *layoutBoutonsPilotage = new QHBoxLayout;

QVBoxLayout *layoutParametrerInterface = new QVBoxLayout;
QHBoxLayout *layoutChoixInterface = new QHBoxLayout;
QVBoxLayout *layoutDetectionInterface = new QVBoxLayout;

QHBoxLayout *layoutAjouterInterface = new QHBoxLayout;

QHBoxLayout *layoutSupprimerInterface = new QHBoxLayout;

QVBoxLayout *mainLayoutScenes = new QVBoxLayout;
QHBoxLayout *layoutScenesDisponibles = new QHBoxLayout;
QHBoxLayout *layoutBoutonsOptionsScenes = new QHBoxLayout;
QHBoxLayout *layoutBoutonsNouvelleScene = new QHBoxLayout;

QVBoxLayout *layoutParametrerConsoles = new QVBoxLayout;
QHBoxLayout *layoutModificationConsole = new QHBoxLayout;
QHBoxLayout *layoutAjoutConsole = new QHBoxLayout;

QVBoxLayout *layoutSequences = new QVBoxLayout;
QHBoxLayout *layoutchoixSequencesDisponibles = new QHBoxLayout;
QHBoxLayout *layoutoptionsSequencesDisponibles = new QHBoxLayout;

//QHBoxLayout *mainLayoutSpectacles = new QHBoxLayout;
QVBoxLayout *layoutCreationSpectacles = new QVBoxLayout;
QVBoxLayout *layoutJeuSpectacles = new QVBoxLayout;
QHBoxLayout *layoutOptionsSpectacles = new QHBoxLayout;
layoutListeSequenceSpectacle = new QVBoxLayout;

listeProjecteursEnregistres->setLayout(layoutListeProjecteursParametres);
```

```
layoutOptionsParametres->addWidget(boutonNouveauProjecteur);
layoutOptionsParametres->addWidget(boutonSupprimerTousProjecteurs);
mainLayoutParametres->addWidget(listeProjecteursEnregistres);
mainLayoutParametres->addLayout(layoutOptionsParametres);
parametresProjecteurs->setLayout(mainLayoutParametres);

utilitaireDocuments->afficherProjecteursEnregistres(projecteursEnregistres)
  ;

mettreAJourProjecteursEnregistres();

QLabel *labelListePilotage = new QLabel(QString::fromUtf8("Pilotage :"),
  this);
layoutListePilotage->addWidget(labelListePilotage);
layoutListePilotage->addWidget(listeChoixPilotage);
layoutListePilotage->addStretch();
QLabel *labelListeProjecteursPilotage = new QLabel(QString::fromUtf8("
  Projecteur :"), this);
layoutListeProjecteursPilotage->addWidget(labelListeProjecteursPilotage);
layoutListeProjecteursPilotage->addWidget(listeProjecteursPilotage);
layoutListeProjecteursPilotage->addStretch();
mainLayoutPilotage->addLayout(layoutListePilotage);
mainLayoutPilotage->addLayout(layoutListeProjecteursPilotage);
for (int i = 0; i < NB_SLIDERS ; ++i)
{
   layoutFadersPilotage->addWidget(slidersCanaux[i]);
}
mainLayoutPilotage->addLayout(layoutFadersPilotage);
layoutBoutonsPilotage->addWidget(boutonDecrementationCanal);
layoutBoutonsPilotage->addWidget(boutonIncrementationCanal);
mainLayoutPilotage->addLayout(layoutBoutonsPilotage);
pagePiloter->setLayout(mainLayoutPilotage);

mainLayout->addWidget(ongletsPrincipaux);

parametresInterface->setLayout(layoutParametrerInterface);
layoutParametrerInterface->addLayout(layoutChoixInterface);
layoutChoixInterface->addWidget(labelInterface);
layoutChoixInterface->addWidget(listeChoixInterface);
layoutChoixInterface->addSpacing(20);
layoutChoixInterface->addWidget(listeTypeInterface);
labelInterface->setFixedSize(70, 15);
layoutDetectionInterface->addWidget(boutonDetecterInterface);
layoutDetectionInterface->addWidget(resultatDetectionInterface);
layoutDetectionInterface->addStretch();
layoutParametrerInterface->addLayout(layoutDetectionInterface);

layoutParametrerInterface->addLayout(layoutAjouterInterface);
layoutAjouterInterface->addWidget(portInterfaceAjouterInterface);
layoutAjouterInterface->addWidget(listeTypeInterfaceAjouterInterface);
layoutAjouterInterface->addWidget(boutonAjouterInterface);

layoutParametrerInterface->addLayout(layoutSupprimerInterface);
layoutSupprimerInterface->addWidget(listeInterfacesSupprimables);
layoutSupprimerInterface->addWidget(boutonSupprimerInterface);
pageCreer->setLayout(mainLayoutScenes);
mainLayoutScenes->addLayout(layoutScenesDisponibles);
mainLayoutScenes->addLayout(layoutBoutonsOptionsScenes);
mainLayoutScenes->addLayout(layoutBoutonsNouvelleScene);
layoutScenesDisponibles->addWidget(labelScenesDisponibles);
layoutScenesDisponibles->addWidget(choixScenesDisponibles);
layoutBoutonsOptionsScenes->addWidget(boutonexecuterScene);
layoutBoutonsOptionsScenes->addWidget(boutonModifierScene);
layoutBoutonsOptionsScenes->addWidget(boutonSupprimerScene);
layoutBoutonsNouvelleScene->addWidget(boutonNouvelleScene);
layoutBoutonsNouvelleScene->addWidget(boutonSupprimerToutesScenes);

parametresConsoles->setLayout(layoutParametrerConsoles);
layoutParametrerConsoles->addLayout(layoutAjoutConsole);
layoutParametrerConsoles->addLayout(layoutModificationConsole);
layoutAjoutConsole->addWidget(champAdresseIPConsole);
layoutAjoutConsole->addWidget(champPortConsole);
layoutAjoutConsole->addWidget(boutonAjouterConsole);
layoutModificationConsole->addWidget(listeChoixConsole);
```

```
    layoutModificationConsole->addWidget(boutonModifierConsole);
    layoutModificationConsole->addWidget(boutonSupprimerConsole);

    pageAssembler->setLayout(layoutSequences);
    layoutSequences->addLayout(layoutchoixSequencesDisponibles);
    layoutSequences->addLayout(layoutoptionsSequencesDisponibles);
    layoutchoixSequencesDisponibles->addWidget(labelSequencesDisponibles);
    layoutchoixSequencesDisponibles->addWidget(choixSequencesDisponibles);
    layoutoptionsSequencesDisponibles->addWidget(boutonExecuterSequence);
    layoutoptionsSequencesDisponibles->addWidget(boutonNouvelleSequence);
    layoutoptionsSequencesDisponibles->addWidget(boutonSupprimmerSequence);

    pagesSpectacles[0]->setLayout(layoutCreationSpectacles);
    pagesSpectacles[1]->setLayout(layoutJeuSpectacles);
    layoutCreationSpectacles->addWidget(labelnomSpectacle);
    layoutCreationSpectacles->addWidget(choixNomSpectacle);
    layoutCreationSpectacles->addWidget(labelSequencesSpectacle);
    layoutCreationSpectacles->addWidget(choixSequencesSpectacle);
    layoutCreationSpectacles->addWidget(boutonchoixSequenceSpectacle);
    layoutCreationSpectacles->addWidget(listeSequencesSpectacle);
    layoutCreationSpectacles->addWidget(boutonSauvegarderSpectacle);

    layoutJeuSpectacles->addWidget(labelSpectaclesDisponibles);
    layoutJeuSpectacles->addWidget(choixSpectaclesDisponibles);
    layoutJeuSpectacles->addSpacerItem(espace);
    layoutJeuSpectacles->addLayout(layoutOptionsSpectacles);

    layoutOptionsSpectacles->addWidget(boutonExecuterSpectacle);
    layoutOptionsSpectacles->addWidget(boutonSupprimerSpectacle);

    listeSequencesSpectacle->setLayout(layoutListeSequenceSpectacle);

    setLayout(mainLayout);
    this->setFixedSize(QSize(800,500));
    this->setWindowTitle(QString::fromUtf8("DMX 2018"));

    actionQuitter = new QAction("&Quitter", this);
    actionQuitter->setShortcut(QKeySequence(QKeySequence::Quit)); // Ctrl+Q
    addAction(actionQuitter);
}
```

8.12.3.14 void IHM : :decrementerCanal ( ) [slot]

Renvoie

void

Références canalDepartFaders, envoyerCanalDepartWing(), NB_CANAUX, NB_SLIDERS, pilotageDMXActif, et setCanal().

Référencé par afficherEtatTouchesControle(), et IHM().

```
{
    pilotageDMXActif = false;
    //canalDepartFaders--;
    canalDepartFaders = ((canalDepartFaders + NB_CANAUX) - NB_SLIDERS) %
      NB_CANAUX;
    qDebug() << "Canal depart =" << canalDepartFaders;
    setCanal();
    pilotageDMXActif = true;
    envoyerCanalDepartWing();
}
```

8.12.3.15 void IHM : :demarrer ( ) [slot]

Renvoie

void

Références PlaybackWing : :demarrer(), et wing.

```
{
    //qDebug() << Q_FUNC_INFO;
    wing->demarrer();
}
```

8.12.3.16 void IHM : :desactiverPilotageConsole ( ) [private]

Renvoie

void

Références afficherEtatFaders(), et wing.

Référencé par activerPilotageIHM().

```
{
    disconnect(wing, SIGNAL(envoyerEtatFaders(EtatFaders)), this, SLOT(
      afficherEtatFaders(EtatFaders)));
}
```

8.12.3.17 void IHM : :desactiverPilotageIHM ( ) [private]

Renvoie

void

Références activerPilotageConsole(), boutonDecrementationCanal, boutonIncrementation-Canal, listeProjecteursPilotage, NB_SLIDERS, pilotageDMXActif, pilotageIHMActif, et slidersCanaux.

Référencé par afficherEtatTouchesControle(), et changerPilotageIHM().

```
{
    pilotageIHMActif = false;

    for(int i = 0; i < NB_SLIDERS ; i++)
    {
        slidersCanaux[i]->setEnabled(false);
    }
    boutonIncrementationCanal->setEnabled(false);
    boutonDecrementationCanal->setEnabled(false);
    listeProjecteursPilotage->setEnabled(false);

    activerPilotageConsole();

    pilotageDMXActif = true;
}
```

8.12.3.18 void IHM : :detecterInterface ( ) [slot]

Renvoie

void

Références EnttecDMXUSB : :closePort(), DMX_USB_PRO, executerCommandeSysteme(), EnttecDMXUSB : :GetFirmware_H(), EnttecDMXUSB : :GetFirmware_L(), EnttecDMXUSB : :GetPortInterface(), interfaceDMX, interfaces, EnttecDMXUSB : :IsAvailable(), EnttecDMXUSB : :IsOpen(), listeChoixInterface, listeTypeInterface, EnttecDMXUSB : :openPort(), resultatDetectionInterface, et Interface : :typeInterface.

Référencé par IHM().

```
{
    int numeroInterface = listeChoixInterface->currentIndex();
    EnttecInterfaces type = (EnttecInterfaces)listeTypeInterface->currentIndex(
      );
    EnttecDMXUSB *interfaceEnttec;
    bool statut = false;

    if(interfaces.count() == 0)
        return;

    #ifdef Q_OS_WIN
        if(interfaceDMX != NULL && interfaceDMX->IsOpen())
        {
            interfaceDMX->closePort();
            statut = true;
        }
        //resultatDetectionInterface->append(executerCommandeSysteme("devcon
       find *").join("\n"));
        resultatDetectionInterface->append("Interfaces disponibles : ");
        resultatDetectionInterface->append(executerCommandeSysteme("mode").join
      ("\n"));
        if(statut)
            interfaceDMX->openPort(interfaceDMX->GetPortInterface());
    #elif defined(Q_OS_UNIX)
        resultatDetectionInterface->append(executerCommandeSysteme("lsusb |
       grep -i serial").join("\n"));
        resultatDetectionInterface->append("Interfaces disponibles : ");
        resultatDetectionInterface->append(executerCommandeSysteme("ls
       /dev/ttyUSB*").join("\n"));
    #endif

    interfaceEnttec = new EnttecDMXUSB(type, interfaces.at(numeroInterface).
      portInterface.toLocal8Bit().data());
    if(interfaceDMX != NULL && interfaceDMX->IsAvailable())
    {
        if((EnttecInterfaces)listeTypeInterface->currentIndex() == DMX_USB_PRO)
        {
            resultatDetectionInterface->append(QString::fromUtf8("Interface
       Enttec DMX USB PRO séléctionnée détectée\n") + QString::fromUtf8("Version : ") +
      QString::number(interfaceEnttec->GetFirmware_H()) + "." + QString::number(
      interfaceEnttec->GetFirmware_L()) + "\n");
        }
        else
        {
            resultatDetectionInterface->append(QString::fromUtf8("Interface
       Enttec OPEN DMX USB séléctionnée détectée \n"));
        }
        Interface intf = interfaces.at(numeroInterface);
        intf.typeInterface = type;
        interfaces.replace(numeroInterface, intf);
    }
    else
    {
        resultatDetectionInterface->append(QString::fromUtf8("Interface Enttec
       non détectée \n"));
    }

    delete interfaceEnttec;
}
```

8.12.3.19 IHM : :effacerAffichageProjecteursEnregistres ( ) [slot]

Efface les projecteurs en index dans listeProjecteursPilotage.

Renvoie

void

Références XMLUtilitaire : :getNbAppareils(), projecteursEnregistres, et utilitaireDocuments.

Référencé par enregistrerProjecteur(), supprimerProjecteur(), et supprimerTousProjecteurs().

```
{
    qDebug() << Q_FUNC_INFO;
    for(int i = 0; i < utilitaireDocuments->getNbAppareils(); i++)
    {
        delete projecteursEnregistres[i];
    }

}
```

8.12.3.20 void IHM : :effacerAffichageProjecteursPilotage ( ) [slot]

Références listeProjecteursPilotage, et projecteursDMX.

Référencé par enregistrerProjecteur(), supprimerProjecteur(), et supprimerTousProjecteurs().

```
{
    for(int i = projecteursDMX.count(); i > 0 ; i-- )
    {
        listeProjecteursPilotage->removeItem(i);
    }
}
```

8.12.3.21 void IHM : :enregistrerProjecteur ( ) [slot]

Renvoie

void

Références XMLUtilitaire : :afficherProjecteursEnregistres(), cacheUuid, choixCanalDepart, choixNBCanaux, choixNom, choixTypes, effacerAffichageProjecteursEnregistres(), effacerAffichageProjecteursPilotage(), XMLUtilitaire : :enregistrerAppareil(), fenetreNouveauProjecteur, XMLUtilitaire : :lireAppareils(), mettreAJourProjecteursEnregistres(), mettreAJourProjecteursPilotage(), XMLUtilitaire : :modifierAppareil(), nomsCanaux, projecteursDMX, projecteursEnregistres, et utilitaireDocuments.

Référencé par IHM().

```
{
    QString nomsCanauxEnregistres[10];
    qDebug() << Q_FUNC_INFO;
    for(int i = 0; i < 10; i++ )
    {
```

```
        nomsCanauxEnregistres[i] = nomsCanaux[i]->text();
    }
    effacerAffichageProjecteursPilotage();
    effacerAffichageProjecteursEnregistres();

    if(fenetreNouveauProjecteur->windowTitle() == "Nouvel Appareil DMX" )
        utilitaireDocuments->enregistrerAppareil(choixNom->text(), choixTypes->
      currentText(), choixNBCanaux->value(), choixCanalDepart->value(),
      nomsCanauxEnregistres);

    else if(fenetreNouveauProjecteur->windowTitle() == "Modifier Appareil DMX"
      )
        utilitaireDocuments->modifierAppareil(cacheUuid ,choixNom->text(),
      choixTypes->currentText(), choixNBCanaux->value(), choixCanalDepart->value(),
      nomsCanauxEnregistres);

    else
    {
        qDebug() << Q_FUNC_INFO << "erreur enregistrement!!";
    }
    utilitaireDocuments->lireAppareils(projecteursDMX);
    utilitaireDocuments->afficherProjecteursEnregistres(projecteursEnregistres)
      ;

    mettreAJourProjecteursEnregistres();
    mettreAJourProjecteursPilotage();
    utilitaireDocuments->lireAppareils(projecteursDMX);
    fenetreNouveauProjecteur->close();
}
```

8.12.3.22 void IHM : :enregistrerScene ( ) [slot]

Renvoie

void

Références cacheScene, choixNomScene, choixScenesDisponibles, XMLUtilitaire-: :enregistrerScene(), fenetreNouvelleScene, XMLUtilitaire : :lireScenes(), Scene : :nom, scenes, utilitaireDocuments, Scene : :uuid, et Scene : :valeursCanauxscene.

Référencé par genererFenetreNouvelleScene().

```
{
    qDebug() << Q_FUNC_INFO;
    cacheScene.nom = choixNomScene->text();
    cacheScene.uuid = QUuid::createUuid().toString();
    utilitaireDocuments->enregistrerScene(cacheScene);
    choixScenesDisponibles->addItem(cacheScene.nom);
    cacheScene.nom = "";
    cacheScene.uuid = "";
    cacheScene.valeursCanauxscene.clear();
    utilitaireDocuments->lireScenes(scenes);

    fenetreNouvelleScene->close();
}
```

8.12.3.23 void IHM : :enregistrerSequence ( ) [slot]

Références cacheSequence, choixnomSequence, choixSequencesDisponibles, XMLUtilitaire : :enregistrerSequence(), Sequence : :ensembleScenes, fenetreNouvelleSequence, XMLUtilitaire : :lireSequences(), Sequence : :nom, sequences, utilitaireDocuments, et Sequence : :uuid.

Référencé par genererFenetreNouvelleSequence().

```
{
    cacheSequence.nom = choixnomSequence->text();
    cacheSequence.uuid = QUuid::createUuid().toString();
    utilitaireDocuments->enregistrerSequence(cacheSequence);
    choixSequencesDisponibles->addItem(cacheSequence.nom);
    cacheSequence.nom = "";
    cacheSequence.uuid = "";

    cacheSequence.ensembleScenes.clear();

    utilitaireDocuments->lireSequences(sequences);

    fenetreNouvelleSequence->close();

}
```

8.12.3.24 void IHM : :enregistrerSpectacle ( ) [slot]

Références cacheSpectacle, choixNomSpectacle, choixSpectaclesDisponibles, XMLUtilitaire : :ecrireSpectacles(), labelsSequencesSpectacle, XMLUtilitaire : :lireSpectacles(), spectacles, et utilitaireDocuments.

Référencé par IHM().

```
{
   qDebug() << Q_FUNC_INFO;
   cacheSpectacle.nom = choixNomSpectacle->text();
   cacheSpectacle.uuid = QUuid::createUuid().toString();
   choixSpectaclesDisponibles->addItem(cacheSpectacle.nom);
   spectacles.append(cacheSpectacle);

   utilitaireDocuments->ecrireSpectacles(spectacles);
   utilitaireDocuments->lireSpectacles(spectacles);
   choixNomSpectacle->setText("");
   for(int i =0; i < cacheSpectacle.ensembleSequences.size(); i++)
   {
      delete labelsSequencesSpectacle.at(i);
   }
   labelsSequencesSpectacle.clear();
   cacheSpectacle.ensembleSequences.clear();
   cacheSpectacle.nom = "";
   cacheSpectacle.uuid = "";

}
```

8.12.3.25 void IHM : :envoyerCanalDepartWing ( ) [slot]

Références PlaybackWing : :envoyerCanalDepartWing(), slidersCanaux, et wing.

Référencé par changerCanalDepart(), decrementerCanal(), incrementerCanal(), et selectionnerProjecteursPilotage().

```
{
    //qDebug() << Q_FUNC_INFO;
    wing->envoyerCanalDepartWing(slidersCanaux[0]->getCanal());
}
```

8.12.3.26 void IHM : :envoyerModificationConsole ( ) [slot]

Références Console : :adresseIP, champAdresseIPModifierConsole, champPortModifierConsole, PlaybackWing : :enregistrerModificationConsole(), fenetreModifierConsole, PlaybackWing : :getAdresseIPConsole(), PlaybackWing : :getNombreConsoles(), PlaybackWing : :getPort(), listeChoixConsole, Console : :port, et wing.

Référencé par genererFenetreModifierConsole().

```
{
    Console console;
    if(champAdresseIPModifierConsole->text() == "")
    {
        console.adresseIP = wing->getAdresseIPConsole(listeChoixConsole->
      currentIndex());
        //qDebug() << "adresse IP console = " << console.adresseIP;
    }
    else console.adresseIP = champAdresseIPModifierConsole->text();
    if(champPortModifierConsole->text() == "")
    {
        console.port = wing->getPort(listeChoixConsole->currentIndex());
        //qDebug() << "port console = " << console.port;
    }
    else console.port = champPortModifierConsole->text().toInt();

    wing->enregistrerModificationConsole(console, listeChoixConsole->
      currentIndex());

    fenetreModifierConsole->close();
    listeChoixConsole->clear();
    for(int i = 0; i < wing->getNombreConsoles(); i++)
    {
        //qDebug()<< wing->getNombreConsoles();
        listeChoixConsole->addItem(wing->getAdresseIPConsole(i));
    }
}
```

8.12.3.27 void IHM : :envoyerTrameDMX (int *valeurFader,* int *valeurCanal*) [slot]

Paramètres

valeurCanal	int le numéro de canal
valeurFader	int la valeur associée au canal sélectionné

Renvoie

void

Références interfaceDMX, pilotageDMXActif, EnttecDMXUSB : :SendDMX(), et EnttecDMXUSB : :SetCanalDMX().

Référencé par IHM().

```
{
    if(pilotageDMXActif)
    {
        if(interfaceDMX != NULL)
        {
            interfaceDMX->SetCanalDMX(valeurCanal, valeurFader);
            interfaceDMX->SendDMX();
        }
    }
}
```

8.12.3.28 QStringList IHM : :executerCommandeSysteme (QString *commande*) [private]

Paramètres

commande	QString

Renvoie

QStringList

Référencé par detecterInterface().

```
{
#ifdef Q_OS_WIN
    // Windows
    //wchar_t wNom[256];
    //wchar_t wLigne[1024];
    char wLigne[1024];
    FILE *wResultat;
    QString wReponse;

    QTextCodec *codec = QTextCodec::codecForName("850");
    wResultat = popen(commande.toLocal8Bit().constData(), "r");
    fgets(wLigne, 1024, wResultat);
    while(! feof(wResultat))
    {
        wReponse += codec->toUnicode(wLigne);
        fgets(wLigne, 1024, wResultat);
    }
    pclose(wResultat);

    //qDebug() << Q_FUNC_INFO << wReponse;

    wReponse.chop(1);
    QStringList wReponses = wReponse.split("\n");

    return wReponses;

#elif defined(Q_OS_UNIX)
    // Unix
    FILE *resultat;
    char ligne[1024];
    QString reponse;

    // lit l'état de la connexion
    resultat = popen(commande.toAscii().constData(), "r");
    fgets(ligne, 1024, resultat);
    while (! feof(resultat))
    {
        reponse += ligne;
        fgets(ligne, 1024, resultat);
    }
    pclose(resultat);

    // enlève le dernier \n
    reponse.chop(1);
    QStringList reponses = reponse.split("\n");

    //qDebug() << Q_FUNC_INFO << reponses;

    return reponses;
#  endif
}
```

8.12.3.29 void IHM : :executerScene (Scene *scene*)

Renvoie

void

Références interfaceDMX, EnttecDMXUSB : :SendDMX(), EnttecDMXUSB : :SetCanalDMX(), et Scene : :valeursCanauxscene.

Référencé par executerSceneDefault(), et executerSequence().

```
{
    qDebug() << Q_FUNC_INFO;
    for(int i = 1; i <= 512; i++)
    {

        if(interfaceDMX != NULL)
        {
            interfaceDMX->SetCanalDMX(i, scene.valeursCanauxscene[i]);
        }

    }
    if(interfaceDMX != NULL)
    {
        interfaceDMX->SendDMX();
    }
}
```

8.12.3.30 void IHM : :executerSceneDefault ( ) [slot]

Références choixScenesDisponibles, executerScene(), et scenes.

Référencé par IHM().

```
{
    qDebug() << Q_FUNC_INFO;
    executerScene(scenes[choixScenesDisponibles->currentIndex()]);
}
```

8.12.3.31 void IHM : :executerSequence (Sequence *sequence*)

Références ThreadAttente : :attendre(), Sequence : :ensembleScenes, et executerScene().

Référencé par executerSequenceDefault(), et executerSpectacle().

```
{
    ThreadAttente thread;
    qDebug() << Q_FUNC_INFO << "nombre de scenes :" << sequence.ensembleScenes.
      count();
    for(int i = 0; i <sequence.ensembleScenes.count(); i++)
    {
        executerScene(sequence.ensembleScenes[i].scene);
        thread.attendre(sequence.ensembleScenes[i].tempo);
    }


}
```

8.12.3.32 void IHM : :executerSequenceDefault ( ) [slot]

Références choixSequencesDisponibles, executerSequence(), et sequences.

Référencé par IHM().

```
{
    qDebug() << Q_FUNC_INFO;
    executerSequence(sequences[choixSequencesDisponibles->currentIndex()]);
}
```

8.12.3.33 void IHM : :executerSpectacle ( ) [slot]

Références choixSpectaclesDisponibles, executerSequence(), et spectacles.

Référencé par IHM().

```
{
    qDebug() << Q_FUNC_INFO << "spectacle :" << spectacles[
      choixSpectaclesDisponibles->currentIndex()].nom ;
    qDebug() << Q_FUNC_INFO << spectacles[choixSpectaclesDisponibles->
      currentIndex()].uuid;
    qDebug() << Q_FUNC_INFO << "nombre sequences :" << spectacles[
      choixSpectaclesDisponibles->currentIndex()].ensembleSequences.size();
    for(int i = 0; i < spectacles[choixSpectaclesDisponibles->currentIndex()].
      ensembleSequences.size(); i++)
    {
        executerSequence(spectacles[choixSpectaclesDisponibles->currentIndex()]
      .ensembleSequences[i]);
    }
}
```

8.12.3.34 void IHM : :genererFenetreCanauxScene ( ) [slot]

Renvoie

void

Références ajouterAppareilScene(), boutonSauvegarderCanauxScene, choixAppareilScene, fenetreCanauxScene, DMXProjecteur : :getCanalDebut(), DMXProjecteur : :getNomCanal(), DMXProjecteur : :getNomCanaux(), NB_SLIDERS, pilotageDMXActif, projecteursDMX, MySlider : :setCanal(), MySlider : :setCanalEnabled(), MySlider : :setConnecte(), MySlider : :setNomCanal(), et slidersCanauxScene.

Référencé par ouvrirFenetreCanauxScene().

```
{
   // qDebug() << Q_FUNC_INFO;
    fenetreCanauxScene = new QDialog;
    fenetreCanauxScene->setWindowTitle(QString::fromUtf8("Paramètres des canaux
       DMX du projecteur à ajouter"));



    boutonSauvegarderCanauxScene = new QPushButton(QString::fromUtf8("Confirmer
       cette Configuration de canaux"), this);

    QVBoxLayout *mainLayoutCanauxScene = new QVBoxLayout;
    QHBoxLayout *layoutSlidersCanauxScene = new QHBoxLayout;

    fenetreCanauxScene->setLayout(mainLayoutCanauxScene);
    mainLayoutCanauxScene->addLayout(layoutSlidersCanauxScene);
    mainLayoutCanauxScene->addWidget(boutonSauvegarderCanauxScene);

    for(int i = 0; i < NB_SLIDERS; i++)
    {
        slidersCanauxScene[i] = new MySlider();
        layoutSlidersCanauxScene->addWidget(slidersCanauxScene[i]);

    }

    DMXProjecteur *projecteur = projecteursDMX.at(choixAppareilScene->
      currentIndex());
    pilotageDMXActif = false;
    for(int i = 0; i < NB_SLIDERS; i ++)
    {
        if(i >= projecteur->getNomCanaux().size())
```

```
        {
            slidersCanauxScene[i]->hide();
        }
        else
        {
            slidersCanauxScene[i]->setConnecte(false);
            slidersCanauxScene[i]->setCanalEnabled(false);
            slidersCanauxScene[i]->setCanal(projecteur->getCanalDebut() + i);
            slidersCanauxScene[i]->setNomCanal(projecteur->getNomCanal(
      projecteur->getCanalDebut() + i));
            slidersCanauxScene[i]->show();
            slidersCanauxScene[i]->setConnecte(true);
        }
    }
    connect(boutonSauvegarderCanauxScene, SIGNAL(clicked()),this,SLOT(
      ajouterAppareilScene()));
    //qDebug() << Q_FUNC_INFO << "fenetre generee" ;
}
```

8.12.3.35 void IHM : :genererFenetreInformationsAppareil ( ) [slot]

Renvoie

void

Références fenetreInformationsAppareil, labelInformationsCanalDepart, labelInformationsCanaux, et listeInformationsCanaux.

Référencé par IHM().

```
{
    fenetreInformationsAppareil = new QDialog;
    labelInformationsCanalDepart = new QLabel;
    for(int i=0; i < 10; i++ )
    {
        labelInformationsCanaux[i] = new QLabel;
    }
    listeInformationsCanaux = new QScrollArea;

    QVBoxLayout *mainLayoutInformations = new QVBoxLayout;
    //QVBoxLayout *layoutListeInformationsCanaux = new QVBoxLayout;

    fenetreInformationsAppareil->setLayout(mainLayoutInformations);

    mainLayoutInformations->addWidget(labelInformationsCanalDepart);
    //mainLayoutInformations->addWidget(listeInformationsCanaux);

    //listeInformationsCanaux->setLayout(layoutListeInformationsCanaux);
    for(int i=0; i < 10; i++ )
    {
         mainLayoutInformations->addWidget(labelInformationsCanaux[i]);
    }
    fenetreInformationsAppareil->setFixedWidth(300);
}
```

8.12.3.36 void IHM : :genererFenetreModifierConsole ( ) [slot]

Renvoie

void

Références boutonEnregistrerModificationConsole, champAdresseIPModifierConsole, champPortModifierConsole, envoyerModificationConsole(), et fenetreModifierConsole.

Référencé par IHM().

```
{
    fenetreModifierConsole = new QDialog;
    fenetreModifierConsole->setWindowTitle("Modifier Console");

    champAdresseIPModifierConsole = new QLineEdit(this);
    champPortModifierConsole = new QLineEdit(this);
    boutonEnregistrerModificationConsole = new QPushButton("Enregistrer", this)
      ;
    champAdresseIPModifierConsole->setPlaceholderText("Adresse IP");
    champPortModifierConsole->setPlaceholderText("Port");

    QHBoxLayout *layoutModificationConsole = new QHBoxLayout;

    fenetreModifierConsole->setLayout(layoutModificationConsole);
    layoutModificationConsole->addWidget(champAdresseIPModifierConsole);
    layoutModificationConsole->addWidget(champPortModifierConsole);
    layoutModificationConsole->addWidget(boutonEnregistrerModificationConsole);

    connect(boutonEnregistrerModificationConsole,SIGNAL(clicked()), this, SLOT(
      envoyerModificationConsole()));
    fenetreModifierConsole->show();
}
```

8.12.3.37 void IHM : :genererFenetreNouveauProjecteur ( ) [slot]

Renvoie

void

Références bloquerChoixNomCanaux(), boutonEnregistrer, choixCanalDepart, choixNBCanaux, choixNom, choixTypes, fenetreNouveauProjecteur, labelCanalDepart, labelNBCanaux, labelNom, labelNomCanaux, labelType, et nomsCanaux.

Référencé par IHM().

```
{
    fenetreNouveauProjecteur = new QDialog;
    fenetreNouveauProjecteur->setWindowTitle("Nouvel Appareil DMX");

    labelNom = new QLabel(QString::fromUtf8("Nom :"), this);
    choixNom = new QLineEdit();

    labelType = new QLabel(QString::fromUtf8("Type :"), this);
    choixTypes = new QComboBox();
    choixTypes->addItem("PAR LED");
    choixTypes->addItem("LYRE");
    choixTypes->addItem("SCANNER");
    choixTypes->addItem("LASER");
    choixTypes->addItem("AUTRES");

    labelNBCanaux = new QLabel(QString::fromUtf8("Nombre de Canaux :"), this);
    choixNBCanaux = new QSpinBox;
    choixNBCanaux->setMinimum(1);
    choixNBCanaux->setMaximum(10);

    labelCanalDepart = new QLabel(QString::fromUtf8("Canal de Depart :"), this)
      ;
    choixCanalDepart = new QSpinBox;
    choixCanalDepart->setMinimum(1);
    choixCanalDepart->setMaximum(512);

    labelNomCanaux = new QLabel(QString::fromUtf8("Noms des Canaux :"), this);
    for(int i = 0; i < 10; i++)
    {
        nomsCanaux[i] = new QLineEdit;
    }

    boutonEnregistrer = new QPushButton(QString::fromUtf8("Enregistrer"), this)
      ;
```

```
    QVBoxLayout *mainLayoutFenetre = new QVBoxLayout;
    QHBoxLayout *layoutEnregistrementNom = new QHBoxLayout;
    QHBoxLayout *layoutEnregistrementType = new QHBoxLayout;
    QHBoxLayout *layoutEnregistrementParametresCanaux = new QHBoxLayout;
    QVBoxLayout *layoutEnregistrementNomsCanaux = new QVBoxLayout;

    layoutEnregistrementNom->addWidget(labelNom);
    layoutEnregistrementNom->addWidget(choixNom);

    layoutEnregistrementType->addWidget(labelType);
    layoutEnregistrementType->addWidget(choixTypes);

    layoutEnregistrementParametresCanaux->addWidget(labelNBCanaux);
    layoutEnregistrementParametresCanaux->addWidget(choixNBCanaux);
    layoutEnregistrementParametresCanaux->addWidget(labelCanalDepart);
    layoutEnregistrementParametresCanaux->addWidget(choixCanalDepart);

    layoutEnregistrementNomsCanaux->addWidget(labelNomCanaux);
    for(int i = 0; i < 10; i++)
    {
        layoutEnregistrementNomsCanaux->addWidget(nomsCanaux[i]);
        connect(choixNBCanaux, SIGNAL(valueChanged(int)),this, SLOT(
      bloquerChoixNomCanaux(int)));
    }

    mainLayoutFenetre->addLayout(layoutEnregistrementNom);
    mainLayoutFenetre->addLayout(layoutEnregistrementType);
    mainLayoutFenetre->addLayout(layoutEnregistrementParametresCanaux);
    mainLayoutFenetre->addLayout(layoutEnregistrementNomsCanaux);
    mainLayoutFenetre->addWidget(boutonEnregistrer);

    fenetreNouveauProjecteur->setLayout(mainLayoutFenetre);


}
```

8.12.3.38 void IHM : :genererFenetreNouvelleScene ( ) [slot]

Renvoie

void

Références boutonParametresCanauxScene, boutonSauvegarderScene, choixAppareilScene, choixNomScene, enregistrerScene(), fenetreNouvelleScene, labelAppareilScene, labelNomScene, layoutlisteAppareilsScene, listeAppareilsScene, et projecteursDMX.

Référencé par IHM().

```
{
    //qDebug() << Q_FUNC_INFO;

    fenetreNouvelleScene = new QDialog;
    fenetreNouvelleScene->setWindowTitle(QString::fromUtf8("Paramètres de la
       nouvelle Scène"));

    labelNomScene = new QLabel(QString::fromUtf8("Nom de la scène:"), this);
    choixNomScene = new QLineEdit(this);
    labelAppareilScene = new QLabel(QString::fromUtf8("Appareils disponibles:")
      , this);
    choixAppareilScene = new QComboBox();
    for(int i = 0; i < projecteursDMX.count(); i++)
    {
    choixAppareilScene->addItem(projecteursDMX.at(i)->getNom() + " (" +
      projecteursDMX.at(i)->getType() + ")");
    }
    boutonParametresCanauxScene = new QPushButton(QString::fromUtf8("Choisir ce
```

```
        projecteur"), this);
    listeAppareilsScene = new QScrollArea();
    boutonSauvegarderScene = new QPushButton(QString::fromUtf8("Sauvegarder
        cette scène"), this);

    QVBoxLayout *mainLayoutNouvelleScene = new QVBoxLayout;
    QHBoxLayout *layoutNomScene = new QHBoxLayout;
    QHBoxLayout *layoutChoixAppareilsScene = new QHBoxLayout;
    layoutlisteAppareilsScene = new QVBoxLayout;

    fenetreNouvelleScene->setLayout(mainLayoutNouvelleScene);
    mainLayoutNouvelleScene->addLayout(layoutNomScene);
    mainLayoutNouvelleScene->addLayout(layoutChoixAppareilsScene);
    mainLayoutNouvelleScene->addWidget(boutonParametresCanauxScene);
    mainLayoutNouvelleScene->addWidget(listeAppareilsScene);
    mainLayoutNouvelleScene->addWidget(boutonSauvegarderScene);

    layoutNomScene->addWidget(labelNomScene);
    layoutNomScene->addWidget(choixNomScene);
    layoutChoixAppareilsScene->addWidget(labelAppareilScene);
    layoutChoixAppareilsScene->addWidget(choixAppareilScene);

    listeAppareilsScene->setLayout(layoutlisteAppareilsScene);
    connect(boutonSauvegarderScene,SIGNAL(clicked()),this,SLOT(enregistrerScene
      ()));

   // qDebug() << Q_FUNC_INFO << "fenetre generee" ;

}
```

8.12.3.39 void IHM : :genererFenetreNouvelleSequence ( ) [slot]

Renvoie

void

Références ajouterSceneSequence(), boutonAjouterScene, boutonEnregistrer-Sequence, choixHeures, choixMinutes, choixnomSequence, choixScenesSequence, choixSecondes, enregistrerSequence(), fenetreNouvelleSequence, labelHeures, labelMinutes, labelnomSequence, labelScenesSequence, labelSecondes, label-Temporisation, layoutListeScenesSequences, listeScenesSequenceChoisies, et scenes.

Référencé par IHM().

```
{
    fenetreNouvelleSequence = new QDialog;
    fenetreNouvelleSequence->setWindowTitle(QString::fromUtf8("Assemblage d'une
        séquence"));

    labelnomSequence = new QLabel(QString::fromUtf8("Nom de la séquence :"),
      this);
    choixnomSequence = new QLineEdit();
    labelScenesSequence = new QLabel(QString::fromUtf8("Scènes disponibles :"),
       this);
    choixScenesSequence = new QComboBox;
    for(int i = 0; i < scenes.count(); i++)
    {
        choixScenesSequence->addItem(scenes.at(i).nom);
    }

    labelTemporisation = new QLabel(QString::fromUtf8("Durée de la scène :"),
      this);
    choixHeures = new QSpinBox();
    labelHeures = new QLabel(QString::fromUtf8("h"), this);
    choixMinutes = new QSpinBox();
    choixMinutes->setMaximum(60);
```

```
    labelMinutes = new QLabel(QString::fromUtf8("min"), this);
    choixSecondes = new QSpinBox();
    choixSecondes->setMaximum(60);
    labelSecondes = new QLabel(QString::fromUtf8("sec"), this);
    boutonAjouterScene = new QPushButton(QString::fromUtf8("Ajouter cette Scène
      "), this);
    listeScenesSequenceChoisies = new QScrollArea;
    boutonEnregistrerSequence = new QPushButton(QString::fromUtf8("Enregistrer"
      ), this);

    QVBoxLayout *mainLayoutNouvelleSequence = new QVBoxLayout;
    QHBoxLayout *layoutNomSequence = new QHBoxLayout;
    QHBoxLayout *layoutScenesSequences = new QHBoxLayout;
    QHBoxLayout *layoutTempsScene = new QHBoxLayout;
    layoutListeScenesSequences = new QVBoxLayout;

    fenetreNouvelleSequence->setLayout(mainLayoutNouvelleSequence);
    mainLayoutNouvelleSequence->addLayout(layoutNomSequence);
    layoutNomSequence->addWidget(labelnomSequence);
    layoutNomSequence->addWidget(choixnomSequence);
    mainLayoutNouvelleSequence->addLayout(layoutScenesSequences);
    layoutScenesSequences->addWidget(labelScenesSequence);
    layoutScenesSequences->addWidget(choixScenesSequence);
    mainLayoutNouvelleSequence->addLayout(layoutTempsScene);
    layoutTempsScene->addWidget(labelTemporisation);
    layoutTempsScene->addWidget(choixHeures);
    layoutTempsScene->addWidget(labelHeures);
    layoutTempsScene->addWidget(choixMinutes);
    layoutTempsScene->addWidget(labelMinutes);
    layoutTempsScene->addWidget(choixSecondes);
    layoutTempsScene->addWidget(labelSecondes);
    mainLayoutNouvelleSequence->addWidget(boutonAjouterScene);
    mainLayoutNouvelleSequence->addWidget(listeScenesSequenceChoisies);
    listeScenesSequenceChoisies->setLayout(layoutListeScenesSequences);
    mainLayoutNouvelleSequence->addWidget(boutonEnregistrerSequence);

    connect(boutonAjouterScene,SIGNAL(clicked()),this,SLOT(ajouterSceneSequence
      ()));
    connect(boutonEnregistrerSequence,SIGNAL(clicked()),this,SLOT(
      enregistrerSequence()));
}
```

8.12.3.40 void IHM : :incrementerCanal ( ) [slot]

Renvoie

void

Références canalDepartFaders, envoyerCanalDepartWing(), NB_CANAUX, NB_SLIDERS, pilotageDMXActif, et setCanal().

Référencé par afficherEtatTouchesControle(), et IHM().

```
{
    pilotageDMXActif = false;
    //canalDepartFaders++;
    canalDepartFaders = (canalDepartFaders + NB_SLIDERS) % NB_CANAUX;
    qDebug() << "Canal depart =" << canalDepartFaders;
    setCanal();
    pilotageDMXActif = true;
    envoyerCanalDepartWing();
}
```

8.12.3.41 void IHM : :mettreAJourProjecteursEnregistres ( ) [slot]

Renvoie

void

Références XMLUtilitaire : :getNbAppareils(), layoutListeProjecteursParametres, ouvrirFenetreInformationsProjecteur(), ouvrirFenetreModifierProjecteur(), projecteursEnregistres, supprimerProjecteur(), et utilitaireDocuments.

Référencé par creerIHM(), enregistrerProjecteur(), et supprimerProjecteur().

```
{
    for(int i = 0; i < utilitaireDocuments->getNbAppareils(); i++)
    {
        layoutListeProjecteursParametres->addWidget(projecteursEnregistres[i]);
        connect(projecteursEnregistres[i], SIGNAL(suppression(QString)), this,
      SLOT(supprimerProjecteur(QString)));
        connect(projecteursEnregistres[i], SIGNAL(modification(QString,QString)
      ), this, SLOT(ouvrirFenetreModifierProjecteur(QString,QString)));
        connect(projecteursEnregistres[i], SIGNAL(informations(QString,QString)
      ), this, SLOT(ouvrirFenetreInformationsProjecteur(QString,QString)));
    }
}
```

8.12.3.42 void IHM : :mettreAJourProjecteursPilotage ( ) [slot]

Renvoie

void

Références listeProjecteursPilotage, et projecteursDMX.

Référencé par enregistrerProjecteur(), et supprimerProjecteur().

```
{

    for(int i = 0; i < projecteursDMX.count(); i++)
    {
        listeProjecteursPilotage->addItem(projecteursDMX.at(i)->getNom() + " ("
       + projecteursDMX.at(i)->getType() + ")");
    }
}
```

8.12.3.43 void IHM : :ouvrirFenetreCanauxScene ( ) [slot]

Renvoie

void

Références choixAppareilScene, fenetreCanauxScene, genererFenetreCanauxScene(), et labelsAppareilsScene.

Référencé par IHM().

```
{
    bool estDejaEnregistre = false;
    for(int i = 0; i < labelsAppareilsScene.size(); i++)
    {
        if(choixAppareilScene->currentText() == labelsAppareilsScene[i]->text()
      )
        {
            estDejaEnregistre = true;
        }
```

```
    }
    if(!estDejaEnregistre)
    {
        genererFenetreCanauxScene();
        fenetreCanauxScene->show();
    }
    else
    {
        QMessageBox::critical(0, QString::fromUtf8("DMX 2018"),
      QString::fromUtf8("L'appareil selectionné est déja présent dans la scène"));
    }
}
```

8.12.3.44 void IHM : :ouvrirFenetreInformationsProjecteur (QString *projecteur,* QString *uuid*) [slot]

Paramètres

projecteur	QString
uuid	Qstring

Renvoie

void

Références fenetreInformationsAppareil, DMXProjecteur : :getCanalDebut(), DMXProjecteur : :getNomCanaux(), labelInformationsCanalDepart, labelInformationsCanaux, et projecteursDMX.

Référencé par mettreAJourProjecteursEnregistres().

```
{
    for(int i = 0; i < projecteursDMX.size(); i++)
    {
        if(projecteursDMX[i]->getUuid() == uuid)
        {
            fenetreInformationsAppareil->setWindowTitle(projecteur +" ("+
      projecteursDMX[i]->getType() +")");
            labelInformationsCanalDepart->setText(QString::fromUtf8("Canal DMX
       de départ :") + QString::number(projecteursDMX[i]->getCanalDebut()));
            DMXProjecteur *projecteurDmx = projecteursDMX[i];

            for(int i = 0; i < 10; i++)
            {
                labelInformationsCanaux[i]->setText("");
            }

            for(int i = 0; i < projecteurDmx->getNomCanaux().size(); i++)
            {
                int canal = projecteurDmx->getCanalDebut() + i;
                labelInformationsCanaux[i]->setText(QString::fromUtf8("Canal N°
       ") + QString::number(canal) + " : " + projecteurDmx->getNomCanaux().at(i));
            }
        }

    }

    fenetreInformationsAppareil->show();
}
```

8.12.3.45 void IHM : :ouvrirFenetreModifierProjecteur (QString *projecteur,* QString *uuid*) [slot]

Paramètres

projecteur	QString
uuid	Qstring

Renvoie

void

Références bloquerChoixNomCanaux(), cacheUuid, choixCanalDepart, choixNBCanaux, choixNom, choixTypes, fenetreNouveauProjecteur, DMXProjecteur : :getNomCanaux(), nomsCanaux, et projecteursDMX.

Référencé par mettreAJourProjecteursEnregistres().

```
{

    fenetreNouveauProjecteur->setWindowTitle("Modifier Appareil DMX");
    cacheUuid = uuid;
    choixNom->setText(projecteur);
    for(int i = 0; i < projecteursDMX.size(); i++)
    {
        if(projecteursDMX[i]->getUuid() == uuid)
        {

            if(projecteursDMX[i]->getType() == "PAR LED")
                choixTypes->setCurrentIndex(0);
            else if(projecteursDMX[i]->getType() == "LYRE")
                choixTypes->setCurrentIndex(1);
            else if(projecteursDMX[i]->getType() == "SCANNER")
                choixTypes->setCurrentIndex(2);
            else if(projecteursDMX[i]->getType() == "LASER")
                choixTypes->setCurrentIndex(3);
            else
                choixTypes->setCurrentIndex(4);

            choixNBCanaux->setValue(projecteursDMX[i]->getNomCanaux().size());
            choixCanalDepart->setValue(projecteursDMX[i]->getCanalDebut());

            DMXProjecteur *projecteurDmx = projecteursDMX[i];

            for(int i = 0; i < projecteurDmx->getNomCanaux().size() ; i++)
            {
                nomsCanaux[i]->setText(projecteurDmx->getNomCanaux().at(i));
            }
            bloquerChoixNomCanaux(choixNBCanaux->value());

        }
    }
    fenetreNouveauProjecteur->show();

}
```

8.12.3.46 void IHM : :ouvrirFenetreNouveauProjecteur ( ) [slot]

Renvoie

void

Références bloquerChoixNomCanaux(), choixCanalDepart, choixNBCanaux, choixNom, choixTypes, fenetreNouveauProjecteur, et nomsCanaux.

Référencé par IHM().

```
{
```

```
    fenetreNouveauProjecteur->setWindowTitle("Nouvel Appareil DMX");
    choixNom->setText("");
    choixTypes->setCurrentIndex(0);
    choixNBCanaux->setValue(0);
    choixCanalDepart->setValue(0);
    for(int i = 0; i < 10; i++)
    {
        nomsCanaux[i]->setText("");
    }
    bloquerChoixNomCanaux(choixNBCanaux->value());
    fenetreNouveauProjecteur->show();
}
```

8.12.3.47 void IHM : :ouvrirFenetreNouvelleScene ( ) [slot]

Renvoie

void

Références choixAppareilScene, choixNomScene, fenetreNouvelleScene, labelsAppareilsScene, et projecteursDMX.

Référencé par IHM().

```
{
    qDebug() << Q_FUNC_INFO << labelsAppareilsScene.count();
    for(int i = 0 ; i < labelsAppareilsScene.count() ; i ++)
    {
            qDebug() << Q_FUNC_INFO << " labelsAppareilsScene.at(" << i <<")";
            delete labelsAppareilsScene.at(i);

    }
    labelsAppareilsScene.clear();
    choixAppareilScene->clear();
    choixNomScene->setText("");
    for(int i = 0; i < projecteursDMX.count(); i++)
    {
    choixAppareilScene->addItem(projecteursDMX.at(i)->getNom() + " (" +
      projecteursDMX.at(i)->getType() + ")");
    }
    fenetreNouvelleScene->show();
}
```

8.12.3.48 void IHM : :ouvrirFenetreNouvelleSequence ( ) [slot]

Renvoie

void

Références choixHeures, choixMinutes, choixnomSequence, choixScenesSequence, choixSecondes, fenetreNouvelleSequence, labelsScenesSequences, et scenes.

Référencé par IHM().

```
{

    for(int i = 0 ; i < labelsScenesSequences.count() ; i ++)
    {
         delete labelsScenesSequences.at(i);
    }
    labelsScenesSequences.clear();
    choixnomSequence->setText("");
    choixHeures->setValue(0);
    choixMinutes->setValue(0);
```

```
    choixSecondes->setValue(0);

    choixScenesSequence->clear();
    for(int i = 0; i < scenes.count(); i++)
    {
        choixScenesSequence->addItem(scenes.at(i).nom);
    }


    fenetreNouvelleSequence->show();
}
```

8.12.3.49 void IHM : :quitter ( ) [slot]

Renvoie

void

Référencé par IHM().

```
{
    close();
}
```

8.12.3.50 void IHM : :recupererDerniereInterfaceUtilisee ( )

Références interfaces, listeChoixInterface, listeTypeInterface, et selectionnerInterface().

Référencé par IHM().

```
{
    for(int i = 0; i < interfaces.count(); i ++)
    {
        //qDebug() << "Interface numero " << i << ", Port : " <<
       interfaces.at(i).portInterface;
        //qDebug() << "Interface numero " << i << ", Utilisee : " <<
       interfaces.at(i).utilisee;
        if(interfaces.at(i).utilisee)
        {
            //qDebug() << "Interface numero" << i << ", Utilisee : " <<
       interfaces[i].utilisee;
            selectionnerInterface(i);
            listeChoixInterface->setCurrentIndex(i);
            listeTypeInterface->setCurrentIndex((int)interfaces[i].
      typeInterface);
        }
    }
}
```

8.12.3.51 void IHM : :recupererDonneesNouvelleInterface ( ) [slot]

Renvoie

void

Références XMLUtilitaire : :ecrireInterfaces(), Interface : :id, interfaces, listeChoix-Interface, listeInterfacesSupprimables, listeTypeInterfaceAjouterInterface, Interface-: :portInterface, portInterfaceAjouterInterface, Interface : :typeInterface, Interface : :utilisee, et utilitaireDocuments.

Référencé par IHM().

```
{
    Interface interfaceAjoutee;

    interfaceAjoutee.id = interfaces.count() + 1;
    interfaceAjoutee.portInterface = portInterfaceAjouterInterface->text();
    interfaceAjoutee.typeInterface = (EnttecInterfaces)
      listeTypeInterfaceAjouterInterface->currentIndex();
    interfaceAjoutee.utilisee = 0;

    interfaces.push_back(interfaceAjoutee);

    utilitaireDocuments->ecrireInterfaces(interfaces);
    listeChoixInterface->addItem(interfaces.back().portInterface);
    listeInterfacesSupprimables->addItem(interfaces.back().portInterface);
}
```

8.12.3.52 void IHM : :selectionnerInterface (int *index*) [slot]

Références interfaceDMX, interfaces, EnttecDMXUSB : :IsOpen(), projecteursDMX, et resultatDetectionInterface.

Référencé par IHM(), et recupererDerniereInterfaceUtilisee().

```
{
    if(interfaces.count() == 0)
    {
        if(interfaceDMX != NULL)
            delete interfaceDMX;
        interfaceDMX = NULL;
        return;
    }

    if(interfaceDMX != NULL)
        delete interfaceDMX;

    for(int i = 0; i < interfaces.count(); i++)
    {
        interfaces[i].utilisee = 0;
    }

    interfaceDMX = new EnttecDMXUSB(interfaces.at(index).typeInterface,
      interfaces.at(index).portInterface.toLocal8Bit().data());
    interfaces[index].utilisee = 1;

    if(interfaceDMX->IsOpen())
    {
        for(int i = 0; i < projecteursDMX.count(); i++)
        {
            projecteursDMX.at(i)->setEnttecDMXUSB(interfaceDMX);
        }
    }
    else
    {
        QMessageBox::critical(0, QString::fromUtf8("DMX 2018"),
      QString::fromUtf8("Impossible d'ouvrir l'interface DMX !"));
        delete interfaceDMX;
        interfaceDMX = NULL;
    }
    resultatDetectionInterface->setText(QString::fromUtf8(""));
}
```

8.12.3.53 void IHM : :selectionnerProjecteursPilotage (int *index*) [slot]

Paramètres

index	

Renvoie

void

Références boutonDecrementationCanal, boutonIncrementationCanal, canalDepartFaders, envoyerCanalDepartWing(), DMXProjecteur : :getCanalDebut(), DMXProjecteur : :getNomCanal(), DMXProjecteur : :getNomCanaux(), NB_SLIDERS, pilotageDMXActif, projecteursDMX, MySlider : :setCanal(), setCanal(), MySlider : :setCanalEnabled(), MySlider : :setConnecte(), MySlider : :setNomCanal(), et slidersCanaux.

Référencé par IHM().

```
{
    if(index == 0)
    {
        //qDebug() << Q_FUNC_INFO << index << "Aucun";
        pilotageDMXActif = false;
        boutonIncrementationCanal->setEnabled(true);
        boutonDecrementationCanal->setEnabled(true);
        canalDepartFaders = 1;
        setCanal();
        pilotageDMXActif = true;
        return;
    }

    DMXProjecteur* projecteurDMX = projecteursDMX.at(index-1);
    //qDebug() << Q_FUNC_INFO << index << projecteurDMX->getNom() <<
       projecteurDMX->getType();
    pilotageDMXActif = false;
    for(int i = 0; i < NB_SLIDERS; i ++)
    {
        if(i >= projecteurDMX->getNomCanaux().size())
        {
            slidersCanaux[i]->hide();
        }
        else
        {
            slidersCanaux[i]->setConnecte(false);
            slidersCanaux[i]->setCanalEnabled(false);
            slidersCanaux[i]->setCanal(projecteurDMX->getCanalDebut() + i);
            slidersCanaux[i]->setNomCanal(projecteurDMX->getNomCanal(
      projecteurDMX->getCanalDebut() + i));
            slidersCanaux[i]->show();
            slidersCanaux[i]->setConnecte(true);
        }
    }
    canalDepartFaders = projecteurDMX->getCanalDebut();
    boutonIncrementationCanal->setEnabled(false);
    boutonDecrementationCanal->setEnabled(false);
    pilotageDMXActif = true;
    envoyerCanalDepartWing();
}
```

8.12.3.54 void IHM : :setCanal ( ) [private]

Renvoie

void

Références canalDepartFaders, listeProjecteursPilotage, NB_CANAUX, NB_SLIDERS, MySlider : :setCanal(), MySlider : :setCanalEnabled(), MySlider : :setConnecte(), MySlider : :setNomCanal(), et slidersCanaux.

Référencé par afficherEtatFaders(), afficherEtatTouchesControle(), changerCanal-

Depart(), creerIHM(), decrementerCanal(), incrementerCanal(), et selectionnerProjecteursPilotage().

```
{
    if(listeProjecteursPilotage->currentIndex() == 0)
    {
        for(int i = 0; i < NB_SLIDERS; i ++)
        {
            slidersCanaux[i]->setConnecte(false);
            if(i == 0)
                slidersCanaux[i]->setCanalEnabled(true);
            slidersCanaux[i]->setNomCanal("");
            slidersCanaux[i]->setCanal(canalDepartFaders + i);
            if(canalDepartFaders + i > NB_CANAUX)
                slidersCanaux[i]->hide();
            else
                slidersCanaux[i]->show();
            slidersCanaux[i]->setConnecte(true);
        }
    }
}
```

8.12.3.55 void IHM : :supprimerConsole ( ) [slot]

Renvoie

void

Références XMLUtilitaire : :ecrireConsole(), PlaybackWing : :getAdresseIPConsole(), PlaybackWing : :getDebutListeConsoles(), PlaybackWing : :getListeConsoles(), PlaybackWing : :getNombreConsoles(), listeChoixConsole, PlaybackWing : :supprimerConsoleListe(), utilitaireDocuments, et wing.

Référencé par IHM().

```
{
    QList<Console>::iterator it;
    it = wing->getDebutListeConsoles();
    for(int i = 0; i < wing->getNombreConsoles(); i++)
    {
        if(wing->getAdresseIPConsole(i) == listeChoixConsole->currentText())
        {
            wing->supprimerConsoleListe(it);
            listeChoixConsole->removeItem(i);
        }
        it++;
    }
    utilitaireDocuments->ecrireConsole(wing->getListeConsoles());
}
```

8.12.3.56 void IHM : :supprimerInterface ( ) [slot]

Renvoie

void

Références XMLUtilitaire : :ecrireInterfaces(), interfaces, listeChoixInterface, listeInterfacesSupprimables, et utilitaireDocuments.

Référencé par IHM().

```
{
```

```
    QList<Interface>::iterator it;
    it = interfaces.begin();
    for(int i = 0; i < interfaces.count(); i++)
    {
        if(interfaces[i].portInterface == listeInterfacesSupprimables->
      currentText())
        {
            interfaces.erase(it);
            listeInterfacesSupprimables->removeItem(i);
            listeChoixInterface->removeItem(i);
        }
        it++;
    }

    utilitaireDocuments->ecrireInterfaces(interfaces);

}
```

8.12.3.57 void IHM : :supprimerProjecteur (QString *uuid*) [slot]

Paramètres

uuid	

Renvoie

void

Références XMLUtilitaire : :afficherProjecteursEnregistres(), effacerAffichage-ProjecteursEnregistres(), effacerAffichageProjecteursPilotage(), XMLUtilitaire : :lire-Appareils(), mettreAJourProjecteursEnregistres(), mettreAJourProjecteursPilotage(), projecteursDMX, projecteursEnregistres, XMLUtilitaire : :supprimerAppareil(), et utilitaireDocuments.

Référencé par mettreAJourProjecteursEnregistres().

```
{
    qDebug() << Q_FUNC_INFO;

    effacerAffichageProjecteursEnregistres();
    effacerAffichageProjecteursPilotage();
    utilitaireDocuments->supprimerAppareil(uuid);

    utilitaireDocuments->afficherProjecteursEnregistres(projecteursEnregistres)
      ;


    utilitaireDocuments->lireAppareils(projecteursDMX);
    mettreAJourProjecteursPilotage();
    mettreAJourProjecteursEnregistres();

}
```

8.12.3.58 void IHM : :supprimerScene ( ) [slot]

Renvoie

void

Références choixScenesDisponibles, XMLUtilitaire : :lireScenes(), scenes, XML-Utilitaire : :supprimerScene(), et utilitaireDocuments.

Référencé par IHM().

```
{
    utilitaireDocuments->supprimerScene(scenes[choixScenesDisponibles->
      currentIndex()].uuid);
    utilitaireDocuments->lireScenes(scenes);
    choixScenesDisponibles->removeItem(choixScenesDisponibles->currentIndex());
    qDebug() << Q_FUNC_INFO;
}
```

8.12.3.59 void IHM : :supprimerSequence ( ) [slot]

Références choixSequencesDisponibles, XMLUtilitaire : :lireSequences(), sequences, XMLUtilitaire : :supprimerSequence(), et utilitaireDocuments.

Référencé par IHM().

```
{
    utilitaireDocuments->supprimerSequence(sequences[choixSequencesDisponibles
      ->currentIndex()].uuid);
    utilitaireDocuments->lireSequences(sequences);
    choixSequencesDisponibles->removeItem(choixSequencesDisponibles->
      currentIndex());
    qDebug() << Q_FUNC_INFO;
}
```

8.12.3.60 void IHM : :supprimerSpectacle ( ) [slot]

Références choixSpectaclesDisponibles, XMLUtilitaire : :ecrireSpectacles(), spectacles, et utilitaireDocuments.

Référencé par IHM().

```
{

    spectacles.remove(choixSpectaclesDisponibles->currentIndex());
    choixSpectaclesDisponibles->removeItem(choixSpectaclesDisponibles->
      currentIndex());
    utilitaireDocuments->ecrireSpectacles(spectacles);
}
```

8.12.3.61 void IHM : :supprimerTousProjecteurs ( ) [slot]

Renvoie

void

Références effacerAffichageProjecteursEnregistres(), effacerAffichageProjecteurs-Pilotage(), XMLUtilitaire : :lireAppareils(), projecteursDMX, XMLUtilitaire : :supprimer-ProjecteursEnregistres(), et utilitaireDocuments.

Référencé par IHM().

```
{
    qDebug() << Q_FUNC_INFO;
    effacerAffichageProjecteursEnregistres();
    effacerAffichageProjecteursPilotage();
    utilitaireDocuments->supprimerProjecteursEnregistres();
    utilitaireDocuments->lireAppareils(projecteursDMX);

}
```

8.12.3.62 void IHM : :verifierAjoutZero (QString & *nombre*)

Renvoie

void

Référencé par ajouterSceneSequence().

```
{
    if(nombre.toInt() < 10)
    {
        nombre = "0" + nombre;
    }
}
```

8.12.4 Documentation des données membres

8.12.4.1 QAction* IHM : :actionQuitter [private]

Référencé par creerIHM(), et IHM().

8.12.4.2 QPushButton* IHM : :boutonAjouterConsole [private]

Référencé par creerIHM(), et IHM().

8.12.4.3 QPushButton* IHM : :boutonAjouterInterface [private]

Référencé par creerIHM(), et IHM().

8.12.4.4 QPushButton* IHM : :boutonAjouterScene [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.5 QPushButton* IHM : :boutonchoixSequenceSpectacle [private]

Référencé par creerIHM(), et IHM().

8.12.4.6 QPushButton* IHM : :boutonDecrementationCanal [private]

Référencé par activerPilotageIHM(), creerIHM(), desactiverPilotageIHM(), IHM(), et selectionnerProjecteursPilotage().

8.12.4.7 QPushButton* IHM : :boutonDetecterInterface [private]

Référencé par creerIHM(), et IHM().

8.12.4.8 QPushButton* IHM : :boutonEnregistrer [private]

Référencé par genererFenetreNouveauProjecteur(), et IHM().

8.12.4.9 QPushButton* IHM : :boutonEnregistrerModificationConsole [private]

Référencé par genererFenetreModifierConsole().

8.12.4.10 QPushButton* IHM : :boutonEnregistrerSequence [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.11 QPushButton* IHM : :boutonexecuterScene [private]

Référencé par creerIHM(), et IHM().

8.12.4.12 QPushButton* IHM : :boutonExecuterSequence [private]

Référencé par creerIHM(), et IHM().

8.12.4.13 QPushButton* IHM : :boutonExecuterSpectacle [private]

Référencé par creerIHM(), et IHM().

8.12.4.14 QPushButton* IHM : :boutonIncrementationCanal [private]

Référencé par activerPilotageIHM(), creerIHM(), desactiverPilotageIHM(), IHM(), et selectionnerProjecteursPilotage().

8.12.4.15 QPushButton* IHM : :boutonModifierConsole [private]

Référencé par creerIHM(), et IHM().

8.12.4.16 QPushButton* IHM : :boutonModifierScene [private]

Référencé par creerIHM().

8.12.4.17 QPushButton* IHM : :boutonNouveauProjecteur [private]

Référencé par creerIHM(), et IHM().

8.12.4.18 QPushButton* IHM : :boutonNouvelleScene [private]

Référencé par creerIHM(), et IHM().

8.12.4.19 QPushButton* IHM : :boutonNouvelleSequence [private]

Référencé par creerIHM(), et IHM().

8.12.4.20 QPushButton* IHM : :boutonParametresCanauxScene [private]

Référencé par genererFenetreNouvelleScene(), et IHM().

8.12.4.21 QPushButton* IHM : :boutonSauvegarderCanauxScene [private]

Référencé par genererFenetreCanauxScene().

8.12.4.22 QPushButton* IHM : :boutonSauvegarderScene [private]

Référencé par genererFenetreNouvelleScene().

8.12.4.23 QPushButton* IHM : :boutonSauvegarderSpectacle [private]

Référencé par creerIHM(), et IHM().

8.12.4.24 QPushButton* IHM : :boutonSupprimerConsole [private]

Référencé par creerIHM(), et IHM().

8.12.4.25 QPushButton* IHM : :boutonSupprimerInterface [private]

Référencé par creerIHM(), et IHM().

8.12.4.26 QPushButton* IHM : :boutonSupprimerScene [private]

Référencé par creerIHM(), et IHM().

8.12.4.27 QPushButton* IHM : :boutonSupprimerSpectacle [private]

Référencé par creerIHM(), et IHM().

8.12.4.28 QPushButton* IHM : :boutonSupprimerTousProjecteurs [private]

Référencé par creerIHM(), et IHM().

8.12.4.29 QPushButton* IHM : :boutonSupprimerToutesScenes [private]

Référencé par creerIHM().

8.12.4.30 QPushButton* IHM : :boutonSupprimmerSequence [private]

Référencé par creerIHM(), et IHM().

8.12.4.31 Scene IHM : :cacheScene [private]

Référencé par ajouterAppareilScene(), et enregistrerScene().

8.12.4.32 Sequence IHM : :cacheSequence [private]

Référencé par ajouterSceneSequence(), et enregistrerSequence().

8.12.4.33 Spectacle IHM : :cacheSpectacle [private]

Référencé par ajouterSequenceSpectacle(), et enregistrerSpectacle().

8.12.4.34 QString IHM : :cacheUuid [private]

Référencé par enregistrerProjecteur(), et ouvrirFenetreModifierProjecteur().

8.12.4.35 int IHM : :canalDepartFaders [private]

Référencé par changerCanalDepart(), creerIHM(), decrementerCanal(), IHM(), incrementerCanal(), selectionnerProjecteursPilotage(), et setCanal().

8.12.4.36 QLineEdit* IHM : :champAdresseIPConsole [private]

Référencé par ajouterConsole(), et creerIHM().

8.12.4.37 QLineEdit* IHM : :champAdresseIPModifierConsole [private]

Référencé par envoyerModificationConsole(), et genererFenetreModifierConsole().

8.12.4.38 QLineEdit* IHM : :champPortConsole [private]

Référencé par ajouterConsole(), et creerIHM().

8.12.4.39 QLineEdit* IHM : :champPortModifierConsole [private]

Référencé par envoyerModificationConsole(), et genererFenetreModifierConsole().

8.12.4.40 QComboBox* IHM : :choixAppareilScene [private]

Référencé par ajouterAppareilScene(), genererFenetreCanauxScene(), genererFenetreNouvelleScene(), ouvrirFenetreCanauxScene(), et ouvrirFenetreNouvelleScene().

8.12.4.41 QSpinBox* IHM : :choixCanalDepart [private]

Référencé par enregistrerProjecteur(), genererFenetreNouveauProjecteur(), ouvrirFenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

8.12.4.42 QSpinBox* IHM : :choixHeures [private]

Référencé par ajouterSceneSequence(), genererFenetreNouvelleSequence(), et ouvrirFenetreNouvelleSequence().

8.12.4.43 QSpinBox* IHM : :choixMinutes [private]

Référencé par ajouterSceneSequence(), genererFenetreNouvelleSequence(), et ouvrirFenetreNouvelleSequence().

8.12.4.44 QSpinBox* IHM : :choixNBCanaux [private]

Référencé par enregistrerProjecteur(), genererFenetreNouveauProjecteur(), ouvrirFenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

8.12.4.45 QLineEdit* IHM : :choixNom [private]

Référencé par enregistrerProjecteur(), genererFenetreNouveauProjecteur(), ouvrirFenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

8.12.4.46 QLineEdit* IHM : :choixNomScene [private]

Référencé par enregistrerScene(), genererFenetreNouvelleScene(), et ouvrirFenetreNouvelleScene().

8.12.4.47 QLineEdit* IHM : :choixnomSequence [private]

Référencé par enregistrerSequence(), genererFenetreNouvelleSequence(), et ouvrirFenetreNouvelleSequence().

8.12.4.48 QLineEdit* IHM : :choixNomSpectacle [private]

Référencé par creerIHM(), et enregistrerSpectacle().

8.12.4.49 QComboBox* IHM : :choixScenesDisponibles [private]

Référencé par creerIHM(), enregistrerScene(), executerSceneDefault(), et supprimerScene().

8.12.4.50 QComboBox* IHM : :choixScenesSequence [private]

Référencé par ajouterSceneSequence(), genererFenetreNouvelleSequence(), et ouvrirFenetreNouvelleSequence().

8.12.4.51 QSpinBox* IHM : :choixSecondes [private]

Référencé par ajouterSceneSequence(), genererFenetreNouvelleSequence(), et ouvrirFenetreNouvelleSequence().

8.12.4.52 QComboBox* IHM : :choixSequencesDisponibles [private]

Référencé par creerIHM(), enregistrerSequence(), executerSequenceDefault(), et supprimerSequence().

8.12.4.53 QComboBox* IHM : :choixSequencesSpectacle [private]

Référencé par ajouterSequenceSpectacle(), et creerIHM().

8.12.4.54 QComboBox* IHM : :choixSpectaclesDisponibles [private]

Référencé par creerIHM(), enregistrerSpectacle(), executerSpectacle(), et supprimerSpectacle().

8.12.4.55 QComboBox* IHM : :choixTypes [private]

Référencé par enregistrerProjecteur(), genererFenetreNouveauProjecteur(), ouvrirFenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

8.12.4.56 QSpacerItem* IHM : :espace [private]

Référencé par creerIHM().

8.12.4.57 QDialog* IHM : :fenetreCanauxScene [private]

Référencé par ajouterAppareilScene(), genererFenetreCanauxScene(), et ouvrirFenetreCanauxScene().

8.12.4.58 QDialog* IHM : :fenetreInformationsAppareil [private]

Référencé par genererFenetreInformationsAppareil(), et ouvrirFenetreInformations-Projecteur().

8.12.4.59 QDialog* IHM : :fenetreModifierConsole [private]

Référencé par envoyerModificationConsole(), et genererFenetreModifierConsole().

8.12.4.60 QDialog* IHM : :fenetreNouveauProjecteur [private]

Référencé par enregistrerProjecteur(), genererFenetreNouveauProjecteur(), ouvrir-FenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

8.12.4.61 QDialog* IHM : :fenetreNouvelleScene [private]

Référencé par enregistrerScene(), genererFenetreNouvelleScene(), et ouvrirFenetre-NouvelleScene().

8.12.4.62 QDialog* IHM : :fenetreNouvelleSequence [private]

Référencé par enregistrerSequence(), genererFenetreNouvelleSequence(), et ouvrir-FenetreNouvelleSequence().

8.12.4.63 int IHM : :incrementTemporisation [private]

8.12.4.64 EnttecDMXUSB* IHM : :interfaceDMX [private]

Référencé par detecterInterface(), envoyerTrameDMX(), executerScene(), IHM(), selectionnerInterface(), et ~IHM().

8.12.4.65 QList<Interface> IHM : :interfaces [private]

Référencé par creerIHM(), detecterInterface(), IHM(), recupererDerniereInterface-Utilisee(), recupererDonneesNouvelleInterface(), selectionnerInterface(), supprimer-Interface(), et ~IHM().

8.12.4.66 QLabel* IHM : :labelAppareilScene [private]

Référencé par genererFenetreNouvelleScene().

8.12.4.67 QLabel* IHM : :labelCanalDepart [private]

Référencé par genererFenetreNouveauProjecteur().

8.12.4.68 QLabel* IHM : :labelCreationSpectacle [private]

8.12.4.69 QLabel* IHM : :labelHeures [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.70 QLabel* IHM : :labelInformationsCanalDepart [private]

Référencé par genererFenetreInformationsAppareil(), et ouvrirFenetreInformations-Projecteur().

8.12.4.71 QLabel* IHM : :labelInformationsCanaux[10] [private]

Référencé par genererFenetreInformationsAppareil(), et ouvrirFenetreInformations-Projecteur().

8.12.4.72 QLabel* IHM : :labelInterface [private]

Référencé par creerIHM().

8.12.4.73 QLabel* IHM : :labelJeuSpectacle [private]

8.12.4.74 QLabel* IHM : :labelMinutes [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.75 QLabel* IHM : :labelNBCanaux [private]

Référencé par genererFenetreNouveauProjecteur().

8.12.4.76 QLabel* IHM : :labelNom [private]

Référencé par genererFenetreNouveauProjecteur().

8.12.4.77 QLabel* IHM : :labelNomCanaux [private]

Référencé par genererFenetreNouveauProjecteur().

8.12.4.78 QLabel* IHM : :labelNomScene [private]

Référencé par genererFenetreNouvelleScene().

8.12.4.79 QLabel* IHM : :labelnomSequence [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.80 QLabel* IHM : :labelnomSpectacle [private]

Référencé par creerIHM().

8.12.4.81 QList<QLabel*> IHM : :labelsAppareilsScene [private]

Référencé par ajouterAppareilScene(), ouvrirFenetreCanauxScene(), et ouvrirFenetre-NouvelleScene().

8.12.4.82 QLabel* IHM : :labelScenesDisponibles [private]

Référencé par creerIHM().

8.12.4.83 QLabel* IHM : :labelScenesSequence [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.84 QLabel* IHM : :labelSecondes [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.85 QLabel* IHM : :labelSequencesDisponibles [private]

Référencé par creerIHM().

8.12.4.86 QLabel* IHM : :labelSequencesSpectacle [private]

Référencé par creerIHM().

8.12.4.87 QLabel* IHM : :labelSpectaclesDisponibles [private]

Référencé par creerIHM().

8.12.4.88 QList<QLabel*> IHM : :labelsScenesSequences [private]

Référencé par ajouterSceneSequence(), et ouvrirFenetreNouvelleSequence().

8.12.4.89 QList<QLabel*> IHM : :labelsSequencesSpectacle [private]

Référencé par ajouterSequenceSpectacle(), et enregistrerSpectacle().

8.12.4.90 QLabel* IHM : :labelTemporisation [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.91 QLabel* IHM : :labelType [private]

Référencé par genererFenetreNouveauProjecteur().

8.12.4.92 QVBoxLayout* IHM : :layoutlisteAppareilsScene [private]

Référencé par ajouterAppareilScene(), et genererFenetreNouvelleScene().

8.12.4.93 QVBoxLayout* IHM : :layoutListeProjecteursParametres [private]

Référencé par creerIHM(), et mettreAJourProjecteursEnregistres().

8.12.4.94 QVBoxLayout* IHM : :layoutListeScenesSequences [private]

Référencé par ajouterSceneSequence(), et genererFenetreNouvelleSequence().

8.12.4.95 QVBoxLayout* IHM : :layoutListeSequenceSpectacle [private]

Référencé par ajouterSequenceSpectacle(), et creerIHM().

8.12.4.96 QScrollArea* IHM : :listeAppareilsScene [private]

Référencé par genererFenetreNouvelleScene().

8.12.4.97 QComboBox* IHM : :listeChoixConsole [private]

Référencé par ajouterConsole(), creerIHM(), envoyerModificationConsole(), et supprimerConsole().

8.12.4.98 QComboBox* IHM : :listeChoixInterface [private]

Référencé par creerIHM(), detecterInterface(), IHM(), recupererDerniereInterfaceUtilisee(), recupererDonneesNouvelleInterface(), et supprimerInterface().

8.12.4.99 QComboBox* IHM : :listeChoixPilotage [private]

Référencé par afficherEtatTouchesControle(), creerIHM(), et IHM().

8.12.4.100 QScrollArea* IHM : :listeInformationsCanaux [private]

Référencé par genererFenetreInformationsAppareil().

8.12.4.101 QComboBox* IHM : :listeInterfacesSupprimables [private]

Référencé par creerIHM(), recupererDonneesNouvelleInterface(), et supprimerInterface().

8.12.4.102 QScrollArea* IHM : :listeProjecteursEnregistres [private]

Référencé par creerIHM().

8.12.4.103 QComboBox* IHM : :listeProjecteursPilotage [private]

Référencé par activerPilotageIHM(), afficherEtatTouchesControle(), creerIHM(), desactiverPilotageIHM(), effacerAffichageProjecteursPilotage(), IHM(), mettreAJourProjecteursPilotage(), et setCanal().

8.12.4.104 QScrollArea* IHM : :listeScenesSequenceChoisies [private]

Référencé par genererFenetreNouvelleSequence().

8.12.4.105 QScrollArea* IHM : :listeSequencesSpectacle [private]

Référencé par creerIHM().

8.12.4.106 QComboBox* IHM : :listeTypeInterface [private]

Référencé par creerIHM(), detecterInterface(), et recupererDerniereInterfaceUtilisee().

8.12.4.107 QComboBox* IHM : :listeTypeInterfaceAjouterInterface [private]

Référencé par creerIHM(), et recupererDonneesNouvelleInterface().

8.12.4.108 int IHM : :nbAppareils [private]

8.12.4.109 QLineEdit* IHM : :nomsCanaux[10] [private]

Référencé par bloquerChoixNomCanaux(), enregistrerProjecteur(), genererFenetreNouveauProjecteur(), ouvrirFenetreModifierProjecteur(), et ouvrirFenetreNouveauProjecteur().

8.12.4.110 QLineEdit* IHM : :numeroIdAjouterInterface [private]

8.12.4.111 QTabWidget* IHM : :ongletsPrincipaux [private]

Référencé par creerIHM().

8.12.4.112 QTabWidget* IHM : :optionsCreations [private]

Référencé par creerIHM().

8.12.4.113 QTabWidget* IHM : :optionsParametres [private]

Référencé par creerIHM().

8.12.4.114 QTabWidget* IHM : :optionsSpectacles [private]

Référencé par creerIHM().

8.12.4.115 QWidget* IHM : :pageAssembler [private]

Référencé par creerIHM().

8.12.4.116 QWidget* IHM : :pageCreer [private]

Référencé par creerIHM().

8.12.4.117 QWidget* IHM : :pageJouer [private]

Référencé par creerIHM().

8.12.4.118 QWidget* IHM : :pagePiloter [private]

Référencé par creerIHM().

8.12.4.119 QWidget* IHM : :pagesSpectacles[2] [private]

Référencé par creerIHM().

8.12.4.120 QWidget* IHM : :parametresConsoles [private]

Référencé par creerIHM().

8.12.4.121 QWidget* IHM : :parametresInterface [private]

Référencé par creerIHM().

8.12.4.122 QWidget* IHM : :parametresProjecteurs [private]

Référencé par creerIHM().

8.12.4.123 bool IHM : :pilotageDMXActif [private]

Référencé par activerPilotageIHM(), changerCanalDepart(), decrementerCanal(), desactiverPilotageIHM(), envoyerTrameDMX(), genererFenetreCanauxScene(), IHM(), incrementerCanal(), et selectionnerProjecteursPilotage().

8.12.4.124 bool IHM : :pilotageIHMActif [private]

Référencé par activerPilotageIHM(), afficherEtatTouchesControle(), desactiverPilotageIHM(), et IHM().

8.12.4.125 QLineEdit* IHM : :portInterfaceAjouterInterface [private]

Référencé par creerIHM(), et recupererDonneesNouvelleInterface().

8.12.4.126 QVector<DMXProjecteur*> IHM : :projecteursDMX [private]

Référencé par creerIHM(), effacerAffichageProjecteursPilotage(), enregistrerProjecteur(), genererFenetreCanauxScene(), genererFenetreNouvelleScene(), IHM(), mettreAJourProjecteursPilotage(), ouvrirFenetreInformationsProjecteur(), ouvrirFenetreModifierProjecteur(), ouvrirFenetreNouvelleScene(), selectionnerInterface(), selectionnerProjecteursPilotage(), supprimerProjecteur(), supprimerTousProjecteurs(), et ~IHM().

8.12.4.127 IDProjecteurSauvegarde* IHM : :projecteursEnregistres[NB_PROJECTEURS] [private]

Référencé par creerIHM(), effacerAffichageProjecteursEnregistres(), enregistrerProjecteur(), mettreAJourProjecteursEnregistres(), et supprimerProjecteur().

8.12.4.128 QTextEdit* IHM : :resultatDetectionInterface [private]

Référencé par creerIHM(), detecterInterface(), et selectionnerInterface().

8.12.4.129 QVector<Scene> IHM : :scenes [private]

Référencé par ajouterSceneSequence(), creerIHM(), enregistrerScene(), executerSceneDefault(), genererFenetreNouvelleSequence(), ouvrirFenetreNouvelleSequence(), et supprimerScene().

8.12.4.130 QVector<Sequence> IHM : :sequences [private]

Référencé par ajouterSequenceSpectacle(), creerIHM(), enregistrerSequence(), executerSequenceDefault(), et supprimerSequence().

8.12.4.131 MySlider* IHM : :slidersCanaux[NB_SLIDERS] [private]

Référencé par activerPilotageIHM(), afficherEtatFaders(), creerIHM(), desactiverPilotageIHM(), envoyerCanalDepartWing(), IHM(), selectionnerProjecteursPilotage(), et setCanal().

8.12.4.132 MySlider* IHM : :slidersCanauxScene[NB_SLIDERS] [private]

Référencé par ajouterAppareilScene(), et genererFenetreCanauxScene().

8.12.4.133 QVector<Spectacle> IHM : :spectacles [private]

Référencé par creerIHM(), enregistrerSpectacle(), executerSpectacle(), et supprimerSpectacle().

8.12.4.134 XMLUtilitaire* IHM : :utilitaireDocuments [private]

Référencé par creerIHM(), effacerAffichageProjecteursEnregistres(), enregistrerProjecteur(), enregistrerScene(), enregistrerSequence(), enregistrerSpectacle(), IHM(), mettreAJourProjecteursEnregistres(), recupererDonneesNouvelleInterface(), supprimerConsole(), supprimerInterface(), supprimerProjecteur(), supprimerScene(), supprimerSequence(), supprimerSpectacle(), supprimerTousProjecteurs(), et ~IHM().

8.12.4.135 PlaybackWing* IHM : :wing [private]

Référencé par activerPilotageConsole(), ajouterConsole(), creerIHM(), demarrer(), desactiverPilotageConsole(), envoyerCanalDepartWing(), envoyerModificationConsole(), IHM(), supprimerConsole(), et ~IHM().

La documentation de cette classe a été générée à partir des fichiers suivants :
- IHM.h
- IHM.cpp

8.13 Référence de la structure Interface

Structure comprenant les différents paramètre d'une interface USB/DMX.

`#include <interface.h>`

Attributs publics

- int id
- QString portInterface
- EnttecInterfaces typeInterface
- bool utilisee

8.13.1 Description détaillée

Version

1.0

8.13.2 Documentation des données membres

8.13.2.1 int Interface : :id

Référencé par XMLUtilitaire : :lireInterfaces(), et IHM : :recupererDonneesNouvelleInterface().

8.13.2.2 QString Interface : :portInterface

Référencé par XMLUtilitaire : :lireInterfaces(), et IHM : :recupererDonneesNouvelleInterface().

8.13.2.3 EnttecInterfaces Interface : :typeInterface

Référencé par IHM : :detecterInterface(), XMLUtilitaire : :lireInterfaces(), et IHM : :recupererDonneesNouvelleInterface().

8.13.2.4 bool Interface : :utilisee

Référencé par XMLUtilitaire : :lireInterfaces(), et IHM : :recupererDonneesNouvelleInterface().

La documentation de cette structure a été générée à partir du fichier suivant :
– interface.h

8.14 Référence de la classe MySlider

Déclaration de la classe myslider permettant d'instancier des widgets de type slider personnalisés.

`#include <myslider.h>`

Connecteurs publics

– void setValeur (int valeur)
  Mutateur de la valeur du SpinBox s.
– void setCanal (int canal)
  Mutateur de la valeur du ScrollBar sb.
– void setNomCanal (QString nomCanal)
  Mutateur de la valeur du Label nom.
– void updateDMX (int canal, int valeur)
  Non utilisé
– void setConnecte (bool etat)
  Connexion/Déconnexion des slots et signaux de la classe MySlider.

Signaux

– void sliderChange (int canal, int valeur)

– void valeurChange (int valeur)
– void canalChange (int canal)

Fonctions membres publiques

– MySlider (QWidget ∗parent=0, int canal=0, int valeur=0, QString nomCanal=QString(""))
– ∼MySlider ()
– void setCanalEnabled (bool etat)
  active le ScrollBar sb
– int getCanal ()
  Assesseur de la valeur du SpinBox s.
– int getValeur ()
  Assesseur de la valeur du SpinBox s.

Connecteurs privés

– void changerValeur (int valeur)
  Emmeteur des signaux valeurChange et sliderChange.
– void changerCanal (int canal)
  Emmeteur des signaux canalChange et sliderChange.

Attributs privés

– QLCDNumber ∗ lcd
– QSlider ∗ s
– QSpinBox ∗ sb
– QLabel ∗ nom
– bool etat

8.14.1 Description détaillée

Version

1.0

8.14.2 Documentation des constructeurs et destructeur

8.14.2.1 MySlider : :MySlider (QWidget ∗ *parent* = 0, int *canal* = 0, int *valeur* = 0, QString *nomCanal* = QString(""))

Références canalChange(), changerCanal(), changerValeur(), etat, lcd, NB_CANAUX, NB_DIGIT, nom, s, sb, VALEUR_MAX, VALEUR_MIN, et valeurChange().

:

```
    QWidget( parent )
{
  lcd = new QLCDNumber(NB_DIGIT, this );
  s = new QSlider( Qt::Vertical, this );
  s->setTickPosition( QSlider::TicksBothSides );
  s->setTickInterval(VALEUR_MAX);
  s->setMinimum(VALEUR_MIN);
  s->setMaximum(VALEUR_MAX);
```

```
    sb = new QSpinBox(this);
    sb->setMaximum(NB_CANAUX);

    nom = new QLabel(nomCanal, this);

    QVBoxLayout *mainLayout = new QVBoxLayout;
    QHBoxLayout *hLayout1 = new QHBoxLayout;
    QHBoxLayout *hLayout2 = new QHBoxLayout;
    QHBoxLayout *hLayout3 = new QHBoxLayout;
    QHBoxLayout *hLayout4 = new QHBoxLayout;
    hLayout1->addWidget(lcd);
    hLayout2->addWidget(s);
    hLayout3->addWidget(sb);
    hLayout4->addWidget(nom);
    hLayout4->setAlignment(Qt::AlignHCenter);
    mainLayout->addLayout(hLayout1);
    mainLayout->addLayout(hLayout2);
    mainLayout->addLayout(hLayout3);
    mainLayout->addLayout(hLayout4);
    setLayout(mainLayout);

    connect( s, SIGNAL(valueChanged(int)), lcd, SLOT(display(int)) );
    connect( s, SIGNAL(valueChanged(int)), this, SLOT(changerValeur(int)) );
    connect( sb, SIGNAL(valueChanged(int)), this, SLOT(changerCanal(int)) );
    connect( this, SIGNAL(valeurChange(int)), lcd, SLOT(display(int)) );
    connect( this, SIGNAL(canalChange(int)), sb, SLOT(setValue(int)) );
    etat = true;

    lcd->setPalette(Qt::black);
    s->setValue(valeur);
    sb->setValue(canal);
}
```

8.14.2.2 MySlider : :~MySlider ( )

```
{
    //qDebug() << Q_FUNC_INFO;
}
```

8.14.3 Documentation des fonctions membres

8.14.3.1 void MySlider : :canalChange (int *canal*) [signal]

Référencé par changerCanal(), MySlider(), et setConnecte().

8.14.3.2 void MySlider : :changerCanal (int *canal*) [private, slot]

Paramètres

canal	int le numéro du canal

Renvoie

void

Références canalChange(), s, et sliderChange().

Référencé par MySlider(), et setConnecte().

```
{
    //qDebug() << "canal = " << canal;
    emit canalChange(canal);
    emit sliderChange(canal, s->value());
}
```

8.14.3.3 void MySlider : :changerValeur (int *valeur*) [private, slot]

Paramètres

valeur	int la valeur dans le canal

Renvoie

void

Références sb, sliderChange(), et valeurChange().

Référencé par MySlider(), et setConnecte().

```
{
    //qDebug() << "valeur = " << valeur;
    emit valeurChange(valeur);
    emit sliderChange(sb->value(), valeur);
}
```

8.14.3.4 int MySlider : :getCanal ( )

Renvoie

int le numéro du canal

Références sb.

Référencé par IHM : :ajouterAppareilScene().

```
{
    return sb->value();
}
```

8.14.3.5 int MySlider : :getValeur ( )

Renvoie

int la valeur dans le canal

Références s.

Référencé par IHM : :ajouterAppareilScene().

```
{
    return s->value();
}
```

8.14.3.6 void MySlider : :setCanal (int *canal*) [slot]

Paramètres

canal	int le numéro du canal

Renvoie

void

Références sb.

Référencé par IHM : :creerIHM(), IHM : :genererFenetreCanauxScene(), IHM-: :selectionnerProjecteursPilotage(), et IHM : :setCanal().

```
{
    sb->setValue(canal);
}
```

8.14.3.7 void MySlider : :setCanalEnabled (bool *etat*)

Paramètres

etat	bool true active le scrollbar sb et false le désactive

Renvoie

void

Références sb.

Référencé par IHM : :creerIHM(), IHM : :genererFenetreCanauxScene(), IHM-: :selectionnerProjecteursPilotage(), et IHM : :setCanal().

```
{
    sb->setEnabled(etat);
}
```

8.14.3.8 void MySlider : :setConnecte (bool *etat*) [slot]

Paramètres

etat	bool true permet de connecter les signaux/slos et false les déconnecte

Renvoie

void

Références canalChange(), changerCanal(), changerValeur(), etat, lcd, s, sb, et valeur-Change().

Référencé par IHM : :genererFenetreCanauxScene(), IHM : :selectionnerProjecteurs-Pilotage(), et IHM : :setCanal().

```
{
    if(etat == this->etat)
        return;

    if(etat)
    {
        connect( s, SIGNAL(valueChanged(int)), lcd, SLOT(display(int)) );
        connect( s, SIGNAL(valueChanged(int)), this, SLOT(changerValeur(int)) )
      ;
        connect( sb, SIGNAL(valueChanged(int)), this, SLOT(changerCanal(int)) )
```

```
      ;
        connect( this, SIGNAL(valeurChange(int)), lcd, SLOT(display(int)) );
        connect( this, SIGNAL(canalChange(int)), sb, SLOT(setValue(int)) );
    }
    else
    {
        disconnect( s, SIGNAL(valueChanged(int)), lcd, SLOT(display(int)) );
        disconnect( s, SIGNAL(valueChanged(int)), this, SLOT(changerValeur(int)
      ) );
        disconnect( sb, SIGNAL(valueChanged(int)), this, SLOT(changerCanal(int)
      ) );
        disconnect( this, SIGNAL(valeurChange(int)), lcd, SLOT(display(int)) );
        disconnect( this, SIGNAL(canalChange(int)), sb, SLOT(setValue(int)) );
    }
    this->etat = etat;
}
```

8.14.3.9 void MySlider : :setNomCanal (QString *nomCanal*) [slot]

Paramètres

nomCanal	QString le libellé du canal

Renvoie

void

Références nom.

Référencé par IHM : :genererFenetreCanauxScene(), IHM : :selectionnerProjecteurs-Pilotage(), et IHM : :setCanal().

```
{
    nom->setText(nomCanal);
}
```

8.14.3.10 void MySlider : :setValeur (int *valeur*) [slot]

Paramètres

valeur	int la valeur dans le canal

Renvoie

void

Références s.

Référencé par IHM : :afficherEtatFaders().

```
{
    s->setValue(valeur);
}
```

8.14.3.11 void MySlider : :sliderChange (int *canal,* int *valeur*) [signal]

Référencé par changerCanal(), et changerValeur().

8.14.3.12 void MySlider : :updateDMX (int *canal,* int *valeur*) [slot]

Paramètres

canal	int le numéro du canal
valeur	int la valeur dans le canal

Renvoie

void

```
{
    Q_UNUSED(canal)
    Q_UNUSED(valeur)
}
```

8.14.3.13 void MySlider : :valeurChange (int *valeur*) [signal]

Référencé par changerValeur(), MySlider(), et setConnecte().

8.14.4 Documentation des données membres

8.14.4.1 bool MySlider : :etat [private]

Référencé par MySlider(), et setConnecte().

8.14.4.2 QLCDNumber* MySlider : :lcd [private]

Référencé par MySlider(), et setConnecte().

8.14.4.3 QLabel* MySlider : :nom [private]

Référencé par MySlider(), et setNomCanal().

8.14.4.4 QSlider* MySlider : :s [private]

Référencé par changerCanal(), getValeur(), MySlider(), setConnecte(), et setValeur().

8.14.4.5 QSpinBox* MySlider : :sb [private]

Référencé par changerValeur(), getCanal(), MySlider(), setCanal(), setCanalEnabled(), et setConnecte().

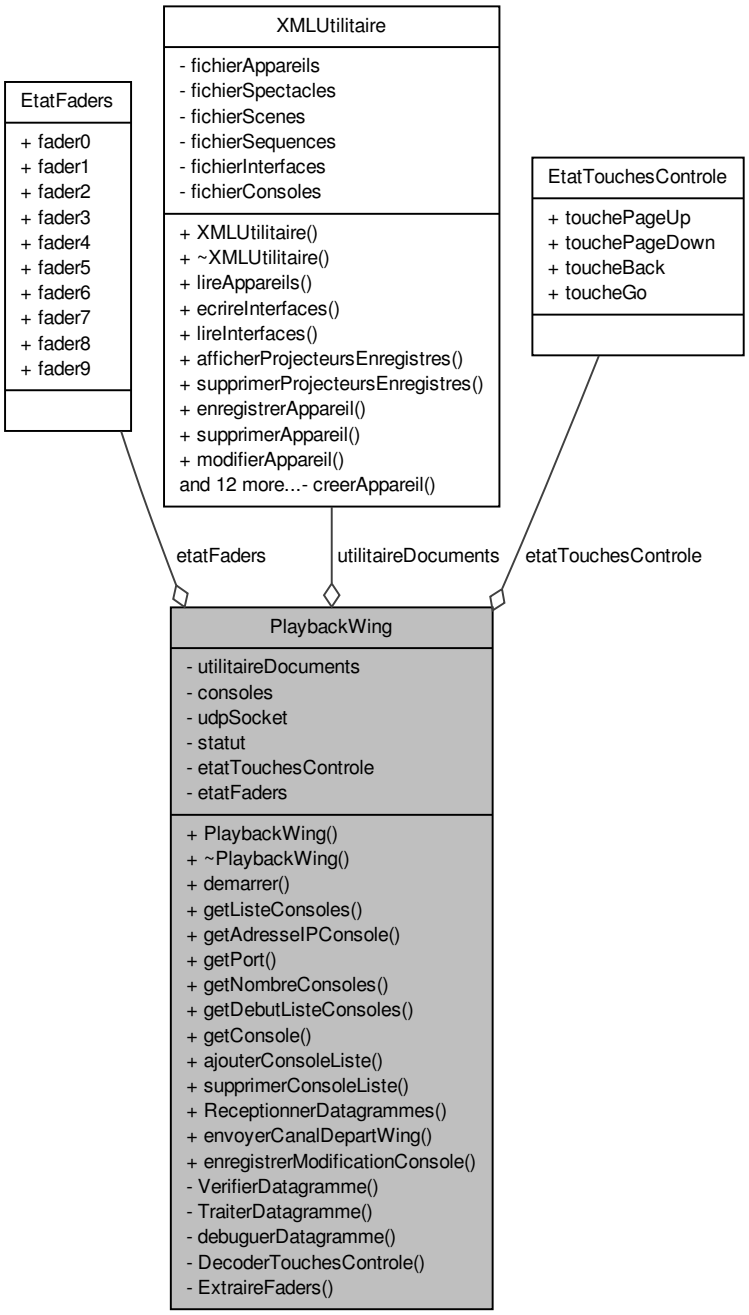
La documentation de cette classe a été générée à partir des fichiers suivants :
– myslider.h
– myslider.cpp

8.15 Référence de la classe PlaybackWing

la classe gérant la console wing

```
#include <PlaybackWing.h>
```

Graphe de collaboration de PlaybackWing :



Connecteurs publics

- void ReceptionnerDatagrammes ()
  Réceptionne les données envoyées par la console.
- void envoyerCanalDepartWing (int canalDepart)
  envoi la valeur du canal de départ actuel à la console afin Playback Wing de l'afficher sur l'écran LED
- void enregistrerModificationConsole (Console console, int numeroConsoleA-Modifier)
  Enregistre les modifications apportées a la console.

Signaux

- void terminer ()
- void envoyerEtatTouchesControle (EtatTouchesControle etatTouchesControle)
- void envoyerEtatFaders (EtatFaders etatFaders)

Fonctions membres publiques

- PlaybackWing (QObject ∗pParent=0)
  Constructeur, créé et connecte le socket udp.
- ∼PlaybackWing ()
  Destructeur.
- void demarrer ()
  Connecte le slot ReceptionnerDatagrames à la possibilité de lire la trame.
- QList< Console > getListeConsoles ()
- QString getAdresseIPConsole (int numeroConsole)
  Retourne l'adresse IP d'une console par rapport à son numéro dans la liste consoles.
- quint16 getPort (int numeroConsole)
- int getNombreConsoles ()
  Retourne le nombre de console dans la liste consoles.
- QList< Console > : :iterator getDebutListeConsoles ()
- Console getConsole (int numeroConsole)
- void ajouterConsoleListe (QString adresseIP, QString port)
- void supprimerConsoleListe (QList< Console > : :iterator it)

Fonctions membres privées

- int VerifierDatagramme (char ∗donneesBrutes, int nbOctets)
  Verifie que la trame reçue est bien une trame WODD.
- int TraiterDatagramme (const QByteArray &datagramme)
  Traite les données envoyées par la console.
- int debuguerDatagramme (char ∗donneesBrutes, int nbOctets)
  Débugue le datagramme.
- void DecoderTouchesControle (unsigned char champTouche)
  Decode les trames si une touche de contrôle est appuyée.
- void ExtraireFaders (const QByteArray &datagramme)
  Attribue les valeurs des faders depuis la trame wing.

Attributs privés

- XMLUtilitaire ∗ utilitaireDocuments
  Utilitaire de gestion des fichiers xml.
- QList< Console > consoles
- QUdpSocket ∗ udpSocket

– bool statut
– EtatTouchesControle etatTouchesControle
– EtatFaders etatFaders

8.15.1 Description détaillée

Auteur

Demont Thomas

Version

1.0

8.15.2 Documentation des constructeurs et destructeur

8.15.2.1 PlaybackWing : :PlaybackWing (QObject ∗ *pParent* = 0)

Paramètres

pParent	QObject

Références consoles, XMLUtilitaire : :lireConsoles(), PORT_ENTTEC, statut, udpSocket, et utilitaireDocuments.

```
                                          :QObject(pParent)
{
   udpSocket = new QUdpSocket(this);

   utilitaireDocuments = new XMLUtilitaire;
   utilitaireDocuments->lireConsoles(consoles);
   // Attachement locale de la socket UDP :
   statut = udpSocket->bind(QHostAddress((QString)"0.0.0.0"), PORT_ENTTEC);
   if(!statut)
   {
       QMessageBox::critical(NULL, QString::fromUtf8("PlaybackWing"),
      QString::fromUtf8("Erreur bind sur le port %1").arg(PORT_ENTTEC));
   }
   //qDebug() << Q_FUNC_INFO << statut;
}
```

8.15.2.2 PlaybackWing : :∼PlaybackWing ( )

Références udpSocket.

```
{
   udpSocket->close();
   //qDebug() << Q_FUNC_INFO;
}
```

8.15.3 Documentation des fonctions membres

8.15.3.1 void PlaybackWing : :ajouterConsoleListe (QString *adresseIP*, QString *port*)

Références Console : :adresseIP, consoles, XMLUtilitaire : :ecrireConsole(), Console : :port, et utilitaireDocuments.

Référencé par IHM : :ajouterConsole().

```
{
    Console consoleAjoutee;

    consoleAjoutee.adresseIP = adresseIP;
    consoleAjoutee.port = port.toInt();

    consoles.push_back(consoleAjoutee);

    utilitaireDocuments->ecrireConsole(consoles);
}
```

8.15.3.2 int PlaybackWing : :debuguerDatagramme (char * *donneesBrutes,* int *nbOctets*) [private]

Paramètres

donnees-Brutes	char* les octets du datagramme
nbOctets	int nombre d'octets dans la datagramme

Renvoie

int

Références INDEX_FIRMWARE, et LG_MESSAGE_WODD.

Référencé par envoyerCanalDepartWing().

```
{
  //qDebug() << Q_FUNC_INFO;
  bool commande = false;
  bool enregistrement = false;
  int etat;
  int valeur;

  //qDebug() << nbOctets;
  //#ifdef DEBUG_PlaybackWing
  if(nbOctets == LG_MESSAGE_WODD)
  {
      for(int i=0;i<INDEX_FIRMWARE;i++)
      {
          printf("%c", donneesBrutes[i]); //WODD
      }
      printf(" ");
      for(int i=INDEX_FIRMWARE;i<nbOctets;i++)
      {
          printf("0x%02X ", (unsigned char)donneesBrutes[i]); // le reste des
       données
      }
      printf("\n\n");
  }
  //#endif

  return 0;
}
```

8.15.3.3 void PlaybackWing : :DecoderTouchesControle (unsigned char *champTouche*) [private]

Paramètres

champ-Touche	le champ de la trame correspondant au touches de contrôle

Renvoie

void

Références BACK, etatTouchesControle, GO, PAGE_DOWN, PAGE_UP, EtatTouches-Controle : :toucheBack, EtatTouchesControle : :toucheGo, EtatTouchesControle-: :touchePageDown, et EtatTouchesControle : :touchePageUp.

Référencé par TraiterDatagramme().

```
{
    // Touches de contrôle (INDEX_TOUCHES_CONTROLE 6)
    /*
        Bit 7: 0=PageUp key pressed, 1=PageUp key released.
        Bit 6: 0=PageDown key pressed, 1=PageDown key released.
        Bit 5: 0=Back key pressed, 1=Back key released.
        Bit 4: 0=Go key pressed, 1=Go key released
    */
    unsigned char masque = 0;

    #ifdef DEBUG_PlaybackWing
    qDebug("0x%02X", champTouche);
    #endif
    masque = 1 << PAGE_UP;
    etatTouchesControle.touchePageUp = (champTouche & masque) >> PAGE_UP;
    masque = 1 << PAGE_DOWN;
    etatTouchesControle.touchePageDown = (champTouche & masque) >> PAGE_DOWN;
    masque = 1 << BACK;
    etatTouchesControle.toucheBack = (champTouche & masque) >> BACK;
    masque = 1 << GO;
    etatTouchesControle.toucheGo = (champTouche & masque) >> GO;

}
```

8.15.3.4 void PlaybackWing : :demarrer ( )

Renvoie

void

Références ReceptionnerDatagrammes(), statut, et udpSocket.

Référencé par IHM : :demarrer(), et IHM : :IHM().

```
{
    //qDebug() << Q_FUNC_INFO;
    if(statut)
    {
       connect(udpSocket, SIGNAL(readyRead()), this, SLOT(
      ReceptionnerDatagrammes()));
       //connect(this, SIGNAL(terminer()), this, SLOT(quit()));
    }
}
```

8.15.3.5 void PlaybackWing : :enregistrerModificationConsole (Console *console*, int *numeroConsoleAModifier*) [slot]

Paramètres

Console	console

Renvoie

void

Références consoles, XMLUtilitaire : :ecrireConsole(), et utilitaireDocuments.

Référencé par IHM : :envoyerModificationConsole().

```
{
    //qDebug() << "nombre de consoles" << consoles.count();
    consoles.removeAt(numeroConsoleAModifier);
    consoles.append(console);
    utilitaireDocuments->ecrireConsole(consoles);
}
```

8.15.3.6 void PlaybackWing : :envoyerCanalDepartWing (int *canalDepart*) [slot]

Renvoie

void

Références consoles, debuguerDatagramme(), et udpSocket.

Référencé par IHM : :envoyerCanalDepartWing(), et IHM : :~IHM().

```
{
    //qDebug()<< Q_FUNC_INFO << canalDepart;

    QByteArray donnees = '\0';
    donnees.append("WIDD");
    donnees.append((char)1);
    for(int i = 0; i < 32; i++)
    {
        donnees.append((char)0);
    }
    donnees.append((char)(canalDepart/10));
    for(int i = 0; i < 4; i++)
    {
        donnees.append((char)0);
    }

    //qDebug()<< Q_FUNC_INFO << (QString)donnees;
    debuguerDatagramme(donnees.data(), donnees.length());
    for(int i = 0; i < consoles.count(); i++)
    {
        udpSocket->writeDatagram(donnees, (QHostAddress)consoles.at(i).
      adresseIP, consoles.at(i).port);
    }

}
```

8.15.3.7 void PlaybackWing : :envoyerEtatFaders (EtatFaders *etatFaders*) [signal]

Référencé par TraiterDatagramme().

8.15.3.8 void PlaybackWing : :envoyerEtatTouchesControle (EtatTouchesControle *etatTouchesControle*) [signal]

Référencé par TraiterDatagramme().

8.15.3.9 void PlaybackWing : :ExtraireFaders (const QByteArray & *datagramme*) [private]

Paramètres

datagramme	la trame wing reçue

Renvoie

void

Références etatFaders, EtatFaders : :fader0, EtatFaders : :fader1, EtatFaders : :fader2, EtatFaders : :fader3, EtatFaders : :fader4, EtatFaders : :fader5, EtatFaders : :fader6, EtatFaders : :fader7, EtatFaders : :fader8, EtatFaders : :fader9, FADER_NUMERO_0, FADER_NUMERO_1, FADER_NUMERO_2, FADER_NUMERO_3, FADER_NUMERO_4, FADER_NUMERO_5, FADER_NUMERO_6, FADER_NUMERO_7, FADER_NUMERO_8, et FADER_NUMERO_9.

Référencé par TraiterDatagramme().

```
{
    //qDebug() << Q_FUNC_INFO;
    /*
        Fader state for faders 0 to 9. Valid range is 0 to 255.
    */
    etatFaders.fader0 = (unsigned char)datagramme[FADER_NUMERO_0];
    etatFaders.fader1 = (unsigned char)datagramme[FADER_NUMERO_1];
    etatFaders.fader2 = (unsigned char)datagramme[FADER_NUMERO_2];
    etatFaders.fader3 = (unsigned char)datagramme[FADER_NUMERO_3];
    etatFaders.fader4 = (unsigned char)datagramme[FADER_NUMERO_4];
    etatFaders.fader5 = (unsigned char)datagramme[FADER_NUMERO_5];
    etatFaders.fader6 = (unsigned char)datagramme[FADER_NUMERO_6];
    etatFaders.fader7 = (unsigned char)datagramme[FADER_NUMERO_7];
    etatFaders.fader8 = (unsigned char)datagramme[FADER_NUMERO_8];
    etatFaders.fader9 = (unsigned char)datagramme[FADER_NUMERO_9];

    //qDebug() << "Etat faders" << ":" << etatFaders.fader0 <<
       etatFaders.fader1 << etatFaders.fader2 << etatFaders.fader3 << etatFaders.fader4 <<
       etatFaders.fader5 << etatFaders.fader6 << etatFaders.fader7 << etatFaders.fader8 <<
       etatFaders.fader9;
}
```

8.15.3.10 QString PlaybackWing : :getAdresseIPConsole (int *numeroConsole*)

Renvoie

QString

Références consoles.

Référencé par IHM : :creerIHM(), IHM : :envoyerModificationConsole(), et IHM-: :supprimerConsole().

```
{
    return consoles.at(numeroConsole).adresseIP;
}
```

8.15.3.11 Console PlaybackWing : :getConsole (int *numeroConsole*)

Références consoles.

Référencé par IHM : :ajouterConsole().

```
{
    return consoles[numeroConsole];
}
```

8.15.3.12 QList< Console > : :iterator PlaybackWing : :getDebutListeConsoles ( )

Références consoles.

Référencé par IHM : :supprimerConsole().

```
{
    return consoles.begin();
}
```

8.15.3.13 QList< Console > PlaybackWing : :getListeConsoles ( )

Références consoles.

Référencé par IHM : :supprimerConsole().

```
{
    return consoles;
}
```

8.15.3.14 int PlaybackWing : :getNombreConsoles ( )

Renvoie

int

Références consoles.

Référencé par IHM : :ajouterConsole(), IHM : :creerIHM(), IHM : :envoyerModification-Console(), et IHM : :supprimerConsole().

```
{
    return consoles.count();
}
```

8.15.3.15 quint16 PlaybackWing : :getPort (int *numeroConsole*)

Références consoles.

Référencé par IHM : :envoyerModificationConsole().

```
{
    return consoles.at(numeroConsole).port;
}
```

8.15.3.16 void PlaybackWing : :ReceptionnerDatagrammes ( ) [slot]

Renvoie

void

Références consoles, terminer(), TraiterDatagramme(), udpSocket, et VerifierDatagramme().

Référencé par demarrer().

```
{
   //qDebug() << Q_FUNC_INFO;
   int nbOctets = 0;
   int etat;

   // datagramme en attente d'être lu ?
   while (udpSocket->hasPendingDatagrams())
   {
      QByteArray donneesDatagramme;
      QHostAddress emetteurAdresse;
      quint16 emetteurPort;


      // Fixe la taille du tableau au nombre d'octets reçus en attente
      donneesDatagramme.resize(udpSocket->pendingDatagramSize());

      // Lit le datagramme en attente
      //nbOctets = udpSocket->readDatagram(datagram.data(), datagram.size());
      nbOctets = udpSocket->readDatagram(donneesDatagramme.data(),
      donneesDatagramme.size(), &emetteurAdresse, &emetteurPort);

      // Vérifie si le datagramme est celui d'une console
      for(int i = 0; i < consoles.count(); i++)
      {
        if(emetteurAdresse == (QHostAddress)consoles.at(i).adresseIP &&
      emetteurPort == (quint16)consoles.at(i).port)
        {
            // Vérifie la validité du datagramme
            etat = VerifierDatagramme(donneesDatagramme.data(), nbOctets);
            if(etat == 1)
            {
                #ifdef DEBUG_PlaybackWing
                QString qs_emetteurAdresse = emetteurAdresse.toString();
                cout << "<" << qs_emetteurAdresse.toStdString() << ":" <<
      emetteurPort << "> datagramme de " << nbOctets << " octet(s) reçu(s)" << endl;
                #endif

                TraiterDatagramme(donneesDatagramme);
                //debuguerDatagramme(donneesDatagramme.data(),
       donneesDatagramme.size());
            }
            else
            {
                #ifdef DEBUG_PlaybackWing
                QString qs_emetteurAdresse = emetteurAdresse.toString();
                cout << "<" << qs_emetteurAdresse.toStdString() << ":" <<
      emetteurPort << "> datagramme de " << nbOctets << " octet(s) reçu(s) : INVALIDE !" <<
       endl;
                #endif
                emit terminer(); /* sinon on quitte ! (à modifier par la suite)
       */
            }
        }
      }
   }
}
```

8.15.3.17 void PlaybackWing : :supprimerConsoleListe (QList< Console > : :iterator *it*)

Références consoles.

Référencé par IHM : :supprimerConsole().

```
{
    consoles.erase(it);
}
```

8.15.3.18 void PlaybackWing : :terminer ( ) [signal]

Référencé par ReceptionnerDatagrammes().

8.15.3.19 int PlaybackWing : :TraiterDatagramme (const QByteArray & *datagramme*) [private]

Paramètres

datagramme	le datagramme a traiter

Renvoie

int

Références DecoderTouchesControle(), envoyerEtatFaders(), envoyerEtatTouches-Controle(), etatFaders, etatTouchesControle, ExtraireFaders(), et INDEX_TOUCHES_-CONTROLE.

Référencé par ReceptionnerDatagrammes().

```
{
    DecoderTouchesControle((unsigned char)datagramme[INDEX_TOUCHES_CONTROLE]);
    ExtraireFaders(datagramme);

    emit envoyerEtatTouchesControle(etatTouchesControle);
    emit envoyerEtatFaders(etatFaders);

    return 1;
}
```

8.15.3.20 int PlaybackWing : :VerifierDatagramme (char * *donneesBrutes,* int *nbOctets*) [private]

Paramètres

donnees-Brutes	char* les données reçues
nbOctets	int le nombre d'octets du datagramme

Renvoie

int 1 si c'est une trame WOOD sinon 0

Références INDEX_FIRMWARE.

Référencé par ReceptionnerDatagrammes().

```
{
    QString typeTrame = "";
```

```
    //qDebug() << donneesBrutes;
    //qDebug() << sizeof(donneesBrutes);
    if(sizeof(donneesBrutes) != 8)
    {
        return 0;
    }

    for(int i=0;i<INDEX_FIRMWARE;i++)
    {
        //qDebug() << donneesBrutes[i];
        typeTrame.append(donneesBrutes[i]);
    }
    //qDebug() << typeTrame;
    if(typeTrame == "WODD")
    {
        return 1;
    }
    else return 0;
}
```

8.15.4 Documentation des données membres

8.15.4.1 QList<Console> PlaybackWing : :consoles [private]

Référencé par ajouterConsoleListe(), enregistrerModificationConsole(), envoyerCanalDepartWing(), getAdresseIPConsole(), getConsole(), getDebutListeConsoles(), getListeConsoles(), getNombreConsoles(), getPort(), PlaybackWing(), ReceptionnerDatagrammes(), et supprimerConsoleListe().

8.15.4.2 EtatFaders PlaybackWing : :etatFaders [private]

Référencé par ExtraireFaders(), et TraiterDatagramme().

8.15.4.3 EtatTouchesControle PlaybackWing : :etatTouchesControle [private]

Référencé par DecoderTouchesControle(), et TraiterDatagramme().

8.15.4.4 bool PlaybackWing : :statut [private]

Référencé par demarrer(), et PlaybackWing().

8.15.4.5 QUdpSocket* PlaybackWing : :udpSocket [private]

Référencé par demarrer(), envoyerCanalDepartWing(), PlaybackWing(), ReceptionnerDatagrammes(), et ~PlaybackWing().

8.15.4.6 XMLUtilitaire* PlaybackWing : :utilitaireDocuments [private]

Référencé par ajouterConsoleListe(), enregistrerModificationConsole(), et PlaybackWing().

La documentation de cette classe a été générée à partir des fichiers suivants :
- PlaybackWing.h
- PlaybackWing.cpp

8.16 Référence de la classe PortSettings

The PortSettings class contain port settings.

`#include <qextserialport/qextserialport.h>`

Attributs publics

- BaudRateType BaudRate
- DataBitsType DataBits
- ParityType Parity
- StopBitsType StopBits
- FlowType FlowControl
- long Timeout_Millisec

8.16.1 Description détaillée

Structure to contain port settings.

```
BaudRateType BaudRate;
DataBitsType DataBits;
ParityType Parity;
StopBitsType StopBits;
FlowType FlowControl;
long Timeout_Millisec;
```

structure to contain port settings

8.16.2 Documentation des données membres

8.16.2.1 BaudRateType PortSettings : :BaudRate

TODO

Référencé par QextSerialPortPrivate : :QextSerialPortPrivate(), QextSerialPortPrivate : :setBaudRate(), QextSerialPortPrivate : :setPortSettings(), et QextSerialPortPrivate : :updatePortSettings().

8.16.2.2 DataBitsType PortSettings : :DataBits

TODO

Référencé par QextSerialPortPrivate : :QextSerialPortPrivate(), QextSerialPortPrivate : :setDataBits(), QextSerialPortPrivate : :setParity(), QextSerialPortPrivate : :setPortSettings(), QextSerialPortPrivate : :setStopBits(), et QextSerialPortPrivate : :updatePortSettings().

8.16.2.3 FlowType PortSettings : :FlowControl

TODO

Référencé par QextSerialPortPrivate : :QextSerialPortPrivate(), QextSerialPortPrivate : :setFlowControl(), QextSerialPortPrivate : :setPortSettings(), et QextSerialPortPrivate : :updatePortSettings().

8.16.2.4 ParityType PortSettings : :Parity

TODO

Référencé par QextSerialPortPrivate : :QextSerialPortPrivate(), QextSerialPortPrivate-: :setParity(), QextSerialPortPrivate : :setPortSettings(), et QextSerialPortPrivate-: :updatePortSettings().

8.16.2.5 StopBitsType PortSettings : :StopBits

TODO

Référencé par QextSerialPortPrivate : :QextSerialPortPrivate(), QextSerialPortPrivate-: :setDataBits(), QextSerialPortPrivate : :setPortSettings(), QextSerialPortPrivate : :set-StopBits(), et QextSerialPortPrivate : :updatePortSettings().

8.16.2.6 long PortSettings : :Timeout_Millisec

TODO

Référencé par QextSerialPortPrivate : :QextSerialPortPrivate(), QextSerialPortPrivate-: :setPortSettings(), QextSerialPortPrivate : :setTimeout(), et QextSerialPortPrivate-: :updatePortSettings().

La documentation de cette classe a été générée à partir du fichier suivant :
– qextserialport.h

8.17 Référence de la classe QextPortInfo

The QextPortInfo class containing port information.

```
#include <qextserialport/qextserialenumerator.h>
```

Attributs publics

– QString portName
  Port name. /∗∗< TODO ∗/.
– QString physName
  Physical name. /∗∗< TODO ∗/.
– QString friendName
  Friendly name. /∗∗< TODO ∗/.
– QString enumName
  Enumerator name. /∗∗< TODO ∗/.
– int vendorID
  Vendor ID. /∗∗< TODO ∗/.
– int productID
  Product ID /∗∗< TODO ∗/.

8.17.1 Description détaillée

Structure containing port information.

```
QString portName;
```

```
QString physName;
QString friendName;
QString enumName;
int vendorID;
int productID;      ///< Product ID
```

8.17.2 Documentation des données membres

8.17.2.1 QString QextPortInfo : :enumName

Référencé par getDeviceDetailsWin(), et QextSerialEnumeratorPrivate : :getPorts_sys().

8.17.2.2 QString QextPortInfo : :friendName

Référencé par getDeviceDetailsWin(), et QextSerialEnumeratorPrivate : :getPorts_sys().

8.17.2.3 QString QextPortInfo : :physName

Référencé par getDeviceDetailsWin(), QextSerialEnumeratorPrivate : :getPorts_sys(), et portInfoFromDevice().

8.17.2.4 QString QextPortInfo : :portName

Référencé par enumerateDevicesWin(), getDeviceDetailsWin(), QextSerialEnumeratorPrivate : :getPorts_sys(), lessThan(), et portInfoFromDevice().

8.17.2.5 int QextPortInfo : :productID

Référencé par enumerateDevicesWin(), getDeviceDetailsWin(), et portInfoFromDevice().

8.17.2.6 int QextPortInfo : :vendorID

Référencé par enumerateDevicesWin(), getDeviceDetailsWin(), et portInfoFromDevice().

La documentation de cette classe a été générée à partir du fichier suivant :
– qextserialenumerator.h

8.18 Référence de la classe QextReadBuffer

`#include <qextserialport/qextserialport_p.h>`

Fonctions membres publiques

– QextReadBuffer (size_t growth=4096)
– ~QextReadBuffer ()
– void clear ()
– int size () const
– bool isEmpty () const

- int read (char *target, int size)
- char * reserve (size_t size)
- void chop (int size)
- void squeeze ()
- QByteArray readAll ()
- int readLine (char *target, int size)
- bool canReadLine () const

Attributs privés

- int len
- char * first
- char * buf
- size_t capacity
- size_t basicBlockSize

8.18.1 Documentation des constructeurs et destructeur

8.18.1.1 QextReadBuffer : :QextReadBuffer (size_t *growth* = 4096)

Paramètres

growth	

```
    : len(0), first(0), buf(0), capacity(0), basicBlockSize(growth) {
}
```

8.18.1.2 QextReadBuffer : :~QextReadBuffer ( )

Références buf.

```
                    {
    delete [] buf;
}
```

8.18.2 Documentation des fonctions membres

8.18.2.1 QextReadBuffer : :canReadLine ( ) const

Renvoie

bool

Références first, et len.

```
                                {
    return memchr(first, '\n', len);
}
```

8.18.2.2 QextReadBuffer : :chop (int *size*)

Paramètres

size	

Références clear(), len, et size().

Référencé par QextSerialPortPrivate : :_q_canRead().

```
                                {
    if (size >= len) {
        clear();
    } else {
        len -= size;
    }
}
```

8.18.2.3 QextReadBuffer : :clear ( )

Références buf, first, et len.

Référencé par chop(), et readAll().

```
                          {
    first = buf;
    len = 0;
}
```

8.18.2.4 QextReadBuffer : :isEmpty ( ) const

Renvoie

bool

Références len.

```
                                {
    return len == 0;
}
```

8.18.2.5 QextReadBuffer : :read (char * *target*, int *size*)

Paramètres

target	
size	

Renvoie

int

Références first, et len.

```
                                                {
    int r = qMin(size, len);
    if (r == 1) {
        *target = *first;
        --len;
        ++first;
    } else {
        memcpy(target, first, r);
        len -= r;
        first += r;
```

```
    }
    return r;
}
```

8.18.2.6 QextReadBuffer : :readAll ( )

Renvoie

QByteArray

Références clear(), first, et len.

```
                                {
    char *f = first;
    int l = len;
    clear();
    return QByteArray(f, l);
}
```

8.18.2.7 QextReadBuffer : :readLine (char * *target*, int *size*)

Paramètres

target	
size	

Renvoie

int

Références first, et len.

```
                                                {
    int r = qMin(size, len);
    char *eol = static_cast<char *>(memchr(first, '\n', r));
    if (eol)
        r = 1+(eol-first);
    memcpy(target, first, r);
    len -= r;
    first += r;
    return int(r);
}
```

8.18.2.8 QextReadBuffer : :reserve (size_t *size*)

Paramètres

size	

Renvoie

char

Références basicBlockSize, buf, capacity, first, et len.

Référencé par QextSerialPortPrivate : :_q_canRead().

```
{
```

```
    if ((first - buf) + len + size > capacity) {
        size_t newCapacity = qMax(capacity, basicBlockSize);
        while (newCapacity < len + size)
            newCapacity *= 2;
        if (newCapacity > capacity) {
            // allocate more space
            char *newBuf = new char[newCapacity];
            memmove(newBuf, first, len);
            delete [] buf;
            buf = newBuf;
            capacity = newCapacity;
        } else {
            // shift any existing data to make space
            memmove(buf, first, len);
        }
        first = buf;
    }
    char *writePtr = first + len;
    len += (int)size;
    return writePtr;
}
```

8.18.2.9 QextReadBuffer : :size ( ) const

Renvoie

int

Références len.

Référencé par chop().

```
                          {
    return len;
}
```

8.18.2.10 QextReadBuffer : :squeeze ( )

Références basicBlockSize, buf, capacity, first, et len.

```
                        {
    if (first != buf) {
        memmove(buf, first, len);
        first = buf;
    }
    size_t newCapacity = basicBlockSize;
    while (newCapacity < size_t(len))
        newCapacity *= 2;
    if (newCapacity < capacity) {
        char *tmp = static_cast<char *>(realloc(buf, newCapacity));
        if (tmp) {
            buf = tmp;
            capacity = newCapacity;
        }
    }
}
```

8.18.3 Documentation des données membres

8.18.3.1 size_t QextReadBuffer : :basicBlockSize [private]

TODO

Référencé par reserve(), et squeeze().

8.18.3.2 char* QextReadBuffer : :buf [private]

TODO

Référencé par clear(), reserve(), squeeze(), et ~QextReadBuffer().

8.18.3.3 size_t QextReadBuffer : :capacity [private]

TODO

Référencé par reserve(), et squeeze().

8.18.3.4 char* QextReadBuffer : :first [private]

TODO

Référencé par canReadLine(), clear(), read(), readAll(), readLine(), reserve(), et squeeze().

8.18.3.5 int QextReadBuffer : :len [private]

TODO

Référencé par canReadLine(), chop(), clear(), isEmpty(), read(), readAll(), readLine(), reserve(), size(), et squeeze().

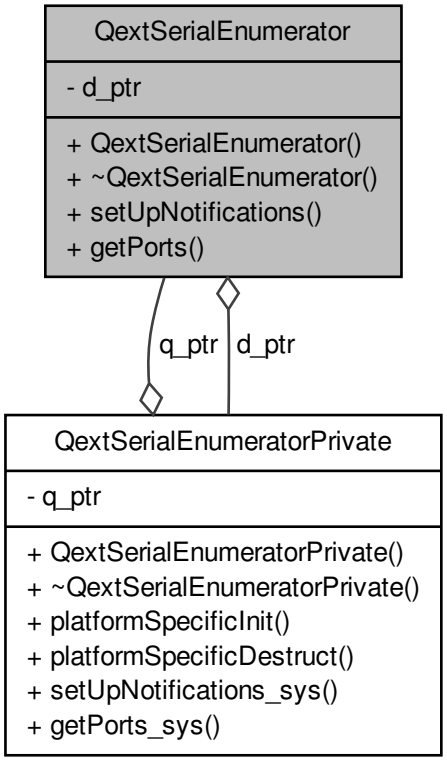
La documentation de cette classe a été générée à partir du fichier suivant :
– qextserialport_p.h

8.19 Référence de la classe QextSerialEnumerator

The QextSerialEnumerator class provides list of ports available in the system.

```
#include <qextserialport/qextserialenumerator.h>
```

Graphe de collaboration de QextSerialEnumerator :



Signaux

– void deviceDiscovered (const QextPortInfo &info)
– void deviceRemoved (const QextPortInfo &info)

Fonctions membres publiques

– QextSerialEnumerator (QObject ∗parent=0)
– ∼QextSerialEnumerator ()
– void setUpNotifications ()

Fonctions membres publiques statiques

– static QList< QextPortInfo > getPorts ()

Attributs privés

– QextSerialEnumeratorPrivate * d_ptr

8.19.1 Description détaillée

8.19.2 Documentation des constructeurs et destructeur

8.19.2.1 QextSerialEnumerator : :QextSerialEnumerator (QObject * *parent* = 0)

Constructs a QextSerialEnumerator object with the given *parent*.

Paramètres

parent	

```
    :QObject(parent), d_ptr(new QextSerialEnumeratorPrivate(this))
{
    if (!QMetaType::isRegistered(QMetaType::type("QextPortInfo")))
        qRegisterMetaType<QextPortInfo>("QextPortInfo");
}
```

8.19.2.2 QextSerialEnumerator : :~QextSerialEnumerator ( )

Destructs the QextSerialEnumerator object.

Références d_ptr.

```
{
    delete d_ptr;
}
```

8.19.3 Documentation des fonctions membres

8.19.3.1 QextSerialEnumerator : :deviceDiscovered (const QextPortInfo & *info*) [signal]

A new device has been connected to the system.

setUpNotifications() must be called first to enable event-driven device notifications. - Currently only implemented on Windows and OS X.

info The device that has been discovered.

Paramètres

info	

8.19.3.2 QextSerialEnumerator : :deviceRemoved (const QextPortInfo & *info*) [signal]

A device has been disconnected from the system.

setUpNotifications() must be called first to enable event-driven device notifications. -

Currently only implemented on Windows and OS X.

info The device that was disconnected.

Paramètres

info	

8.19.3.3 QextSerialEnumerator : :getPorts ( ) [static]

Get list of ports.

return list of ports currently available in the system.

Renvoie

QList<QextPortInfo>

Références QextSerialEnumeratorPrivate : :getPorts_sys().

```
{
    return QextSerialEnumeratorPrivate::getPorts_sys();
}
```

8.19.3.4 QextSerialEnumerator : :setUpNotifications ( )

Enable event-driven notifications of board discovery/removal.

Références QESP_WARNING.

```
{
    Q_D(QextSerialEnumerator);
    if (!d->setUpNotifications_sys(true))
        QESP_WARNING("Setup Notification Failed...");
}
```

8.19.4 Documentation des données membres

8.19.4.1 QextSerialEnumeratorPrivate* QextSerialEnumerator : :d_ptr [private]

TODO

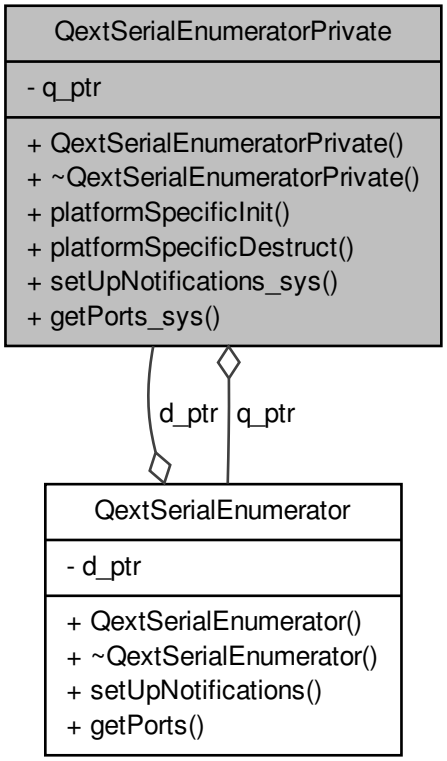
Référencé par ~QextSerialEnumerator().

La documentation de cette classe a été générée à partir des fichiers suivants :
- qextserialenumerator.h
- qextserialenumerator.cpp

8.20 Référence de la classe QextSerialEnumeratorPrivate

```
#include <qextserialport/qextserialenumerator_p.h>
```

Graphe de collaboration de QextSerialEnumeratorPrivate :



Fonctions membres publiques

- QextSerialEnumeratorPrivate (QextSerialEnumerator ∗enumrator)
- ∼QextSerialEnumeratorPrivate ()
- void platformSpecificInit ()
- void platformSpecificDestruct ()
- bool setUpNotifications_sys (bool setup)

Fonctions membres publiques statiques

- static QList< QextPortInfo > getPorts_sys ()

Attributs privés

- QextSerialEnumerator ∗ q_ptr

8.20.1 Documentation des constructeurs et destructeur

8.20.1.1 QextSerialEnumeratorPrivate : :QextSerialEnumeratorPrivate (QextSerialEnumerator ∗ *enumrator*)

Paramètres

enumrator	

Références platformSpecificInit().

```
    :q_ptr(enumrator)
{
    platformSpecificInit();
}
```

8.20.1.2 QextSerialEnumeratorPrivate : :∼QextSerialEnumeratorPrivate ( )

Références platformSpecificDestruct().

```
{
    platformSpecificDestruct();
}
```

8.20.2 Documentation des fonctions membres

8.20.2.1 QList< QextPortInfo > QextSerialEnumeratorPrivate : :getPorts_sys ( ) [static]

Renvoie

QList<QextPortInfo>

Get list of ports.

return list of ports currently available in the system.

Références QextPortInfo : :enumName, QextPortInfo : :friendName, QextPortInfo-: :physName, portInfoFromDevice(), et QextPortInfo : :portName.

Référencé par QextSerialEnumerator : :getPorts(), et setUpNotifications_sys().

```
{
    QList<QextPortInfo> infoList;
#ifndef QESP_NO_UDEV

    struct udev *ud = udev_new();
    if (!ud) {
        qCritical() << "Unable to enumerate ports because udev is not
       initialized.";
        return infoList;
    }

    struct udev_enumerate *enumerate = udev_enumerate_new(ud);
    udev_enumerate_add_match_subsystem(enumerate, "tty");
    udev_enumerate_scan_devices(enumerate);
    struct udev_list_entry *list = udev_enumerate_get_list_entry(enumerate);
    struct udev_list_entry *entry;
    udev_list_entry_foreach(entry, list) {
```

```
        const char *path;
        struct udev_device *dev;

        // Have to grab the actual udev device here...
        path = udev_list_entry_get_name(entry);
        dev = udev_device_new_from_syspath(ud, path);

        infoList.append(portInfoFromDevice(dev));

        // Done with this device
        udev_device_unref(dev);
    }
    // Done with the list and this udev
    udev_enumerate_unref(enumerate);
    udev_unref(ud);
#else

    QStringList portNamePrefixes, portNameList;
    portNamePrefixes << QLatin1String("ttyS*"); // list normal serial ports
       first

    QDir dir(QLatin1String("/dev"));
    portNameList = dir.entryList(portNamePrefixes, (QDir::System | QDir::Files)
      , QDir::Name);

    // remove the values which are not serial ports for e.g.  /dev/ttysa
    for (int i = 0; i < portNameList.size(); i++) {
        bool ok;
        QString current = portNameList.at(i);
        // remove the ttyS part, and check, if the other part is a number
        current.remove(0,4).toInt(&ok, 10);
        if (!ok) {
            portNameList.removeAt(i);
            i--;
        }
    }

    // get the non standard serial ports names
    // (USB-serial, bluetooth-serial, 18F PICs, and so on)
    // if you know an other name prefix for serial ports please let us know
    portNamePrefixes.clear();
    portNamePrefixes << QLatin1String("ttyACM*") << QLatin1String("ttyUSB*") <<
       QLatin1String("rfcomm*");
    portNameList += dir.entryList(portNamePrefixes, (QDir::System | QDir::Files
      ), QDir::Name);

    foreach (QString str , portNameList) {
        QextPortInfo inf;
        inf.physName = QLatin1String("/dev/")+str;
        inf.portName = str;

        if (str.contains(QLatin1String("ttyS"))) {
            inf.friendName = QLatin1String("Serial port ")+str.remove(0, 4);
        }
        else if (str.contains(QLatin1String("ttyUSB"))) {
            inf.friendName = QLatin1String("USB-serial adapter ")+str.remove(0,
       6);
        }
        else if (str.contains(QLatin1String("rfcomm"))) {
            inf.friendName = QLatin1String("Bluetooth-serial adapter ")+str.
      remove(0, 6);
        }
        inf.enumName = QLatin1String("/dev"); // is there a more helpful name
       for this?
        infoList.append(inf);
    }
#endif


    return infoList;
}
```

8.20.2.2 void QextSerialEnumeratorPrivate : :platformSpecificDestruct ( )

default

Référencé par ~QextSerialEnumeratorPrivate().

```
{
#ifndef QESP_NO_UDEV

    if (notifier) {
        notifier->setEnabled(false);
        delete notifier;
    }

    if (monitor)
        udev_monitor_unref(monitor);

    if (udev)
        udev_unref(udev);
#endif

}
```

8.20.2.3 void QextSerialEnumeratorPrivate : :platformSpecificInit ( )

Référencé par QextSerialEnumeratorPrivate().

```
{
#ifndef QESP_NO_UDEV

    monitor = NULL;
    notifierFd = -1;
    notifier = NULL;

    udev = udev_new();
    if (!udev)
        qCritical() << "Unable to initialize udev notifications";
#endif

}
```

8.20.2.4 bool QextSerialEnumeratorPrivate : :setUpNotifications_sys (bool *setup*)

Paramètres

setup	

Renvoie

bool

Références getPorts_sys().

```
{
    Q_UNUSED(setup);
#ifndef QESP_NO_UDEV

    Q_Q(QextSerialEnumerator);
    if (!udev) {
        qCritical() << "Unable to initialize notifications because udev is not
       initialized.";
        return false;
    }
```

```
    // Emit signals immediately for devices already connected (Windows version
       seems to behave
    // this way)
    foreach (QextPortInfo i, getPorts_sys())
        Q_EMIT q->deviceDiscovered(i);

    // Look for tty devices from udev.
    monitor = udev_monitor_new_from_netlink(udev, "udev");
    udev_monitor_filter_add_match_subsystem_devtype(monitor, "tty", NULL);
    udev_monitor_enable_receiving(monitor);
    notifierFd = udev_monitor_get_fd(monitor);
    notifier = new QSocketNotifier(notifierFd, QSocketNotifier::Read);
    q->connect(notifier, SIGNAL(activated(int)), q, SLOT(_q_deviceEvent()));
    notifier->setEnabled(true);

    return true;
#else

    return false;
#endif

}
```

8.20.3 Documentation des données membres

8.20.3.1 QextSerialEnumerator* QextSerialEnumeratorPrivate : :q_ptr [private]

TODO

La documentation de cette classe a été générée à partir des fichiers suivants :
- qextserialenumerator_p.h
- qextserialenumerator.cpp
- qextserialenumerator_linux.cpp
- qextserialenumerator_osx.cpp
- qextserialenumerator_unix.cpp
- qextserialenumerator_win.cpp

8.21 Référence de la classe QextSerialPort

The QextSerialPort class encapsulates a serial port on both POSIX and Windows systems.

`#include <qextserialport/qextserialport.h>`

Types publics

- enum QueryMode { Polling, EventDriven }

Connecteurs publics

- void setPortName (const QString &name)
- void setQueryMode (QueryMode mode)
- void setBaudRate (BaudRateType)
- void setDataBits (DataBitsType)
- void setParity (ParityType)
- void setStopBits (StopBitsType)

- void setFlowControl (FlowType)
- void setTimeout (long)
- void setDtr (bool set=true)
- void setRts (bool set=true)

Signaux

- void dsrChanged (bool status)

Fonctions membres publiques

- QextSerialPort (QueryMode mode=EventDriven, QObject *parent=0)
- QextSerialPort (const QString &name, QueryMode mode=EventDriven, QObject *parent=0)
- QextSerialPort (const PortSettings &s, QueryMode mode=EventDriven, QObject *parent=0)
- QextSerialPort (const QString &name, const PortSettings &s, QueryMode mode=EventDriven, QObject *parent=0)
- ~QextSerialPort ()
- QString portName () const
- QueryMode queryMode () const
- BaudRateType baudRate () const
- DataBitsType dataBits () const
- ParityType parity () const
- StopBitsType stopBits () const
- FlowType flowControl () const
- bool open (OpenMode mode)
- bool isSequential () const
- void close ()
- void flush ()
- qint64 bytesAvailable () const
- bool canReadLine () const
- QByteArray readAll ()
- ulong lastError () const
- ulong lineStatus ()
- QString errorString ()

Fonctions membres protégées

- qint64 readData (char *data, qint64 maxSize)
- qint64 writeData (const char *data, qint64 maxSize)

Propriétés

- QString portName
- QueryMode queryMode

Fonctions membres privées

- Q_PRIVATE_SLOT (d_func(), void _q_canRead()) QextSerialPortPrivate *const d_ptr

8.21.1 Description détaillée

8.21.2 Documentation des énumérations membres

8.21.2.1 enum QextSerialPort : :QueryMode

This enum type specifies query mode used in a serial port :

Polling asynchronously read and write EventDriven synchronously read and write

Valeurs énumérées :

Polling

EventDriven

```
                {
    Polling,
    EventDriven
};
```

8.21.3 Documentation des constructeurs et destructeur

8.21.3.1 QextSerialPort : :QextSerialPort (QextSerialPort : :QueryMode *mode* = EventDriven, QObject * *parent* = 0) [explicit]

Default constructor. Note that the name of the device used by a QextSerialPort is dependent on your OS. Possible naming conventions and their associated OS are :

```
OS Constant       Used By         Naming Convention
-------------     -------------   ------------------------
Q_OS_WIN          Windows         COM1, COM2
Q_OS_IRIX         SGI/IRIX        /dev/ttyf1, /dev/ttyf2
Q_OS_HPUX         HP-UX           /dev/tty1p0, /dev/tty2p0
Q_OS_SOLARIS      SunOS/Slaris    /dev/ttya, /dev/ttyb
Q_OS_OSF          Digital UNIX    /dev/tty01, /dev/tty02
Q_OS_FREEBSD      FreeBSD         /dev/ttyd0, /dev/ttyd1
Q_OS_OPENBSD      OpenBSD         /dev/tty00, /dev/tty01
Q_OS_LINUX        Linux           /dev/ttyS0, /dev/ttyS1
<none>                            /dev/ttyS0, /dev/ttyS1
```

This constructor assigns the device name to the name of the first port on the specified system. See the other constructors if you need to open a different port. Default *mode* is EventDriven. As a subclass of QObject, *parent* can be specified.

Références setPortName(), et setQueryMode().

```
    : QIODevice(parent), d_ptr(new QextSerialPortPrivate(this))
{
#ifdef Q_OS_WIN

    setPortName(QLatin1String("COM1"));

#elif defined(Q_OS_IRIX)

    setPortName(QLatin1String("/dev/ttyf1"));

#elif defined(Q_OS_HPUX)

    setPortName(QLatin1String("/dev/tty1p0"));

#elif defined(Q_OS_SOLARIS)

    setPortName(QLatin1String("/dev/ttya"));
```

```
#elif defined(Q_OS_OSF) //formally DIGITAL UNIX

    setPortName(QLatin1String("/dev/tty01"));

#elif defined(Q_OS_FREEBSD)

    setPortName(QLatin1String("/dev/ttyd1"));

#elif defined(Q_OS_OPENBSD)

    setPortName(QLatin1String("/dev/tty00"));

#else

    setPortName(QLatin1String("/dev/ttyS0"));
#endif

    setQueryMode(mode);
}
```

8.21.3.2 QextSerialPort : :QextSerialPort (const QString & *name*, QextSerialPort : :QueryMode *mode* = EventDriven, QObject ∗ *parent* = 0) [explicit]

Constructs a serial port attached to the port specified by name. *name* is the name of the device, which is windowsystem-specific, e.g."COM1" or "/dev/ttyS0". *mode*

Références setPortName(), et setQueryMode().

```
    : QIODevice(parent), d_ptr(new QextSerialPortPrivate(this))
{
    setQueryMode(mode);
    setPortName(name);
}
```

8.21.3.3 QextSerialPort : :QextSerialPort (const PortSettings & *settings*, QextSerialPort : :QueryMode *mode* = EventDriven, QObject ∗ *parent* = 0) [explicit]

Constructs a port with default name and specified *settings*.

Références setQueryMode().

```
    : QIODevice(parent), d_ptr(new QextSerialPortPrivate(this))
{
    Q_D(QextSerialPort);
    setQueryMode(mode);
    d->setPortSettings(settings);
}
```

8.21.3.4 QextSerialPort : :QextSerialPort (const QString & *name*, const PortSettings & *settings*, QextSerialPort : :QueryMode *mode* = EventDriven, QObject ∗ *parent* = 0)

Constructs a port with specified *name* , *mode* and *settings*.

Références setPortName(), et setQueryMode().

```
    : QIODevice(parent), d_ptr(new QextSerialPortPrivate(this))
```

```
{
    Q_D(QextSerialPort);
    setPortName(name);
    setQueryMode(mode);
    d->setPortSettings(settings);
}
```

8.21.3.5 QextSerialPort : :~QextSerialPort ( )

Destructs the QextSerialPort object.

Références close().

```
{
    if (isOpen()) {
        close();
    }
    delete d_ptr;
}
```

8.21.4 Documentation des fonctions membres

8.21.4.1 QextSerialPort : :baudRate ( ) const

Returns the baud rate of the serial port. For a list of possible return values see the definition of the enum BaudRateType.

Renvoie

BaudRateType

```
{
    QReadLocker locker(&d_func()->lock);
    return d_func()->settings.BaudRate;
}
```

8.21.4.2 QextSerialPort : :bytesAvailable ( ) const

Returns the number of bytes waiting in the port's receive queue. This function will return 0 if the port is not currently open, or -1 on error.

Renvoie

qint64

Référencé par readAll().

```
{
    QWriteLocker locker(&d_func()->lock);
    if (isOpen()) {
        qint64 bytes = d_func()->bytesAvailable_sys();
        if (bytes != -1) {
            return bytes + d_func()->readBuffer.size()
                    + QIODevice::bytesAvailable();
        } else {
            return -1;
        }
    }
    return 0;
}
```

8.21.4.3 QextSerialPort : :canReadLine ( ) const

Renvoie

bool

```
{
    QReadLocker locker(&d_func()->lock);
    return QIODevice::canReadLine() || d_func()->readBuffer.canReadLine();
}
```

8.21.4.4 QextSerialPort : :close ( )

Closes a serial port. This function has no effect if the serial port associated with the class is not currently open.

Référencé par EnttecDMXUSB : :closePort(), et ~QextSerialPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (isOpen()) {
        // Be a good QIODevice and call QIODevice::close() before really
       close()
        //  so the aboutToClose() signal is emitted at the proper time
        QIODevice::close(); // mark ourselves as closed
        d->close_sys();
        d->readBuffer.clear();
    }
}
```

8.21.4.5 QextSerialPort : :dataBits ( ) const

Returns the number of data bits used by the port. For a list of possible values returned by this function, see the definition of the enum DataBitsType.

Renvoie

DataBitsType

```
{
    QReadLocker locker(&d_func()->lock);
    return d_func()->settings.DataBits;
}
```

8.21.4.6 QextSerialPort : :dsrChanged (bool *status*) [signal]

This signal is emitted whenever dsr line has changed its state. You may use this signal to check if device is connected.

status true when DSR signal is on, false otherwise.

Paramètres

status	

8.21.4.7 QextSerialPort : :errorString ( )

Returns a human-readable description of the last device error that occurred.

Renvoie

QString

Références E_AGAIN, E_BREAK_CONDITION, E_BUFFER_OVERRUN, E_CAUGHT_NON_BLOCKED_SIGNAL, E_FILE_NOT_FOUND, E_FRAMING_ERROR, E_INVALID_DEVICE, E_INVALID_FD, E_IO_ERROR, E_NO_ERROR, E_NO_MEMORY, E_PERMISSION_DENIED, E_PORT_TIMEOUT, E_READ_FAILED, E_RECEIVE_OVERFLOW, E_RECEIVE_PARITY_ERROR, E_TRANSMIT_OVERFLOW, E_WRITE_FAILED, et portName().

```
{
    Q_D(QextSerialPort);
    QReadLocker locker(&d->lock);
    switch(d->lastErr) {
    case E_NO_ERROR:
        return tr("No Error has occurred");
    case E_INVALID_FD:
        return tr("Invalid file descriptor (port was not opened correctly)");
    case E_NO_MEMORY:
        return tr("Unable to allocate memory tables (POSIX)");
    case E_CAUGHT_NON_BLOCKED_SIGNAL:
        return tr("Caught a non-blocked signal (POSIX)");
    case E_PORT_TIMEOUT:
        return tr("Operation timed out (POSIX)");
    case E_INVALID_DEVICE:
        return tr("The file opened by the port is not a valid device");
    case E_BREAK_CONDITION:
        return tr("The port detected a break condition");
    case E_FRAMING_ERROR:
        return tr("The port detected a framing error (usually caused by
       incorrect baud rate settings)");
    case E_IO_ERROR:
        return tr("There was an I/O error while communicating with the port");
    case E_BUFFER_OVERRUN:
        return tr("Character buffer overrun");
    case E_RECEIVE_OVERFLOW:
        return tr("Receive buffer overflow");
    case E_RECEIVE_PARITY_ERROR:
        return tr("The port detected a parity error in the received data");
    case E_TRANSMIT_OVERFLOW:
        return tr("Transmit buffer overflow");
    case E_READ_FAILED:
        return tr("General read operation failure");
    case E_WRITE_FAILED:
        return tr("General write operation failure");
    case E_FILE_NOT_FOUND:
        return tr("The %1 file doesn't exists").arg(this->portName());
    case E_PERMISSION_DENIED:
        return tr("Permission denied");
    case E_AGAIN:
        return tr("Device is already locked");
    default:
        return tr("Unknown error: %1").arg(d->lastErr);
    }
}
```

8.21.4.8 QextSerialPort : :flowControl ( ) const

Returns the type of flow control used by the port. For a list of possible values returned by this function, see the definition of the enum FlowType.

Renvoie

FlowType

```
{
    QReadLocker locker(&d_func()->lock);
    return d_func()->settings.FlowControl;
}
```

8.21.4.9 QextSerialPort : :flush ( )

Flushes all pending I/O to the serial port. This function has no effect if the serial port associated with the class is not currently open.

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (isOpen())
        d->flush_sys();
}
```

8.21.4.10 QextSerialPort : :isSequential ( ) const

Returns true if device is sequential, otherwise returns false. Serial port is sequential device so this function always returns true. Check QIODevice : :isSequential() documentation for more information.

Renvoie

bool

```
{
    return true;
}
```

8.21.4.11 QextSerialPort : :lastError ( ) const

Return the error number, or 0 if no error occurred.

Renvoie

ulong

```
{
    QReadLocker locker(&d_func()->lock);
    return d_func()->lastErr;
}
```

8.21.4.12 QextSerialPort : :lineStatus ( )

Returns the line status as stored by the port function. This function will retrieve the states of the following lines : DCD, CTS, DSR, and RI. On POSIX systems, the following additional lines can be monitored : DTR, RTS, Secondary TXD, and Secondary RXD. The value returned is an unsigned long with specific bits indicating which lines are high. The following constants should be used to examine the states of individual lines :

```
Mask        Line
------      ----
LS_CTS      CTS
LS_DSR      DSR
LS_DCD      DCD
LS_RI       RI
LS_RTS      RTS (POSIX only)
LS_DTR      DTR (POSIX only)
LS_ST       Secondary TXD (POSIX only)
LS_SR       Secondary RXD (POSIX only)
```

This function will return 0 if the port associated with the class is not currently open.

Renvoie

ulong

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (isOpen())
        return d->lineStatus_sys();
    return 0;
}
```

8.21.4.13 QextSerialPort : :open (OpenMode *mode*)

Opens a serial port and sets its OpenMode to *mode*. Note that this function does not specify which device to open. Returns true if successful ; otherwise returns false.This function has no effect if the port associated with the class is already open. The port is also configured to the current settings, as stored in the settings structure.

Paramètres

mode	

Renvoie

bool

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (mode != QIODevice::NotOpen && !isOpen())
        d->open_sys(mode);

    return isOpen();
}
```

8.21.4.14 QextSerialPort : :parity ( ) const

Returns the type of parity used by the port. For a list of possible values returned by this function, see the definition of the enum ParityType.

Renvoie

ParityType

```
{
    QReadLocker locker(&d_func()->lock);
    return d_func()->settings.Parity;
}
```

8.21.4.15 QString QextSerialPort : :portName ( ) const

Référencé par errorString().

8.21.4.16 QextSerialPort : :Q_PRIVATE_SLOT (d_func() , void *_q_canRead()*) const [private]

TODO

8.21.4.17 QueryMode QextSerialPort : :queryMode ( ) const

8.21.4.18 QextSerialPort : :readAll ( )

Reads all available data from the device, and returns it as a QByteArray. This function has no way of reporting errors ; returning an empty QByteArray() can mean either that no data was currently available for reading, or that an error occurred.

Renvoie

QByteArray

Références bytesAvailable().

```
{
    int avail = this->bytesAvailable();
    return (avail > 0) ? this->read(avail) : QByteArray();
}
```

8.21.4.19 QextSerialPort : :readData (char * *data*, qint64 *maxSize*) [protected]

Reads a block of data from the serial port. This function will read at most maxlen bytes from the serial port and place them in the buffer pointed to by data. Return value is the number of bytes actually read, or -1 on error.

Avertissement

before calling this function ensure that serial port associated with this class is currently open (use isOpen() function to check if port is open).

Paramètres

data	
maxSize	

Renvoie

qint64

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    qint64 bytesFromBuffer = 0;
    if (!d->readBuffer.isEmpty()) {
        bytesFromBuffer = d->readBuffer.read(data, maxSize);
        if (bytesFromBuffer == maxSize)
            return bytesFromBuffer;
    }
    qint64 bytesFromDevice = d->readData_sys(data+bytesFromBuffer, maxSize-
      bytesFromBuffer);
    if (bytesFromDevice < 0) {
        return -1;
    }
    return bytesFromBuffer + bytesFromDevice;
}
```

8.21.4.20 QextSerialPort : :setBaudRate (BaudRateType *baudRate*) [slot]

Sets the baud rate of the serial port to *baudRate*. Note that not all rates are applicable on all platforms. The following table shows translations of the various baud rate constants on Windows(including NT/2000) and POSIX platforms. Speeds marked with an $*$ are speeds that are usable on both Windows and POSIX.

RATE	Windows Speed	POSIX Speed
BAUD50	X	50
BAUD75	X	75
*BAUD110	110	110
BAUD134	X	134.5
BAUD150	X	150
BAUD200	X	200
*BAUD300	300	300
*BAUD600	600	600
*BAUD1200	1200	1200
BAUD1800	X	1800
*BAUD2400	2400	2400
*BAUD4800	4800	4800
*BAUD9600	9600	9600
BAUD14400	14400	X
*BAUD19200	19200	19200
*BAUD38400	38400	38400
BAUD56000	56000	X
*BAUD57600	57600	57600
BAUD76800	X	76800
*BAUD115200	115200	115200
BAUD128000	128000	X
BAUD230400	X	230400
BAUD256000	256000	X
BAUD460800	X	460800
BAUD500000	X	500000
BAUD576000	X	576000
BAUD921600	X	921600
BAUD1000000	X	1000000
BAUD1152000	X	1152000
BAUD1500000	X	1500000
BAUD2000000	X	2000000
BAUD2500000	X	2500000
BAUD3000000	X	3000000
BAUD3500000	X	3500000
BAUD4000000	X	4000000

Paramètres

BaudRate-Type	

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (d->settings.BaudRate != baudRate)
        d->setBaudRate(baudRate, true);
}
```

8.21.4.21 QextSerialPort : :setDataBits (DataBitsType *dataBits*) [slot]

Sets the number of data bits used by the serial port to *dataBits*. Possible values of dataBits are :

```
DATA_5      5 data bits
DATA_6      6 data bits
DATA_7      7 data bits
DATA_8      8 data bits
```

note : This function is subject to the following restrictions : 5 data bits cannot be used with 2 stop bits. 1.5 stop bits can only be used with 5 data bits. 8 data bits cannot be used with space parity on POSIX systems.

Paramètres

DataBits-Type	

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (d->settings.DataBits != dataBits)
        d->setDataBits(dataBits, true);
}
```

8.21.4.22 QextSerialPort : :setDtr (bool *set* = true) [slot]

Sets DTR line to the requested state (*set* default to high). This function will have no effect if the port associated with the class is not currently open.

Paramètres

set	

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (isOpen())
        d->setDtr_sys(set);
}
```

8.21.4.23 QextSerialPort : :setFlowControl (FlowType *flow*) [slot]

Sets the flow control used by the port to *flow*. Possible values of flow are :

```
FLOW_OFF            No flow control
FLOW_HARDWARE       Hardware (RTS/CTS) flow control
FLOW_XONXOFF        Software (XON/XOFF) flow control
```

Paramètres

FlowType	

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (d->settings.FlowControl != flow)
        d->setFlowControl(flow, true);
}
```

8.21.4.24 QextSerialPort : :setParity (ParityType *parity*) [slot]

Sets the parity associated with the serial port to *parity*. The possible values of parity are :

```
PAR_SPACE       Space Parity
PAR_MARK        Mark Parity
PAR_NONE        No Parity
PAR_EVEN        Even Parity
PAR_ODD         Odd Parity
```

Paramètres

ParityType	

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (d->settings.Parity != parity)
        d->setParity(parity, true);
}
```

8.21.4.25 QextSerialPort : :setPortName (const QString & *name*) [slot]

Sets the *name* of the device associated with the object, e.g. "COM1", or "/dev/ttyS0".

Paramètres

name	

Référencé par EnttecDMXUSB : :openPort(), et QextSerialPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    d->port = name;
}
```

8.21.4.26 QextSerialPort : :setQueryMode (QueryMode *mode*) [slot]

Set desired serial communication handling style. You may choose from polling or event driven approach. This function does nothing when port is open ; to apply changes port must be reopened.

In event driven approach read() and write() functions are acting asynchronously. They return immediately and the operation is performed in the background, so they doesn't freeze the calling thread. To determine when operation is finished, QextSerialPort runs separate thread and monitors serial port events. Whenever the event occurs, adequate signal is emitted.

When polling is set, read() and write() are acting synchronously. Signals are not working in this mode and some functions may not be available. The advantage of polling is that it generates less overhead due to lack of signals emissions and it doesn't start separate thread to monitor events.

Generally event driven approach is more capable and friendly, although some applications may need as low overhead as possible and then polling comes.

mode query mode.

Paramètres

mode	

Référencé par EnttecDMXUSB : :openPort(), et QextSerialPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (mode != d->queryMode) {
        d->queryMode = mode;
    }
}
```

8.21.4.27 QextSerialPort : :setRts (bool *set* = true) [slot]

Sets RTS line to the requested state *set* (high by default). This function will have no effect if the port associated with the class is not currently open.

Paramètres

set	

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (isOpen())
```

```
        d->setRts_sys(set);
}
```

8.21.4.28 QextSerialPort : :setStopBits (StopBitsType *stopBits*) [slot]

Sets the number of stop bits used by the serial port to *stopBits*. Possible values of stopBits are :

```
        STOP_1      1 stop bit
        STOP_1_5    1.5 stop bits
        STOP_2      2 stop bits
```

note : This function is subject to the following restrictions : 2 stop bits cannot be used with 5 data bits. 1.5 stop bits cannot be used with 6 or more data bits. POSIX does not support 1.5 stop bits.

Paramètres

StopBits-Type	

Référencé par EnttecDMXUSB : :openPort().

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (d->settings.StopBits != stopBits)
        d->setStopBits(stopBits, true);
}
```

8.21.4.29 QextSerialPort : :setTimeout (long *millisec*) [slot]

For Unix :

Sets the read and write timeouts for the port to *millisec* milliseconds. Note that this is a per-character timeout, i.e. the port will wait this long for each individual character, not for the whole read operation. This timeout also applies to the bytesWaiting() function.

note : POSIX does not support millisecond-level control for I/O timeout values. Any timeout set using this function will be set to the next lowest tenth of a second for the purposes of detecting read or write timeouts. For example a timeout of 550 milliseconds will be seen by the class as a timeout of 500 milliseconds for the purposes of reading and writing the port. However millisecond-level control is allowed by the select() system call, so for example a 550-millisecond timeout will be seen as 550 milliseconds on POSIX systems for the purpose of detecting available bytes in the read buffer.

For Windows :

Sets the read and write timeouts for the port to *millisec* milliseconds. Setting 0 indicates that timeouts are not used for read nor write operations ; however read() and write() functions will still block. Set -1 to provide non-blocking behaviour (read() and write() will return immediately).

note : this function does nothing in event driven mode.

Paramètres

long	

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    if (d->settings.Timeout_Millisec != millisec)
        d->setTimeout(millisec, true);
}
```

8.21.4.30 QextSerialPort : :stopBits ( ) const

Returns the number of stop bits used by the port. For a list of possible return values, see the definition of the enum StopBitsType.

Renvoie

StopBitsType

```
{
    QReadLocker locker(&d_func()->lock);
    return d_func()->settings.StopBits;
}
```

8.21.4.31 QextSerialPort : :writeData (const char * *data*, qint64 *maxSize*) [protected]

Writes a block of data to the serial port. This function will write len bytes from the buffer pointed to by data to the serial port. Return value is the number of bytes actually written, or -1 on error.

Avertissement

before calling this function ensure that serial port associated with this class is currently open (use isOpen() function to check if port is open).

Paramètres

data	
maxSize	

Renvoie

qint64

```
{
    Q_D(QextSerialPort);
    QWriteLocker locker(&d->lock);
    return d->writeData_sys(data, maxSize);
}
```

8.21.5 Documentation des propriétés

8.21.5.1 QextSerialPort : :portName [read, write]

Returns the name set by setPortName().

Renvoie

QString

8.21.5.2 QextSerialPort : :queryMode [read, write]

Get query mode.

Renvoie

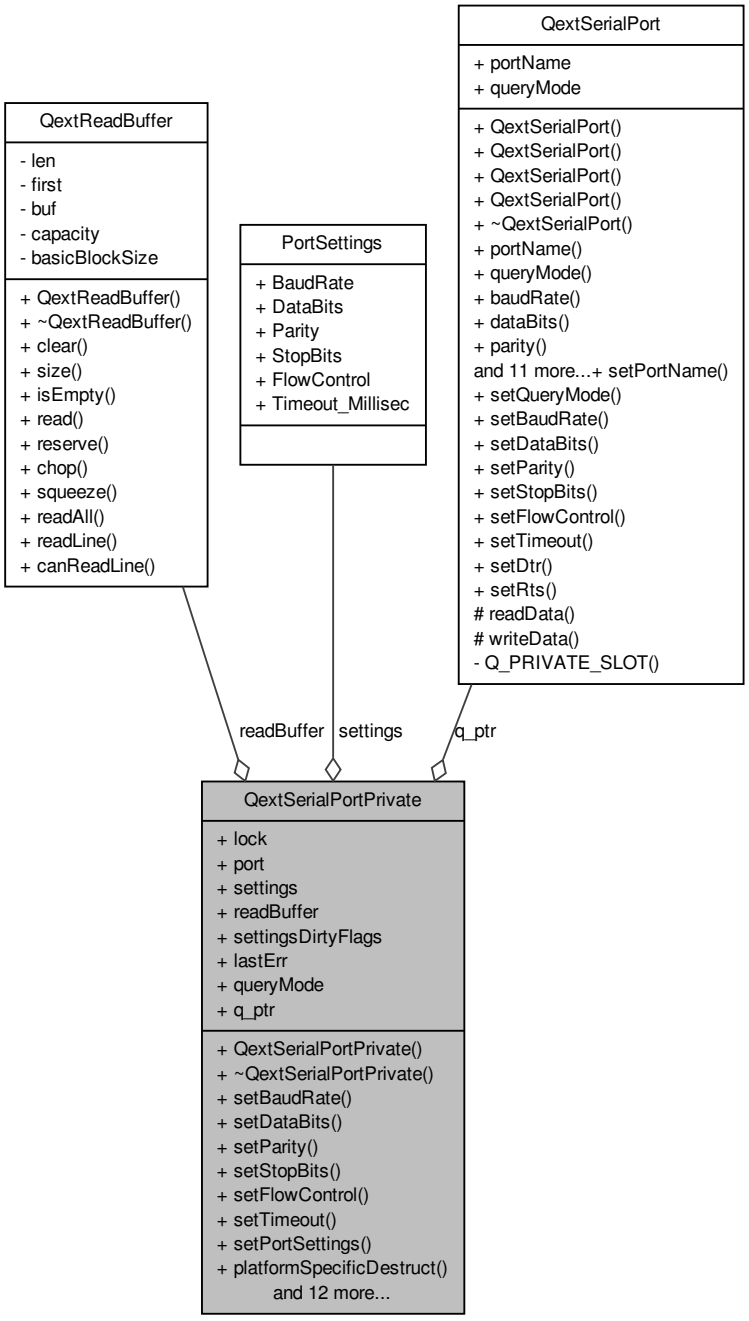
QueryMode

La documentation de cette classe a été générée à partir des fichiers suivants :
– qextserialport.h
– qextserialport.cpp

8.22 Référence de la classe QextSerialPortPrivate

```
#include <qextserialport/qextserialport_p.h>
```

Graphe de collaboration de QextSerialPortPrivate :



Types publics

- enum DirtyFlagEnum { DFE_BaudRate = 0x0001, DFE_Parity = 0x0002, DFE_StopBits = 0x0004, DFE_DataBits = 0x0008, DFE_Flow = 0x0010, DFE_TimeOut = 0x0100, DFE_ALL = 0x0fff, DFE_Settings_Mask = 0x00ff }

Fonctions membres publiques

- QextSerialPortPrivate (QextSerialPort ∗q)
- ∼QextSerialPortPrivate ()
- void setBaudRate (BaudRateType baudRate, bool update=true)
- void setDataBits (DataBitsType dataBits, bool update=true)
- void setParity (ParityType parity, bool update=true)
- void setStopBits (StopBitsType stopbits, bool update=true)
- void setFlowControl (FlowType flow, bool update=true)
- void setTimeout (long millisec, bool update=true)
- void setPortSettings (const PortSettings &settings, bool update=true)
- void platformSpecificDestruct ()
- void platformSpecificInit ()
- void translateError (ulong error)
- void updatePortSettings ()
- qint64 readData_sys (char ∗data, qint64 maxSize)
- qint64 writeData_sys (const char ∗data, qint64 maxSize)
- void setDtr_sys (bool set=true)
- void setRts_sys (bool set=true)
- bool open_sys (QIODevice : :OpenMode mode)
- bool close_sys ()
- bool flush_sys ()
- ulong lineStatus_sys ()
- qint64 bytesAvailable_sys () const
- void _q_canRead ()

Attributs publics

- QReadWriteLock lock
- QString port
- PortSettings settings
- QextReadBuffer readBuffer
- int settingsDirtyFlags
- ulong lastErr
- QextSerialPort : :QueryMode queryMode
- QextSerialPort ∗ q_ptr

8.22.1 Documentation des énumérations membres

8.22.1.1 enum QextSerialPortPrivate : :DirtyFlagEnum

Valeurs énumérées :

DFE_BaudRate

DFE_Parity

DFE_StopBits

DFE_DataBits

DFE_Flow

DFE_TimeOut

DFE_ALL

DFE_Settings_Mask

```
{
    DFE_BaudRate = 0x0001,
    DFE_Parity = 0x0002,
    DFE_StopBits = 0x0004,
    DFE_DataBits = 0x0008,
    DFE_Flow = 0x0010,
    DFE_TimeOut = 0x0100,
    DFE_ALL = 0x0fff,
    DFE_Settings_Mask = 0x00ff //without TimeOut
};
```

8.22.2 Documentation des constructeurs et destructeur

8.22.2.1 QextSerialPortPrivate : :QextSerialPortPrivate (QextSerialPort ∗ *q*)

Paramètres

q	

Références BAUD9600, PortSettings : :BaudRate, DATA_8, PortSettings : :DataBits, DFE_ALL, E_NO_ERROR, FLOW_OFF, PortSettings : :FlowControl, lastErr, PAR_NONE, PortSettings : :Parity, platformSpecificInit(), settings, settingsDirtyFlags, STOP_1, PortSettings : :StopBits, et PortSettings : :Timeout_Millisec.

```
    :lock(QReadWriteLock::Recursive), q_ptr(q)
{
    lastErr = E_NO_ERROR;
    settings.BaudRate = BAUD9600;
    settings.Parity = PAR_NONE;
    settings.FlowControl = FLOW_OFF;
    settings.DataBits = DATA_8;
    settings.StopBits = STOP_1;
    settings.Timeout_Millisec = 10;
    settingsDirtyFlags = DFE_ALL;

    platformSpecificInit();
}
```

8.22.2.2 QextSerialPortPrivate : :∼QextSerialPortPrivate ( )

Références platformSpecificDestruct().

```
{
    platformSpecificDestruct();
}
```

8.22.3 Documentation des fonctions membres

8.22.3.1 QextSerialPortPrivate : :_q_canRead ( )

Références bytesAvailable_sys(), QextReadBuffer : :chop(), readBuffer, readData_sys(), et QextReadBuffer : :reserve().

Référencé par open_sys().

```
{
    qint64 maxSize = bytesAvailable_sys();
    if (maxSize > 0) {
        char *writePtr = readBuffer.reserve(size_t(maxSize));
        qint64 bytesRead = readData_sys(writePtr, maxSize);
        if (bytesRead < maxSize)
            readBuffer.chop(maxSize - bytesRead);
        Q_Q(QextSerialPort);
        Q_EMIT q->readyRead();
    }
}
```

8.22.3.2 qint64 QextSerialPortPrivate : :bytesAvailable_sys ( ) const

Renvoie

qint64

Référencé par _q_canRead().

```
{
    int bytesQueued;
    if (::ioctl(fd, FIONREAD, &bytesQueued) == -1) {
        return (qint64)-1;
    }
    return bytesQueued;
}
```

8.22.3.3 bool QextSerialPortPrivate : :close_sys ( )

Renvoie

bool

Références flush_sys().

```
{
    // Force a flush and then restore the original termios
    flush_sys();
    // Using both TCSAFLUSH and TCSANOW here discards any pending input
    ::tcsetattr(fd, TCSAFLUSH | TCSANOW, &oldTermios);   // Restore termios
    ::close(fd);
    if (readNotifier) {
        delete readNotifier;
        readNotifier = 0;
    }
    return true;
}
```

8.22.3.4 bool QextSerialPortPrivate : :flush_sys ( )

Renvoie

bool

Référencé par close_sys().

```
{
    ::tcdrain(fd);
    return true;
}
```

8.22.3.5 ulong QextSerialPortPrivate : :lineStatus_sys (void)

Renvoie

ulong

Références LS_CTS, LS_DCD, LS_DSR, LS_DTR, LS_RI, LS_RTS, LS_SR, et LS_ST.

```
{
    unsigned long Status=0, Temp=0;
    ::ioctl(fd, TIOCMGET, &Temp);
    if (Temp & TIOCM_CTS) Status |= LS_CTS;
    if (Temp & TIOCM_DSR) Status |= LS_DSR;
    if (Temp & TIOCM_RI) Status |= LS_RI;
    if (Temp & TIOCM_CD) Status |= LS_DCD;
    if (Temp & TIOCM_DTR) Status |= LS_DTR;
    if (Temp & TIOCM_RTS) Status |= LS_RTS;
    if (Temp & TIOCM_ST) Status |= LS_ST;
    if (Temp & TIOCM_SR) Status |= LS_SR;
    return Status;
}
```

8.22.3.6 bool QextSerialPortPrivate : :open_sys (QIODevice : :OpenMode *mode*)

Paramètres

mode	

Renvoie

bool

Références _q_canRead(), DFE_ALL, QextSerialPort : :EventDriven, fullPortName(), port, queryMode, settingsDirtyFlags, translateError(), et updatePortSettings().

```
{
    Q_Q(QextSerialPort);
    //note: linux 2.6.21 seems to ignore O_NDELAY flag
    if ((fd = ::open(fullPortName(port).toLatin1() ,O_RDWR | O_NOCTTY |
      O_NDELAY)) != -1) {

        /*In the Private class, We can not call QIODevice::open()*/
        q->setOpenMode(mode);             // Flag the port as opened
        ::tcgetattr(fd, &oldTermios);    // Save the old termios
        currentTermios = oldTermios;   // Make a working copy
        ::cfmakeraw(&currentTermios);   // Enable raw access

        /*set up other port settings*/
        currentTermios.c_cflag |= CREAD|CLOCAL;
        currentTermios.c_lflag &= (~(ICANON|ECHO|ECHOE|ECHOK|ECHONL|ISIG));
        currentTermios.c_iflag &= (~(INPCK|IGNPAR|PARMRK|ISTRIP|ICRNL|IXANY));
        currentTermios.c_oflag &= (~OPOST);
        currentTermios.c_cc[VMIN] = 0;
#ifdef _POSIX_VDISABLE  // Is a disable character available on this system?

        // Some systems allow for per-device disable-characters, so get the
        //  proper value for the configured device
        const long vdisable = ::fpathconf(fd, _PC_VDISABLE);
        currentTermios.c_cc[VINTR] = vdisable;
        currentTermios.c_cc[VQUIT] = vdisable;
        currentTermios.c_cc[VSTART] = vdisable;
        currentTermios.c_cc[VSTOP] = vdisable;
        currentTermios.c_cc[VSUSP] = vdisable;
#endif //_POSIX_VDISABLE
```

```
        settingsDirtyFlags = DFE_ALL;
        updatePortSettings();

        if (queryMode == QextSerialPort::EventDriven) {
            readNotifier = new QSocketNotifier(fd, QSocketNotifier::Read, q);
            q->connect(readNotifier, SIGNAL(activated(int)), q, SLOT(_q_canRead
    ()));
        }
        return true;
    } else {
        translateError(errno);
        return false;
    }
}
```

8.22.3.7 void QextSerialPortPrivate : :platformSpecificDestruct ( )

Standard destructor.

Référencé par ~QextSerialPortPrivate().

```
{
}
```

8.22.3.8 void QextSerialPortPrivate : :platformSpecificInit ( )

Référencé par QextSerialPortPrivate().

```
{
    fd = 0;
    readNotifier = 0;
}
```

8.22.3.9 qint64 QextSerialPortPrivate : :readData_sys (char * *data*, qint64 *maxSize*)

Paramètres

data	
maxSize	

Renvoie

qint64

Reads a block of data from the serial port. This function will read at most maxSize bytes from the serial port and place them in the buffer pointed to by data. Return value is the number of bytes actually read, or -1 on error.

Avertissement

before calling this function ensure that serial port associated with this class is currently open (use isOpen() function to check if port is open).

Références E_READ_FAILED, et lastErr.

Référencé par _q_canRead().

```
{
    int retVal = ::read(fd, data, maxSize);
    if (retVal == -1)
        lastErr = E_READ_FAILED;

    return retVal;
}
```

8.22.3.10 QextSerialPortPrivate : :setBaudRate (BaudRateType *baudRate,* bool *update* = `true`)

Paramètres

baudRate	
update	

Références BAUD110, BAUD115200, BAUD1200, BAUD19200, BAUD2400, BAUD300, BAUD38400, BAUD4800, BAUD57600, BAUD600, BAUD9600, PortSettings: :BaudRate, DFE_BaudRate, QESP_PORTABILITY_WARNING, QESP_WARNING, settings, settingsDirtyFlags, et updatePortSettings().

Référencé par setPortSettings().

```
{
    switch (baudRate) {
#ifdef Q_OS_WIN

    //Windows Special
    case BAUD14400:
    case BAUD56000:
    case BAUD128000:
    case BAUD256000:
        QESP_PORTABILITY_WARNING()<<"QextSerialPort Portability Warning: POSIX
       does not support baudRate:"<<baudRate;
#elif defined(Q_OS_UNIX)

    //Unix Special
    case BAUD50:
    case BAUD75:
    case BAUD134:
    case BAUD150:
    case BAUD200:
    case BAUD1800:
#  ifdef B76800

    case BAUD76800:
#  endif

#  if defined(B230400) && defined(B4000000)

    case BAUD230400:
    case BAUD460800:
    case BAUD500000:
    case BAUD576000:
    case BAUD921600:
    case BAUD1000000:
    case BAUD1152000:
    case BAUD1500000:
    case BAUD2000000:
    case BAUD2500000:
    case BAUD3000000:
    case BAUD3500000:
    case BAUD4000000:
#  endif

        QESP_PORTABILITY_WARNING()<<"QextSerialPort Portability Warning:
       Windows does not support baudRate:"<<baudRate;
```

```
#endif

    case BAUD110:
    case BAUD300:
    case BAUD600:
    case BAUD1200:
    case BAUD2400:
    case BAUD4800:
    case BAUD9600:
    case BAUD19200:
    case BAUD38400:
    case BAUD57600:
    case BAUD115200:
#if defined(Q_OS_WIN) || defined(Q_OS_MAC)

    default:
#endif

        settings.BaudRate = baudRate;
        settingsDirtyFlags |= DFE_BaudRate;
        if (update && q_func()->isOpen())
            updatePortSettings();
        break;
#if !(defined(Q_OS_WIN) || defined(Q_OS_MAC))

    default:
        QESP_WARNING()<<"QextSerialPort does not support baudRate:"<<baudRate;
#endif

    }
}
```

8.22.3.11 QextSerialPortPrivate : :setDataBits (DataBitsType *dataBits,* bool *update =* `true`)

Paramètres

dataBits	
update	

Références DATA_5, DATA_6, DATA_7, DATA_8, PortSettings : :DataBits, DFE_DataBits, QESP_WARNING, settings, settingsDirtyFlags, STOP_2, PortSettings : :StopBits, et updatePortSettings().

Référencé par setPortSettings().

```
{
    switch(dataBits) {

    case DATA_5:
        if (settings.StopBits == STOP_2) {
            QESP_WARNING("QextSerialPort: 5 Data bits cannot be used with 2
       stop bits.");
        }
        else {
            settings.DataBits = dataBits;
            settingsDirtyFlags |= DFE_DataBits;
        }
        break;

    case DATA_6:
#ifdef Q_OS_WIN

        if (settings.StopBits == STOP_1_5) {
            QESP_WARNING("QextSerialPort: 6 Data bits cannot be used with 1.5
       stop bits.");
        }
        else
```

```
#endif

        {
            settings.DataBits = dataBits;
            settingsDirtyFlags |= DFE_DataBits;
        }
        break;

    case DATA_7:
#ifdef Q_OS_WIN

        if (settings.StopBits == STOP_1_5) {
            QESP_WARNING("QextSerialPort: 7 Data bits cannot be used with 1.5
       stop bits.");
        }
        else
#endif

        {
            settings.DataBits = dataBits;
            settingsDirtyFlags |= DFE_DataBits;
        }
        break;

    case DATA_8:
#ifdef Q_OS_WIN

        if (settings.StopBits == STOP_1_5) {
            QESP_WARNING("QextSerialPort: 8 Data bits cannot be used with 1.5
       stop bits.");
        }
        else
#endif

        {
            settings.DataBits = dataBits;
            settingsDirtyFlags |= DFE_DataBits;
        }
        break;
    default:
        QESP_WARNING()<<"QextSerialPort does not support Data bits:"<<dataBits;
    }
    if (update && q_func()->isOpen())
        updatePortSettings();
}
```

8.22.3.12 void QextSerialPortPrivate : :setDtr_sys (bool *set* = `true`)

Paramètres

set	

```
{
    int status;
    ::ioctl(fd, TIOCMGET, &status);
    if (set)
        status |= TIOCM_DTR;
    else
        status &= ~TIOCM_DTR;
    ::ioctl(fd, TIOCMSET, &status);
}
```

8.22.3.13 QextSerialPortPrivate : :setFlowControl (FlowType *flow,* bool *update* = `true`)

Paramètres

flow	
update	

Références DFE_Flow, PortSettings : :FlowControl, settings, settingsDirtyFlags, et updatePortSettings().

Référencé par setPortSettings().

```
{
    settings.FlowControl = flow;
    settingsDirtyFlags |= DFE_Flow;
    if (update && q_func()->isOpen())
        updatePortSettings();
}
```

8.22.3.14 QextSerialPortPrivate : :setParity (ParityType *parity,* bool *update* = `true`)

Paramètres

parity	
update	

Références DATA_8, PortSettings : :DataBits, DFE_Parity, PAR_EVEN, PAR_NONE, PAR_ODD, PAR_SPACE, PortSettings : :Parity, QESP_PORTABILITY_WARNING, QESP_WARNING, settings, settingsDirtyFlags, et updatePortSettings().

Référencé par setPortSettings().

```
{
    switch (parity) {
    case PAR_SPACE:
        if (settings.DataBits == DATA_8) {
#ifdef Q_OS_WIN

            QESP_PORTABILITY_WARNING("QextSerialPort Portability Warning: Space
       parity with 8 data bits is not supported by POSIX systems.");
#else

            QESP_WARNING("Space parity with 8 data bits is not supported by
       POSIX systems.");
#endif

        }
        break;

#ifdef Q_OS_WIN

        /*mark parity - WINDOWS ONLY*/
    case PAR_MARK:
        QESP_PORTABILITY_WARNING("QextSerialPort Portability Warning:  Mark
       parity is not supported by POSIX systems");
        break;
#endif


    case PAR_NONE:
    case PAR_EVEN:
    case PAR_ODD:
        break;
    default:
        QESP_WARNING()<<"QextSerialPort does not support Parity:" << parity;
    }
```

```
    settings.Parity = parity;
    settingsDirtyFlags |= DFE_Parity;
    if (update && q_func()->isOpen())
        updatePortSettings();
}
```

8.22.3.15 QextSerialPortPrivate : :setPortSettings (const PortSettings & *settings*, bool *update* = `true`)

Paramètres

settings	
update	

Références PortSettings : :BaudRate, PortSettings : :DataBits, DFE_ALL, PortSettings-: :FlowControl, PortSettings : :Parity, setBaudRate(), setDataBits(), setFlowControl(), setParity(), setStopBits(), setTimeout(), settingsDirtyFlags, PortSettings : :StopBits, -PortSettings : :Timeout_Millisec, et updatePortSettings().

```
{
    setBaudRate(settings.BaudRate, false);
    setDataBits(settings.DataBits, false);
    setStopBits(settings.StopBits, false);
    setParity(settings.Parity, false);
    setFlowControl(settings.FlowControl, false);
    setTimeout(settings.Timeout_Millisec, false);
    settingsDirtyFlags = DFE_ALL;
    if (update && q_func()->isOpen())
        updatePortSettings();
}
```

8.22.3.16 void QextSerialPortPrivate : :setRts_sys (bool *set* = `true`)

Paramètres

set	

```
{
    int status;
    ::ioctl(fd, TIOCMGET, &status);
    if (set)
        status |= TIOCM_RTS;
    else
        status &= ~TIOCM_RTS;
    ::ioctl(fd, TIOCMSET, &status);
}
```

8.22.3.17 QextSerialPortPrivate : :setStopBits (StopBitsType *stopbits*, bool *update* = `true`)

Paramètres

stopbits	
update	

Références DATA_5, PortSettings : :DataBits, DFE_StopBits, QESP_PORTABILITY-_WARNING, QESP_WARNING, settings, settingsDirtyFlags, STOP_1, STOP_2, Port-Settings : :StopBits, et updatePortSettings().

Référencé par setPortSettings().

```
{
    switch (stopBits) {

        /*one stop bit*/
    case STOP_1:
        settings.StopBits = stopBits;
        settingsDirtyFlags |= DFE_StopBits;
        break;

#ifdef Q_OS_WIN

        /*1.5 stop bits*/
    case STOP_1_5:
        QESP_PORTABILITY_WARNING("QextSerialPort Portability Warning: 1.5 stop
       bit operation is not supported by POSIX.");
        if (settings.DataBits != DATA_5) {
            QESP_WARNING("QextSerialPort: 1.5 stop bits can only be used with 5
       data bits");
        }
        else {
            settings.StopBits = stopBits;
            settingsDirtyFlags |= DFE_StopBits;
        }
        break;
#endif


        /*two stop bits*/
    case STOP_2:
        if (settings.DataBits == DATA_5) {
            QESP_WARNING("QextSerialPort: 2 stop bits cannot be used with 5
       data bits");
        }
        else {
            settings.StopBits = stopBits;
            settingsDirtyFlags |= DFE_StopBits;
        }
        break;
    default:
        QESP_WARNING()<<"QextSerialPort does not support stop bits: "<<stopBits
      ;
    }
    if (update && q_func()->isOpen())
        updatePortSettings();
}
```

8.22.3.18 QextSerialPortPrivate : :setTimeout (long *millisec,* bool *update* = `true`)

Paramètres

millisec	
update	

Références DFE_TimeOut, settings, settingsDirtyFlags, PortSettings : :Timeout_-Millisec, et updatePortSettings().

Référencé par setPortSettings().

```
{
    settings.Timeout_Millisec = millisec;
    settingsDirtyFlags |= DFE_TimeOut;
    if (update && q_func()->isOpen())
        updatePortSettings();
}
```

8.22.3.19 void QextSerialPortPrivate : :translateError (ulong *error*)

Paramètres

error	Translates a system-specific error code to a QextSerialPort error code. Used internally.

Références E_AGAIN, E_CAUGHT_NON_BLOCKED_SIGNAL, E_INVALID_FD, E_NO_MEMORY, E_PERMISSION_DENIED, et lastErr.

Référencé par open_sys().

```
{
    switch (error) {
    case EBADF:
    case ENOTTY:
        lastErr = E_INVALID_FD;
        break;
    case EINTR:
        lastErr = E_CAUGHT_NON_BLOCKED_SIGNAL;
        break;
    case ENOMEM:
        lastErr = E_NO_MEMORY;
        break;
    case EACCES:
        lastErr = E_PERMISSION_DENIED;
        break;
    case EAGAIN:
        lastErr = E_AGAIN;
        break;
    }
}
```

8.22.3.20 void QextSerialPortPrivate : :updatePortSettings ( )

Références BAUD110, BAUD115200, BAUD1200, BAUD19200, BAUD2400, BAUD300, BAUD38400, BAUD4800, BAUD57600, BAUD600, BAUD9600, PortSettings : :BaudRate, DATA_5, DATA_6, DATA_7, DATA_8, PortSettings : :DataBits, DFE_BaudRate, DFE_DataBits, DFE_Flow, DFE_Parity, DFE_Settings_Mask, DFE_StopBits, DFE_TimeOut, FLOW_HARDWARE, FLOW_OFF, FLOW_XONXOFF, PortSettings : :FlowControl, PAR_EVEN, PAR_NONE, PAR_ODD, PAR_SPACE, PortSettings : :Parity, setBaudRate2Termios(), settings, settingsDirtyFlags, STOP_1, STOP_2, PortSettings : :StopBits, et PortSettings : :Timeout_Millisec.

Référencé par open_sys(), setBaudRate(), setDataBits(), setFlowControl(), setParity(), setPortSettings(), setStopBits(), et setTimeout().

```
{
    if (!q_func()->isOpen() || !settingsDirtyFlags)
        return;

    if (settingsDirtyFlags & DFE_BaudRate) {
        switch (settings.BaudRate) {
        case BAUD50:
            setBaudRate2Termios(&currentTermios, B50);
            break;
        case BAUD75:
            setBaudRate2Termios(&currentTermios, B75);
            break;
        case BAUD110:
            setBaudRate2Termios(&currentTermios, B110);
            break;
        case BAUD134:
```

```
            setBaudRate2Termios(&currentTermios, B134);
            break;
        case BAUD150:
            setBaudRate2Termios(&currentTermios, B150);
            break;
        case BAUD200:
            setBaudRate2Termios(&currentTermios, B200);
            break;
        case BAUD300:
            setBaudRate2Termios(&currentTermios, B300);
            break;
        case BAUD600:
            setBaudRate2Termios(&currentTermios, B600);
            break;
        case BAUD1200:
            setBaudRate2Termios(&currentTermios, B1200);
            break;
        case BAUD1800:
            setBaudRate2Termios(&currentTermios, B1800);
            break;
        case BAUD2400:
            setBaudRate2Termios(&currentTermios, B2400);
            break;
        case BAUD4800:
            setBaudRate2Termios(&currentTermios, B4800);
            break;
        case BAUD9600:
            setBaudRate2Termios(&currentTermios, B9600);
            break;
        case BAUD19200:
            setBaudRate2Termios(&currentTermios, B19200);
            break;
        case BAUD38400:
            setBaudRate2Termios(&currentTermios, B38400);
            break;
        case BAUD57600:
            setBaudRate2Termios(&currentTermios, B57600);
            break;
#ifdef B76800

        case BAUD76800:
            setBaudRate2Termios(&currentTermios, B76800);
            break;
#endif

        case BAUD115200:
            setBaudRate2Termios(&currentTermios, B115200);
            break;
#if defined(B230400) && defined(B4000000)

        case BAUD230400:
            setBaudRate2Termios(&currentTermios, B230400);
            break;
        case BAUD460800:
            setBaudRate2Termios(&currentTermios, B460800);
            break;
        case BAUD500000:
            setBaudRate2Termios(&currentTermios, B500000);
            break;
        case BAUD576000:
            setBaudRate2Termios(&currentTermios, B576000);
            break;
        case BAUD921600:
            setBaudRate2Termios(&currentTermios, B921600);
            break;
        case BAUD1000000:
            setBaudRate2Termios(&currentTermios, B1000000);
            break;
        case BAUD1152000:
            setBaudRate2Termios(&currentTermios, B1152000);
            break;
        case BAUD1500000:
            setBaudRate2Termios(&currentTermios, B1500000);
            break;
```

```
        case BAUD2000000:
            setBaudRate2Termios(&currentTermios, B2000000);
            break;
        case BAUD2500000:
            setBaudRate2Termios(&currentTermios, B2500000);
            break;
        case BAUD3000000:
            setBaudRate2Termios(&currentTermios, B3000000);
            break;
        case BAUD3500000:
            setBaudRate2Termios(&currentTermios, B3500000);
            break;
        case BAUD4000000:
            setBaudRate2Termios(&currentTermios, B4000000);
            break;
#endif

#ifdef Q_OS_MAC

        default:
            setBaudRate2Termios(&currentTermios, settings.BaudRate);
            break;
#endif

        }
    }
    if (settingsDirtyFlags & DFE_Parity) {
        switch (settings.Parity) {
        case PAR_SPACE:
            /*space parity not directly supported - add an extra data bit to
       simulate it*/
            settingsDirtyFlags |= DFE_DataBits;
            break;
        case PAR_NONE:
            currentTermios.c_cflag &= (~PARENB);
            break;
        case PAR_EVEN:
            currentTermios.c_cflag &= (~PARODD);
            currentTermios.c_cflag |= PARENB;
            break;
        case PAR_ODD:
            currentTermios.c_cflag |= (PARENB|PARODD);
            break;
        }
    }
    /*must after Parity settings*/
    if (settingsDirtyFlags & DFE_DataBits) {
        if (settings.Parity != PAR_SPACE) {
            currentTermios.c_cflag &= (~CSIZE);
            switch(settings.DataBits) {
            case DATA_5:
                currentTermios.c_cflag |= CS5;
                break;
            case DATA_6:
                currentTermios.c_cflag |= CS6;
                break;
            case DATA_7:
                currentTermios.c_cflag |= CS7;
                break;
            case DATA_8:
                currentTermios.c_cflag |= CS8;
                break;
            }
        } else {
            /*space parity not directly supported - add an extra data bit to
       simulate it*/
            currentTermios.c_cflag &= ~(PARENB|CSIZE);
            switch(settings.DataBits) {
            case DATA_5:
                currentTermios.c_cflag |= CS6;
                break;
            case DATA_6:
                currentTermios.c_cflag |= CS7;
                break;
```

```
            case DATA_7:
                currentTermios.c_cflag |= CS8;
                break;
            case DATA_8:
                /*this will never happen, put here to Suppress an warning*/
                break;
            }
        }
    }
    if (settingsDirtyFlags & DFE_StopBits) {
        switch (settings.StopBits) {
        case STOP_1:
            currentTermios.c_cflag &= (~CSTOPB);
            break;
        case STOP_2:
            currentTermios.c_cflag |= CSTOPB;
            break;
        }
    }
    if (settingsDirtyFlags & DFE_Flow) {
        switch(settings.FlowControl) {
        case FLOW_OFF:
            currentTermios.c_cflag &= (~CRTSCTS);
            currentTermios.c_iflag &= (~(IXON|IXOFF|IXANY));
            break;
        case FLOW_XONXOFF:
            /*software (XON/XOFF) flow control*/
            currentTermios.c_cflag &= (~CRTSCTS);
            currentTermios.c_iflag |= (IXON|IXOFF|IXANY);
            break;
        case FLOW_HARDWARE:
            currentTermios.c_cflag |= CRTSCTS;
            currentTermios.c_iflag &= (~(IXON|IXOFF|IXANY));
            break;
        }
    }

    /*if any thing in currentTermios changed, flush*/
    if (settingsDirtyFlags & DFE_Settings_Mask)
        ::tcsetattr(fd, TCSAFLUSH, &currentTermios);

    if (settingsDirtyFlags & DFE_TimeOut) {
        int millisec = settings.Timeout_Millisec;
        if (millisec == -1) {
            ::fcntl(fd, F_SETFL, O_NDELAY);
        }
        else {
            //O_SYNC should enable blocking ::write()
            //however this seems not working on Linux 2.6.21 (works on OpenBSD
       4.2)
            ::fcntl(fd, F_SETFL, O_SYNC);
        }
        ::tcgetattr(fd, &currentTermios);
        currentTermios.c_cc[VTIME] = millisec/100;
        ::tcsetattr(fd, TCSAFLUSH, &currentTermios);
    }

    settingsDirtyFlags = 0;
}
```

8.22.3.21 qint64 QextSerialPortPrivate : :writeData_sys (const char * *data*, qint64 *maxSize*)

Paramètres

data	
maxSize	

Renvoie

qint64

Writes a block of data to the serial port. This function will write maxSize bytes from the buffer pointed to by data to the serial port. Return value is the number of bytes actually written, or -1 on error.

Avertissement

before calling this function ensure that serial port associated with this class is currently open (use isOpen() function to check if port is open).

Références E_WRITE_FAILED, et lastErr.

```
{
    int retVal = ::write(fd, data, maxSize);
    if (retVal == -1)
        lastErr = E_WRITE_FAILED;

    return (qint64)retVal;
}
```

8.22.4 Documentation des données membres

8.22.4.1 ulong QextSerialPortPrivate : :lastErr

TODO

Référencé par QextSerialPortPrivate(), readData_sys(), translateError(), et writeData_sys().

8.22.4.2 QReadWriteLock QextSerialPortPrivate : :lock [mutable]

TODO

8.22.4.3 QString QextSerialPortPrivate : :port

TODO

Référencé par open_sys().

8.22.4.4 QextSerialPort* QextSerialPortPrivate : :q_ptr

TODO

8.22.4.5 QextSerialPort : :QueryMode QextSerialPortPrivate : :queryMode

TODO

Référencé par open_sys().

8.22.4.6 QextReadBuffer QextSerialPortPrivate : :readBuffer

TODO

Référencé par _q_canRead().

8.22.4.7 PortSettings QextSerialPortPrivate : :settings

TODO

Référencé par QextSerialPortPrivate(), setBaudRate(), setDataBits(), setFlowControl(), setParity(), setStopBits(), setTimeout(), et updatePortSettings().

8.22.4.8 int QextSerialPortPrivate : :settingsDirtyFlags

TODO

Référencé par open_sys(), QextSerialPortPrivate(), setBaudRate(), setDataBits(), setFlowControl(), setParity(), setPortSettings(), setStopBits(), setTimeout(), et updatePortSettings().

La documentation de cette classe a été générée à partir des fichiers suivants :
– qextserialport_p.h
– qextserialport.cpp
– qextserialport_unix.cpp
– qextserialport_win.cpp

8.23 Référence de la structure Scene

Structure permettant de représenter le contenu d'un plan d'éclairage d'appareils XML.

`#include <scene.h>`

Attributs publics

– QString nom
– QString uuid
– QMap< int, int > valeursCanauxscene

8.23.1 Description détaillée

Structure permettant de représenter un plan d'eclairage XML implémenté dans une séquence avec une durée donnée pour ledit plan.

Auteur

Reynier Tony

Version

1.1

8.23.2 Documentation des données membres

8.23.2.1 QString Scene : :nom

Référencé par XMLUtilitaire : :enregistrerScene(), IHM : :enregistrerScene(), XMLUtilitaire : :lireScenes(), et XMLUtilitaire : :lireSceneSequence().

8.23.2.2 QString Scene : :uuid

Référencé par XMLUtilitaire : :enregistrerScene(), IHM : :enregistrerScene(), XMLUtilitaire : :lireScenes(), XMLUtilitaire : :lireSceneSequence(), XMLUtilitaire : :lireSequences(), et XMLUtilitaire : :lireSequenceSpectacle().

8.23.2.3 QMap<int,int> Scene : :valeursCanauxscene

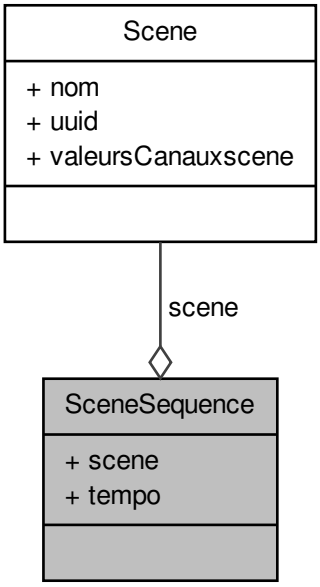
Référencé par IHM : :ajouterAppareilScene(), XMLUtilitaire : :enregistrerScene(), IHM : :enregistrerScene(), IHM : :executerScene(), XMLUtilitaire : :lireScenes(), et XMLUtilitaire : :lireSceneSequence().

La documentation de cette structure a été générée à partir du fichier suivant :
– scene.h

8.24 Référence de la structure SceneSequence

`#include <scene.h>`

Graphe de collaboration de SceneSequence :



Attributs publics

– Scene scene
– int tempo

8.24.1 Documentation des données membres

8.24.1.1 Scene SceneSequence : :scene

Référencé par IHM : :ajouterSceneSequence(), XMLUtilitaire : :lireSceneSequence(), XMLUtilitaire : :lireSequences(), et XMLUtilitaire : :lireSequenceSpectacle().

8.24.1.2 int SceneSequence : :tempo

Référencé par IHM : :ajouterSceneSequence(), XMLUtilitaire : :lireSequences(), et XMLUtilitaire : :lireSequenceSpectacle().

La documentation de cette structure a été générée à partir du fichier suivant :
– scene.h

8.25 Référence de la structure Sequence

Structure permettant de représenter le contenu d'une séquence d'enchainement de scènes temporisées.

`#include <sequence.h>`

Attributs publics

– QString nom
– QString uuid
– QVector< SceneSequence > ensembleScenes

8.25.1 Description détaillée

Auteur

Reynier Tony

Version

1.1

8.25.2 Documentation des données membres

8.25.2.1 QVector<SceneSequence> Sequence : :ensembleScenes

Référencé par IHM : :ajouterSceneSequence(), XMLUtilitaire : :enregistrerSequence(), IHM : :enregistrerSequence(), IHM : :executerSequence(), XMLUtilitaire : :lireSequences(), et XMLUtilitaire : :lireSequenceSpectacle().

8.25.2.2 QString Sequence : :nom

Référencé par IHM : :ajouterSequenceSpectacle(), XMLUtilitaire : :enregistrerSequence(), IHM : :enregistrerSequence(), XMLUtilitaire : :lireSequences(), et XMLUtilitaire : :lireSpectacles().

8.25.2.3 QString Sequence : :uuid

Référencé par IHM : :ajouterSequenceSpectacle(), XMLUtilitaire : :enregistrerSequence(), IHM : :enregistrerSequence(), XMLUtilitaire : :lireSequences(), XMLUtilitaire : :lireSequenceSpectacle(), et XMLUtilitaire : :lireSpectacles().

La documentation de cette structure a été générée à partir du fichier suivant :
– sequence.h

8.26 Référence de la structure ThreadAttente

Classe heritant de QThread et servant à génerer les temps d'attentes entre les scènes d'une séquence.

`#include <ThreadAttente.h>`

Fonctions membres publiques

– void attendre (int temps)
Méthode de génération d'un temps d'attente.

8.26.1 Description détaillée

Auteur

Reynier Tony

Version

1.1

8.26.2 Documentation des fonctions membres

8.26.2.1 void ThreadAttente : :attendre (int *temps*)

Paramètres

temps	int

Référencé par IHM : :executerSequence().

```
{
    this->sleep(temps);
}
```

La documentation de cette structure a été générée à partir des fichiers suivants :
– ThreadAttente.h
– ThreadAttente.cpp

8.27 Référence de la classe XMLUtilitaire

Classe gérant les fichiers XML de l'application.

`#include <xmlutilitaire.h>`

Fonctions membres publiques

– XMLUtilitaire (QObject *parent=0)
– ~XMLUtilitaire ()
– bool lireAppareils (QVector< DMXProjecteur * > &projecteursDMX)
Méthode recuperant tous les appareils depuis appareils.xml pour affecter leurs differentes valeurs dans le conteneur projecteursDMX.
– bool ecrireInterfaces (const QList< Interface > &interfaces)
Réecrit le fichier adaptateurs.xml et supprimant les éléments et en écrivant ceux présent dans la QList interfaces passée en paramètre.
– bool lireInterfaces (QList< Interface > &interfaces)
Lit le fichier adaptateurs.xml et remplis la QList interfaces passée en paramètre en conséquence.
– bool afficherProjecteursEnregistres (IDProjecteurSauvegarde *projecteur[])
Méthode recuperant les caracteristiques des projecteurs enregistrés dans appareils.xml pour les retranscrir sous forme de widget IDProjecteurSauvegarde affiché dans l'IHM.
– bool supprimerProjecteursEnregistres ()
Méthode de suppression de tous les projecteurs depuis appareils.xml.
– bool enregistrerAppareil (QString nom, QString type, int nbCanaux, int canalDepart, QString nomsCanaux[])
Méthode d'ajout d'un projecteur dans appareils.xml.
– bool supprimerAppareil (QString uuid)
Méthode de suppression d'un projecteur dans appareils.xml.
– bool modifierAppareil (QString uuid, QString nom, QString type, int nbCanaux, int canalDepart, QString nomsCanaux[])
Méthode de modification d'un projecteur dans appareils.xml.
– int getNbAppareils ()
Accesseur du nombres d'appareils depuis appareils.xml.
– int lireScenes (QVector< Scene > &scenesEnregistrees)
Méthode recuperant toutes les scenes depuis scenes.xml pour affecter leurs differentes valeurs dans le conteneur scenesEnregistrees.
– bool enregistrerScene (Scene scene)
Méthode d'enregistrement d'une scene dans scenes.xml.
– bool supprimerScene (QString uuid)
Méthode de suppression d'une scene dans scenes.xml.
– int lireSequences (QVector< Sequence > &sequencesEnregistrees)
Méthode de lecture des élements contenus dans sequences.xml.
– bool lireSceneSequence (SceneSequence &scene)
Méthode de lecture d'une scene requise pour une sequence dans scenes.xml.
– bool enregistrerSequence (Sequence sequence)
Méthode d'enregistrement d'une séquence dans sequences.xml.
– bool supprimerSequence (QString uuid)
Méthode de suppression d'une séquence dans sequences.xml.
– int lireSpectacles (QVector< Spectacle > &spectaclesEnregistres)
Méthode de lecture des élements contenus dans spectacles.xml.
– bool lireSequenceSpectacle (Sequence &sequence)

Méthode de lecture d'une sequence requise pour un spectacle dans sequences.xml.

- bool ecrireSpectacles (QVector< Spectacle > &spectacles)
- bool lireConsoles (QList< Console > &consoles)

  Lit le fichier consoles.xml et remplis la QList consoles passée en paramètre en conséquence.
- bool ecrireConsole (const QList< Console > &consoles)

  Réecrit le fichier consoles.xml et supprimant les éléments et en écrivant ceux présent dans la QList consoles passée en paramètre.

Fonctions membres privées

- DMXProjecteur * creerAppareil (const QDomElement &elementAppareil)

  Méthode retournant un objet herité de DMXProjecteur dont la classe dépend du QDomElement en argument.

Attributs privés

- QFile fichierAppareils
- QFile fichierSpectacles
- QFile fichierScenes
- QFile fichierSequences
- QFile fichierInterfaces
- QFile fichierConsoles

8.27.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.1

8.27.2 Documentation des constructeurs et destructeur

8.27.2.1 XMLUtilitaire : :XMLUtilitaire (QObject * *parent* = 0)

Références fichierAppareils, fichierConsoles, fichierInterfaces, fichierScenes, fichierSequences, et fichierSpectacles.

```
                                        : QObject(parent)
{
    fichierAppareils.setFileName(QCoreApplication::applicationDirPath() + "/" +
       "appareils.xml");

    fichierSpectacles.setFileName(QCoreApplication::applicationDirPath() + "/"
      + "spectacles.xml");

    fichierScenes.setFileName(QCoreApplication::applicationDirPath() + "/" + "
      scenes.xml");

    fichierSequences.setFileName(QCoreApplication::applicationDirPath() + "/" +
       "sequences.xml");

    fichierInterfaces.setFileName(QCoreApplication::applicationDirPath() + "/"
```

```
      + "adaptateurs.xml");

    fichierConsoles.setFileName(QCoreApplication::applicationDirPath() + "/" +
      "consoles.xml");
}
```

8.27.2.2 XMLUtilitaire : :∼XMLUtilitaire ( )

Références fichierAppareils, fichierConsoles, fichierInterfaces, fichierScenes, fichierSequences, et fichierSpectacles.

```
{
    if (fichierAppareils.isOpen())
    {
        fichierAppareils.close();
    }

    if (fichierSpectacles.isOpen())
    {
        fichierSpectacles.close();
    }

    if (fichierScenes.isOpen())
    {
        fichierScenes.close();
    }

    if (fichierSequences.isOpen())
    {
        fichierSequences.close();
    }

    if (fichierInterfaces.isOpen())
    {
        fichierInterfaces.close();
    }

    if (fichierConsoles.isOpen())
    {
        fichierConsoles.close();
    }

    //qDebug() << Q_FUNC_INFO;
}
```

8.27.3 Documentation des fonctions membres

8.27.3.1 bool XMLUtilitaire : :afficherProjecteursEnregistres (IDProjecteurSauvegarde ∗ *projecteur[ ]*)

Paramètres

∗*projecteur[ ]*	

Renvoie

bool

Références fichierAppareils, IDProjecteurSauvegarde : :setLabel(), et IDProjecteurSauvegarde : :setUuid().

Référencé par IHM : :creerIHM(), IHM : :enregistrerProjecteur(), et IHM : :supprimerProjecteur().

```
{
    if(!fichierAppareils.open(QIODevice::ReadOnly))
    {
        qDebug() << Q_FUNC_INFO << " Erreur ouverture " << fichierAppareils.
      fileName();
        return false;
    }
    else
    {
        //qDebug() << Q_FUNC_INFO;
        int numProjecteur = 0;
        QDomDocument documentAppareils;
        documentAppareils.setContent(&fichierAppareils, false);
        QDomElement root = documentAppareils.documentElement();

        QDomNode noeudAppareils = root.firstChild();
        while(!noeudAppareils.isNull())
        {
            QDomElement appareil = noeudAppareils.toElement();

            projecteur[numProjecteur] = new IDProjecteurSauvegarde();
            projecteur[numProjecteur]->setLabel(appareil.attribute("nom"));
            projecteur[numProjecteur]->setUuid(appareil.attribute("uuid"));

            numProjecteur++;
            noeudAppareils = noeudAppareils.nextSibling();
        }
        fichierAppareils.close();

        return true;
    }
}
```

8.27.3.2 DMXProjecteur ∗ XMLUtilitaire : :creerAppareil (const QDomElement & *elementAppareil*) [private]

Paramètres

element-Appareil	const QDomElement& un référence constante sur un élément XML représentant un appareil

Renvoie

DMXProjecteur∗ un pointeur vers un projecteur DMX

Référencé par lireAppareils().

```
{
    QString type = elementAppareil.attribute("type");

    // Lit les attributs de l'appareil
    //qDebug() << Q_FUNC_INFO << elementAppareil.attribute("nom") <<
       elementAppareil.attribute("nbCanal").toInt() << elementAppareil.attribute("type") <<
       elementAppareil.attribute("uuid");

    if(type == "LASER")
    {
        return new DMXLaser(elementAppareil.attribute("nom"), 1,
      elementAppareil.attribute("nbCanal").toInt(), elementAppareil.attribute("uuid"), type);
    }
    else if(type == "LYRE")
    {
        return new DMXLyre(elementAppareil.attribute("nom"), 1, elementAppareil
      .attribute("nbCanal").toInt(), elementAppareil.attribute("uuid"), type);
    }
    else if(type == "PAR LED")
    {
```

```
        return new DMXPAR(elementAppareil.attribute("nom"), 1, elementAppareil.
      attribute("nbCanal").toInt(), elementAppareil.attribute("uuid"), type);
    }
    else if(type == "SCANNER")
    {
        return new DMXScanner(elementAppareil.attribute("nom"), 1,
      elementAppareil.attribute("nbCanal").toInt(), elementAppareil.attribute("uuid"), type);
    }
    else if(type == "SPECIAL")
    {
        return new DMXSpecial(elementAppareil.attribute("nom"), 1,
      elementAppareil.attribute("nbCanal").toInt(), elementAppareil.attribute("uuid"), type);
    }
    else
    {
        return new DMXProjecteur(elementAppareil.attribute("nom"), 1,
      elementAppareil.attribute("nbCanal").toInt(), elementAppareil.attribute("uuid"), type);
    }
}
```

8.27.3.3 bool XMLUtilitaire : :ecrireConsole (const QList< Console > & *consoles*)

Paramètres

consoles	const QList<Console>& une référence constante sur la liste des consoles

Renvoie

bool

Références fichierConsoles.

Référencé par PlaybackWing : :ajouterConsoleListe(), PlaybackWing : :enregistrer-ModificationConsole(), et IHM : :supprimerConsole().

```
{
    if (!fichierConsoles.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierConsoles.
      fileName();
    }
    if(!(fichierConsoles.isOpen()))
    {
        QMessageBox::critical(0, QString::fromUtf8("Erreur"), QString::fromUtf8
      ("Le fichier %1 n'a pas pu être ouvert !").arg(fichierConsoles.fileName()));
        return false;
    }
    else
    {
        QDomDocument documentXML;

        fichierConsoles.resize(0);
        //documentXML.setContent(&fichierInterfaces, false);
        QDomNode xmlNode = documentXML.createProcessingInstruction("xml","
      version=\"1.0\" encoding=\"UTF-8\"");
        documentXML.insertBefore(xmlNode, documentXML.firstChild());

        if(consoles.size() > 0)
        {
            QDomElement root = documentXML.createElement("consoles");
            documentXML.appendChild(root);

            for(int i = 0; i < consoles.size(); i++)
            {
                QDomElement elementConsole = documentXML.createElement("console
      ");
```

```
                root.appendChild(elementConsole);

                QDomElement elementAdresseIP = documentXML.createElement("
    adresseIP");
                elementConsole.appendChild(elementAdresseIP);
                QDomText text = documentXML.createTextNode(consoles[i].
    adresseIP);
                elementAdresseIP.appendChild(text);

                QDomElement elementPort = documentXML.createElement("port");
                elementConsole.appendChild(elementPort);
                text = documentXML.createTextNode(QString::number(consoles[i].
    port));
                elementPort.appendChild(text);
            }
        }
        QTextStream out(&fichierConsoles);
        documentXML.save(out, 2);
    }

    fichierConsoles.close();

    return true;
}
```

8.27.3.4 bool XMLUtilitaire : :ecrireInterfaces (const QList< Interface > & *interfaces*)

Paramètres

interfaces	const QList<Interface>& une référence constante sur la liste des interfaces

Renvoie

bool

Références fichierInterfaces.

Référencé par IHM : :recupererDonneesNouvelleInterface(), IHM : :supprimerInterface(), et IHM : :~IHM().

```
{
    if (!fichierInterfaces.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierInterfaces.
      fileName();
    }
    if(!(fichierInterfaces.isOpen()))
    {
        QMessageBox::critical(0, QString::fromUtf8("Erreur"), QString::fromUtf8
      ("Le fichier %1 n'a pas pu être ouvert !").arg(fichierInterfaces.fileName()));
        return false;
    }
    else
    {
        QDomDocument documentXML;

        fichierInterfaces.resize(0);
        //documentXML.setContent(&fichierInterfaces, false);
        QDomNode xmlNode = documentXML.createProcessingInstruction("xml","
      version=\"1.0\" encoding=\"UTF-8\"");
        documentXML.insertBefore(xmlNode, documentXML.firstChild());

        if(interfaces.size() > 0)
        {
            QDomElement root = documentXML.createElement("interfaces");
            documentXML.appendChild(root);
```

```
            //qDebug() << Q_FUNC_INFO << interfaces.size() <<
        interfaces.count();
            for(int i = 0; i < interfaces.size(); i++)
            {
                QDomElement elementInterface = documentXML.createElement("
    interface");
                elementInterface.setAttribute("id", interfaces[i].id);
                elementInterface.setAttribute("utilisee", interfaces[i].
    utilisee);
                root.appendChild(elementInterface);

                QDomElement elementPeripherique = documentXML.createElement("
    port");
                elementInterface.appendChild(elementPeripherique);

                QDomText text = documentXML.createTextNode(interfaces[i].
    portInterface);
                elementPeripherique.appendChild(text);

                QDomElement elementType = documentXML.createElement("type");
                elementInterface.appendChild(elementType);

                text = documentXML.createTextNode(QString::number(interfaces[i]
    .typeInterface));
                elementType.appendChild(text);
            }
        }

        //qDebug() << documentXML.toByteArray();
        //qDebug() << fichierInterfaces.readAll();
        QTextStream out(&fichierInterfaces);
        documentXML.save(out, 2);
    }

    fichierInterfaces.close();

    return true;
}
```

8.27.3.5 bool XMLUtilitaire : :ecrireSpectacles (QVector< Spectacle > & *spectacles*)

Références fichierSpectacles.

Référencé par IHM : :enregistrerSpectacle(), et IHM : :supprimerSpectacle().

```
{
    if(!fichierSpectacles.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierSpectacles.
      fileName();
        return 0;
    }
    else
    {
        QDomDocument documentSpectacles;

        fichierSpectacles.resize(0);
        //documentXML.setContent(&fichierSpectacles, false);
        QDomNode xmlNode = documentSpectacles.createProcessingInstruction("xml"
      ,"version=\"1.0\" encoding=\"UTF-8\"");
        documentSpectacles.insertBefore(xmlNode, documentSpectacles.firstChild(
      ));
        if(spectacles.size() > 0)
        {
            QDomElement root = documentSpectacles.createElement("spectacles");
            documentSpectacles.appendChild(root);

            for(int i = 0; i < spectacles.size(); i++)
            {
               QDomElement elementSpectacle = documentSpectacles.createElement(
```

```
    "spectacle");
             root.appendChild(elementSpectacle);
             elementSpectacle.setAttribute("nom",spectacles[i].nom);
             elementSpectacle.setAttribute("uuid",spectacles[i].uuid);
             for(int j =0; j < spectacles[i].ensembleSequences.size(); j++)
             {
                 QDomElement elementSequence = documentSpectacles.
    createElement("sequence");
                 elementSpectacle.appendChild(elementSequence);
                 elementSequence.setAttribute("nom", spectacles[i].
    ensembleSequences[j].nom);
                 elementSequence.setAttribute("uuid", spectacles[i].
    ensembleSequences[j].uuid);
             }
         }

      }
      QTextStream out(&fichierSpectacles);
      fichierSpectacles.resize(0);
      documentSpectacles.save(out, 2);
  }
  fichierSpectacles.close();
  return true;

}
```

8.27.3.6 bool XMLUtilitaire : :enregistrerAppareil (QString *nom,* QString *type,* int *nbCanaux,* int *canalDepart,* QString *nomsCanaux[]*)

Paramètres

nom	QString le nom de l'appareil
type	QString le type d'appareil
nbCanaux	int le nombre de canaux pour l'appareil
canalDepart	int le numéro de canal de départ de l'appareil
noms-Canaux[]	QString un tableau contenant lesz libellés des canaux de l'appareil

Renvoie

bool

Références fichierAppareils.

Référencé par IHM : :enregistrerProjecteur().

```
{
    qDebug() << Q_FUNC_INFO;
    if(!fichierAppareils.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierAppareils.
      fileName();
        return false;
    }
    else
    {
        QUuid monUuid = QUuid::createUuid();
        QDomDocument documentAppareils;
        documentAppareils.setContent(&fichierAppareils, false);
        QDomElement root = documentAppareils.documentElement();

        QDomElement nouvelAppareil = documentAppareils.createElement("appareil"
      );
        nouvelAppareil.setAttribute("nom", nom);
        nouvelAppareil.setAttribute("nbCanaux", nbCanaux);
```

```
        nouvelAppareil.setAttribute("type", type);
        nouvelAppareil.setAttribute("uuid", monUuid.toString());

        QDomElement nouveauxCanaux[10];
        QDomText texteNomCanal;


        for(int i = 0; i < nbCanaux; i++ )
        {
            nouveauxCanaux[i] = documentAppareils.createElement("canal");
            nouveauxCanaux[i].setAttribute("id", canalDepart+i);
            texteNomCanal = documentAppareils.createTextNode(nomsCanaux[i]);
            nouveauxCanaux[i].appendChild(texteNomCanal);
            nouvelAppareil.appendChild(nouveauxCanaux[i]);
        }

        root.appendChild(nouvelAppareil);
        int nouveauNbAppareils = root.attribute("nbAppareils").toInt() + 1 ;
        root.setAttribute("nbAppareils", nouveauNbAppareils);

        QTextStream out(&fichierAppareils);
        fichierAppareils.resize(0);
        documentAppareils.save(out, 2);
        fichierAppareils.close();

        return true;
    }
}
```

8.27.3.7 bool XMLUtilitaire : :enregistrerScene (Scene *scene*)

Paramètres

scene	Scene la nouvelle scene implémentée dans le fichier XML

Renvoie

bool true si la modification a été réalisée sinon false

Références fichierScenes, Scene : :nom, Scene : :uuid, et Scene : :valeurs-Canauxscene.

Référencé par IHM : :enregistrerScene().

```
{

    if (!fichierScenes.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierScenes.
      fileName();
        return false;
    }
    else
    {
        QDomDocument documentScenes;
        documentScenes.setContent(&fichierScenes, false);
        QDomElement root = documentScenes.documentElement();
        QDomText texteNomCanal;
        QDomElement nouvelleScene = documentScenes.createElement("scene");
        nouvelleScene.setAttribute("nom", scene.nom);
        nouvelleScene.setAttribute("uuid", scene.uuid);
        QDomElement nouveauxCanaux[512];
        for(int i = 0; i < 512; i ++)
        {
            if(scene.valeursCanauxscene[i] != 0)
            {
                nouveauxCanaux[i] =  documentScenes.createElement("canal");
                nouveauxCanaux[i].setAttribute("id", i);
```

```
            texteNomCanal = documentScenes.createTextNode(QString::number(
    scene.valeursCanauxscene[i]));
            nouveauxCanaux[i].appendChild(texteNomCanal);
            nouvelleScene.appendChild(nouveauxCanaux[i]);
        }
      }
      root.appendChild(nouvelleScene);

      QTextStream out(&fichierScenes);
      fichierScenes.resize(0);
      documentScenes.save(out, 2);
      fichierScenes.close();

      return true;
    }
}
```

8.27.3.8 bool XMLUtilitaire : :enregistrerSequence (Sequence *sequence*)

Paramètres

sequence	Sequence la nouvelle séquence à implémenter dans sequences.xml

Renvoie

bool true si la modification a été réalisée sinon false

Références Sequence : :ensembleScenes, fichierSequences, Sequence : :nom, et Sequence : :uuid.

Référencé par IHM : :enregistrerSequence().

```
{

    if (!fichierSequences.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierSequences.
      fileName();
        return false;
    }
    else
    {
        QDomDocument documentSequences;
        documentSequences.setContent(&fichierSequences, false);
        QDomElement root = documentSequences.documentElement();
        QDomElement nouvelleSequence = documentSequences.createElement("
      sequence");
        nouvelleSequence.setAttribute("nom", sequence.nom);
        nouvelleSequence.setAttribute("uuid", sequence.uuid);
        qDebug() << Q_FUNC_INFO << "nombre de scenes:" << sequence.
      ensembleScenes.count();
        for(int i = 0; i < sequence.ensembleScenes.count(); i++)
        {

            QDomElement elementScene = documentSequences.createElement("scene")
      ;
            elementScene.setAttribute("uuid", sequence.ensembleScenes.at(i).
      scene.uuid);
            elementScene.setAttribute("temps", sequence.ensembleScenes.at(i).
      tempo);
            nouvelleSequence.appendChild(elementScene);
        }

        root.appendChild(nouvelleSequence);

        QTextStream out(&fichierSequences);
        fichierSequences.resize(0);
```

```
        documentSequences.save(out, 2);
        fichierSequences.close();

        return true;
    }

}
```

8.27.3.9 int XMLUtilitaire : :getNbAppareils ( )

Renvoie

int

Références fichierAppareils.

Référencé par IHM : :effacerAffichageProjecteursEnregistres(), et IHM : :mettreAJourProjecteursEnregistres().

```
{
    if (!fichierAppareils.open(QIODevice::ReadOnly))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierAppareils.
      fileName();
        return 0;
    }
    else
    {
        QDomDocument docAppareils;
        docAppareils.setContent(&fichierAppareils, false);
        QDomElement root = docAppareils.documentElement();
        //qDebug() << Q_FUNC_INFO << "nbAppareils:" <<
     root.attribute("nbAppareils") ;
        fichierAppareils.close();
        return root.attribute("nbAppareils").toInt();
    }
}
```

8.27.3.10 bool XMLUtilitaire : :lireAppareils (QVector< DMXProjecteur * > & *projecteursDMX*)

Paramètres

&projecteurs-DMX	

Renvoie

bool

Références creerAppareil(), fichierAppareils, DMXProjecteur : :setCanalDebut(), et DMXProjecteur : :setNomCanaux().

Référencé par IHM : :enregistrerProjecteur(), IHM : :IHM(), IHM : :supprimerProjecteur(), et IHM : :supprimerTousProjecteurs().

```
{
    if (!fichierAppareils.open(QIODevice::ReadOnly))
    {
```

```
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierAppareils.
    fileName();
}
else
{
    // Vider le conteneur avant de le remplir
    for(int i = 0; i < projecteursDMX.count(); i++)
    {
        delete projecteursDMX.at(i);
    }
    projecteursDMX.clear();

    QDomDocument documentXML;

    documentXML.setContent(&fichierAppareils, false);
    QDomElement racine = documentXML.documentElement(); // <appareils>
    if(racine.isNull())
    {
        qDebug() << Q_FUNC_INFO << "Erreur racine !";
        fichierAppareils.close();
        return false;
    }

    QDomNode noeudAppareil = racine.firstChild();
    if(noeudAppareil.isNull())
    {
        qDebug() << Q_FUNC_INFO << "Erreur racine vide !";
        fichierAppareils.close();
        return false;
    }

    while(!noeudAppareil.isNull())
    {
        QDomElement elementAppareil = noeudAppareil.toElement(); //
    <appareil nom="" nbCanal="" type="" uuid="">
        if(elementAppareil.isNull())
        {
            qDebug() << Q_FUNC_INFO << "Erreur element !";
            break;
        }

        DMXProjecteur* projecteurDMX = creerAppareil(elementAppareil);

        QDomNode noeudCanal = elementAppareil.firstChild();
        if(!noeudCanal.isNull())
        {
            int canalDebut = 1;
            QStringList nomCanaux;
            int i = 0;
            while(!noeudCanal.isNull())
            {
                QDomElement elementCanal = noeudCanal.toElement(); //
    <canal id="">XXX</canal>

                // Lit les attributs du canal
                //qDebug() << Q_FUNC_INFO <<
    elementCanal.attribute("id").toInt();
                if(i == 0)
                    canalDebut = elementCanal.attribute("id").toInt();
                // Lit l'élément du canal
                //qDebug() << Q_FUNC_INFO << elementCanal.text();
                nomCanaux << elementCanal.text();

                noeudCanal = noeudCanal.nextSibling();
                ++i;
            }
            projecteurDMX->setCanalDebut(canalDebut);
            projecteurDMX->setNomCanaux(nomCanaux);
        }

        projecteursDMX.push_back(projecteurDMX);
        noeudAppareil = noeudAppareil.nextSibling();
    }
}
```

```
    fichierAppareils.close();

    return true;
}
```

8.27.3.11 bool XMLUtilitaire : :lireConsoles (QList< Console > & *consoles*)

Paramètres

consoles	QList<Console>& une référence sur la liste des consoles

Renvoie

bool true si la liste a été modifiée sinon false

Références Console : :adresseIP, fichierConsoles, et Console : :port.

Référencé par PlaybackWing : :PlaybackWing().

```
{
    if (!fichierConsoles.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierConsoles.
      fileName();
    }
    if(!(fichierConsoles.isOpen()))
    {
        QMessageBox::critical(0, QString::fromUtf8("Erreur"), QString::fromUtf8
      ("Le fichier %1 n'a pas pu être ouvert !").arg(fichierConsoles.fileName()));
        return false;
    }
    else
    {
        QDomDocument documentXML;

        documentXML.setContent(&fichierConsoles, false);
        QDomElement racine = documentXML.documentElement(); // <consoles>
        if(racine.isNull())
        {
            qDebug()<< Q_FUNC_INFO << "<erreur racine !";
            fichierConsoles.close();
            return false;
        }

        QDomNode noeudConsole = racine.firstChild(); // <console>
        if(noeudConsole.isNull())
        {
            qDebug()<< Q_FUNC_INFO << "erreur racine vide !";
            fichierConsoles.close();
            return false;
        }

        while(!noeudConsole.isNull())
        {
            Console console;
            QDomElement elementConsole = noeudConsole.toElement(); // <console>
            if(elementConsole.isNull())
            {
                qDebug()<< Q_FUNC_INFO << "erreur element interface !";
                break;
            }

            QDomNode noeudAdresseIP = elementConsole.firstChild();
            QDomNode noeudPort = noeudAdresseIP.nextSibling();
            if(!noeudAdresseIP.isNull() && !noeudPort.isNull())
            {
                QDomElement elementAdresseIP = noeudAdresseIP.toElement(); //
      <adresseIP>
```

```
                QDomElement elementPort = noeudPort.toElement(); //<port>

                console.adresseIP = elementAdresseIP.text(); // l'adresse IP
                //qDebug() << console.adresseIP;
                console.port= elementPort.text().toInt(); // le port
                //qDebug() << console.port;
            }
            else qDebug()<< Q_FUNC_INFO << "erreur noeud adresseIP/port !";

            consoles.push_back(console);
            //qDebug() <<"consoles = " << consoles.count();
            noeudConsole = noeudConsole.nextSibling();
        }
    }

    fichierConsoles.close();

    return true;
}
```

8.27.3.12 bool XMLUtilitaire : :lireInterfaces (QList< Interface > & *interfaces*)

Paramètres

interfaces	QList<Interface>& une référence sur la liste des interfaces

Renvoie

bool true si la liste a été modifiée sinon false

Références fichierInterfaces, Interface : :id, Interface : :portInterface, Interface : :typeInterface, et Interface : :utilisee.

Référencé par IHM : :IHM().

```
{
    if (!fichierInterfaces.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierInterfaces.
      fileName();
    }
    if(!(fichierInterfaces.isOpen()))
    {
        QMessageBox::critical(0, QString::fromUtf8("Erreur"), QString::fromUtf8
      ("Le fichier %1 n'a pas pu être ouvert !").arg(fichierInterfaces.fileName()));
        return false;
    }
    else
    {
        QDomDocument documentXML;

        documentXML.setContent(&fichierInterfaces, false);
        QDomElement racine = documentXML.documentElement(); // <interfaces>
        if(racine.isNull())
        {
            qDebug() << "<XMLUtilitaire::lireInterfaces(QList<Interface>
       interfaces)> erreur racine !";
            fichierInterfaces.close();
            return false;
        }

        QDomNode noeudInterface = racine.firstChild(); // <interface>
        if(noeudInterface.isNull())
        {
            qDebug() << "<XMLUtilitaire::lireInterfaces(QList<Interface>
       interfaces)> erreur racine vide !";
            fichierInterfaces.close();
            return false;
```

```
        }

        while(!noeudInterface.isNull())
        {
            Interface intf;
            QDomElement elementInterface = noeudInterface.toElement(); //
       <interface>
            if(elementInterface.isNull())
            {
                qDebug() << "<XMLUtilitaire::lireInterfaces(QList<Interface>
       interfaces)> erreur element interface !";
                fichierInterfaces.close();
                break;
            }

            QDomNode noeudPeripherique = elementInterface.firstChild();
            QDomNode noeudTypeInterface = noeudPeripherique.nextSibling();
            if(!noeudPeripherique.isNull() && !noeudTypeInterface.isNull())
            {
                QDomElement elementPeripherique = noeudPeripherique.toElement()
      ; // <peripherique>
                QDomElement elementTypeInterface = noeudTypeInterface.toElement
       (); //<type>

                intf.id = elementInterface.attribute("id").toInt(); // l'id
                //qDebug() << intf.id;
                intf.utilisee = elementInterface.attribute("utilisee").toInt();
        // l'utilisation
                //qDebug() << intf.utilisee;
                intf.portInterface = elementPeripherique.text(); // le device
                //qDebug() << intf.portInterface;
                intf.typeInterface = (EnttecInterfaces)elementTypeInterface.
      text().toInt(); // le type d'interface
                //qDebug() << intf.typeInterface;
            }
            else qDebug() << "<XMLUtilitaire::lireInterfaces(QList<Interface>
        interfaces)> erreur noeud peripherique/type !";

            interfaces.push_back(intf);
            //qDebug() <<"interfaces = " << interfaces.count();
            noeudInterface = noeudInterface.nextSibling();
        }
    }

    fichierInterfaces.close();

    return true;
}
```

8.27.3.13 int XMLUtilitaire : :lireScenes (QVector< Scene > & *scenesEnregistrees*)

Paramètres

&scenes-Enregistrees	

Renvoie

int

Références fichierScenes, Scene : :nom, Scene : :uuid, et Scene : :valeurs-Canauxscene.

Référencé par IHM : :creerIHM(), IHM : :enregistrerScene(), et IHM : :supprimerScene().

```
{
    //qDebug() << Q_FUNC_INFO;
```

```
    if(!fichierScenes.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierScenes.fileName
      ();
        return 0;
    }
    else
    {
        scenesEnregistrees.clear();


        QDomDocument documentScenes;
        documentScenes.setContent(&fichierScenes, false);
        QDomElement root = documentScenes.documentElement();
        QDomNode noeudScenes = root.firstChild();
        Scene sceneAjoutee;

        while(!noeudScenes.isNull())
        {

            sceneAjoutee.nom = noeudScenes.toElement().attribute("nom");
            sceneAjoutee.uuid = noeudScenes.toElement().attribute("uuid");
            QDomNode canal = noeudScenes.firstChild();
            for( int i = 1; i <= 512; i++)
            {
               sceneAjoutee.valeursCanauxscene[i] = 0;
            }
            while(!canal.isNull())
            {
                sceneAjoutee.valeursCanauxscene[canal.toElement().attribute("id
      ").toInt()] = canal.toElement().text().toInt() ;
                canal = canal.nextSibling();
            }
            scenesEnregistrees.push_back(sceneAjoutee);
            noeudScenes = noeudScenes.nextSibling();
        }
        fichierScenes.close();

        return scenesEnregistrees.size();
    }
}
```

8.27.3.14 bool XMLUtilitaire : :lireSceneSequence (SceneSequence & *scene*)

Paramètres

&scene	SceneSequence réference à une SceneSequence

Renvoie

bool true si l'operation a été réalisée sinon false

Références fichierScenes, Scene : :nom, SceneSequence : :scene, Scene : :uuid, et Scene : :valeursCanauxscene.

Référencé par lireSequences(), et lireSequenceSpectacle().

```
{
    if (!fichierScenes.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture !" << fichierScenes.
      fileName();
        return false;
    }
    else
    {
        QDomDocument documentScenes;
        documentScenes.setContent(&fichierScenes, false);
```

```
        QDomElement root = documentScenes.documentElement();
        QDomNode noeudScenes = root.firstChild();
        while(!noeudScenes.isNull())
        {
            if(scene.scene.uuid == noeudScenes.toElement().attribute("uuid"))
            {
                scene.scene.nom = noeudScenes.toElement().attribute("nom");
                QDomNode canal = noeudScenes.firstChild();
                for( int i = 1; i <= 512; i++)
                {
                   scene.scene.valeursCanauxscene[i] = 0;
                }
                while(!canal.isNull())
                {
                    scene.scene.valeursCanauxscene[canal.toElement().attribute(
    "id").toInt()] = canal.toElement().text().toInt() ;
                    canal = canal.nextSibling();
                }
                fichierScenes.close();
                return true;

            }
            noeudScenes = noeudScenes.nextSibling();
        }
    }
    fichierScenes.close();
    return false;
}
```

8.27.3.15 int XMLUtilitaire : :lireSequences (QVector< Sequence > & *sequencesEnregistrees*)

Paramètres

&sequences-Enregistrees	QVector<Sequence> réference à un conteneur d'objets Sequence

Renvoie

int donnant le nombre de sequences recuperees

Références Sequence : :ensembleScenes, fichierSequences, lireSceneSequence(), Sequence : :nom, SceneSequence : :scene, SceneSequence : :tempo, Sequence : :uuid, et Scene : :uuid.

Référencé par IHM : :creerIHM(), IHM : :enregistrerSequence(), et IHM : :supprimerSequence().

```
{
    //qDebug() << Q_FUNC_INFO;
    if(!fichierSequences.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierSequences.
      fileName();
        return 0;
    }
    else
    {
        sequencesEnregistrees.clear();


        QDomDocument documentSequences;
        documentSequences.setContent(&fichierSequences, false);
        QDomElement root = documentSequences.documentElement();
```

```
        QDomNode noeudSequences = root.firstChild();
        Sequence sequenceAjoutee;
        SceneSequence sceneSequenceAjoutee;
        while(!noeudSequences.isNull())
        {

            sequenceAjoutee.nom = noeudSequences.toElement().attribute("nom");
            sequenceAjoutee.uuid = noeudSequences.toElement().attribute("uuid")
    ;
            QDomNode noeudScenesSequence = noeudSequences.firstChild();

            sequenceAjoutee.ensembleScenes.clear();
            while(!noeudScenesSequence.isNull())
            {

                sceneSequenceAjoutee.scene.uuid = noeudScenesSequence.toElement
    ().attribute("uuid");
                sceneSequenceAjoutee.tempo = noeudScenesSequence.toElement().
    attribute("temps").toInt();
                lireSceneSequence(sceneSequenceAjoutee);
                sequenceAjoutee.ensembleScenes.append(sceneSequenceAjoutee);
                noeudScenesSequence = noeudScenesSequence.nextSibling();
            }

            sequencesEnregistrees.push_back(sequenceAjoutee);
            noeudSequences = noeudSequences.nextSibling();
        }
        fichierSequences.close();

        return sequencesEnregistrees.size();
    }
}
```

8.27.3.16 XMLUtilitaire : :lireSequenceSpectacle (Sequence & *sequence*)

Méthode d'écriture d'une sequence dans sequences.xml.

Paramètres

&sequence	Sequence réference à une Sequence

Renvoie

bool true si l'operation a été réalisée sinon false

Paramètres

&sequences	QVector<Spectacle> réference à un conteneur de Spectacles

Renvoie

bool true si l'operation a été réalisée sinon false

Références Sequence : :ensembleScenes, fichierSequences, lireSceneSequence(), SceneSequence : :scene, SceneSequence : :tempo, Sequence : :uuid, et Scene : :uuid.

Référencé par lireSpectacles().

```
{
    qDebug() << Q_FUNC_INFO;
    if(!fichierSequences.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierSequences.
    fileName();
```

```
        return false;
    }
    else
    {
        QDomDocument documentSequences;
        documentSequences.setContent(&fichierSequences, false);
        QDomElement root = documentSequences.documentElement();
        QDomNode noeudSequences = root.firstChild();

        while(!noeudSequences.isNull())
        {
            if(sequence.uuid == noeudSequences.toElement().attribute("uuid"))
            {
                QDomNode noeudScenes = noeudSequences.toElement().firstChild();
                SceneSequence scene;
                while(!noeudScenes.isNull())
                {
                    scene.tempo = noeudScenes.toElement().attribute("temps").
      toInt();
                    scene.scene.uuid = noeudScenes.toElement().attribute("uuid"
      );
                    lireSceneSequence(scene);
                    sequence.ensembleScenes.append(scene);
                    noeudScenes = noeudScenes.nextSibling();
                }
                fichierSequences.close();
                return true;
            }
            noeudSequences = noeudSequences.nextSibling();
        }
        fichierSequences.close();

        return false;

    }
}
```

8.27.3.17 int XMLUtilitaire : :lireSpectacles (QVector< Spectacle > & *spectaclesEnregistres*)

Paramètres

&spectacles-Enregistres	QVector<Spectacle> réference à un conteneur d'objets Spectacle

Renvoie

int donnant le nombre de spectacles recuperees

Références fichierSpectacles, lireSequenceSpectacle(), Sequence : :nom, et Sequence : :uuid.

Référencé par IHM : :creerIHM(), et IHM : :enregistrerSpectacle().

```
{
    if(!fichierSpectacles.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierSpectacles.
      fileName();
        return 0;
    }
    else
    {
        spectaclesEnregistres.clear();
        QDomDocument documentSpectacles;
        documentSpectacles.setContent(&fichierSpectacles, false);
        QDomElement root = documentSpectacles.documentElement();
```

```
        QDomNode noeudSpectacles = root.firstChild();
        Spectacle spectacle;

        while(!noeudSpectacles.isNull())
        {
            QDomElement elementSpectacle = noeudSpectacles.toElement();
            spectacle.nom = elementSpectacle.attribute("nom");
            spectacle.uuid = elementSpectacle.attribute("uuid");

            QDomNode noeudSequences = elementSpectacle.firstChild();
            Sequence sequence;
            while(!noeudSequences.isNull())
            {
                sequence.nom = noeudSequences.toElement().attribute("nom");
                sequence.uuid = noeudSequences.toElement().attribute("uuid");
                lireSequenceSpectacle(sequence);
                spectacle.ensembleSequences.append(sequence);
                noeudSequences = noeudSequences.nextSibling();
            }


            spectaclesEnregistres.append(spectacle);
            noeudSpectacles = noeudSpectacles.nextSibling();
        }

        fichierSpectacles.close();
        return spectaclesEnregistres.size();
    }
}
```

8.27.3.18 bool XMLUtilitaire : :modifierAppareil (QString *uuid,* QString *nom,* QString *type,* int *nbCanaux,* int *canalDepart,* QString *nomsCanaux[]*)

Paramètres

uuid	QString l'uuid de l'appareil
nom	QString le nom de l'appareil
type	QString le type de l'appareil
nbCanaux	int le nombre de canaux de l'appareil
canalDepart	int le numéro de canal de départ de l'appareil
nomsCanaux[]	QString un tableau contenant les libellés des canaux de l'appareil

Renvoie

bool true si la modification a été réalisée sinon false

Références fichierAppareils.

Référencé par IHM : :enregistrerProjecteur().

```
{

    if(!fichierAppareils.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierAppareils.
      fileName();
        return false;
    }
    else
    {

        QDomDocument documentAppareils;
        documentAppareils.setContent(&fichierAppareils, false);
```

```
        QDomElement root = documentAppareils.documentElement();
        QDomNode noeudAppareils = root.firstChild();
        while(!noeudAppareils.isNull())
        {

            QDomElement appareil = noeudAppareils.toElement();
            QDomText texteNomCanal;
            QDomElement nouveauxCanaux[10];
            if(appareil.attribute("uuid") == uuid)
            {

                appareil.setAttribute("nom", nom);
                appareil.setAttribute("nbCanaux", nbCanaux);
                appareil.setAttribute("type", type);

                QDomNode noeudCanaux = appareil.firstChild();

                while(!noeudCanaux.isNull())
                {

                    if(!noeudCanaux.nextSibling().isNull())
                    {

                        noeudCanaux = noeudCanaux.nextSibling();
                        appareil.removeChild(noeudCanaux.previousSibling());
                    }
                    else
                    {

                        appareil.removeChild(noeudCanaux);
                        noeudCanaux = noeudCanaux.nextSibling();
                    }
                }

                for(int i = 0; i < nbCanaux; i++ )
                {
                    nouveauxCanaux[i] = documentAppareils.createElement("canal"
    );

                    nouveauxCanaux[i].setAttribute("id", canalDepart+i);
                    texteNomCanal = documentAppareils.createTextNode(nomsCanaux
    [i]);
                    nouveauxCanaux[i].appendChild(texteNomCanal);
                    appareil.appendChild(nouveauxCanaux[i]);
                }


                QTextStream out(&fichierAppareils);
                fichierAppareils.resize(0);
                documentAppareils.save(out, 2);
                fichierAppareils.close();
                return true;
            }
            noeudAppareils = noeudAppareils.nextSibling();
        }
        qDebug() << Q_FUNC_INFO << "Erreur modification !" ;
        fichierAppareils.close();

        return false;
    }
}
```

8.27.3.19 bool XMLUtilitaire : :supprimerAppareil (QString *uuid*)

Paramètres

uuid	

Renvoie

bool

Références fichierAppareils.

Référencé par IHM : :supprimerProjecteur().

```
{
    qDebug() << Q_FUNC_INFO;
    if(!fichierAppareils.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierAppareils.
      fileName();
        return false;
    }
    else
    {
        QDomDocument documentAppareils;
        documentAppareils.setContent(&fichierAppareils, false);
        QDomElement root = documentAppareils.documentElement();
        QDomNode noeudAppareils = root.firstChild();
        while(!noeudAppareils.isNull())
        {
            QDomElement appareil = noeudAppareils.toElement();

            if(appareil.attribute("uuid") == uuid)
            {
                root.removeChild(noeudAppareils);
                int nouveauNbAppareils = root.attribute("nbAppareils").toInt()
      - 1 ;
                root.setAttribute("nbAppareils", nouveauNbAppareils);
                QTextStream out(&fichierAppareils);
                fichierAppareils.resize(0);
                documentAppareils.save(out, 2);
                fichierAppareils.close();
                return true;
            }
            noeudAppareils = noeudAppareils.nextSibling();
        }
        qDebug() << Q_FUNC_INFO << "Erreur supression !" ;
        fichierAppareils.close();

        return false;
    }
}
```

8.27.3.20 bool XMLUtilitaire : :supprimerProjecteursEnregistres ( )

Renvoie

bool

Références fichierAppareils.

Référencé par IHM : :supprimerTousProjecteurs().

```
{
    if(!fichierAppareils.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierAppareils.
      fileName();
        return false;
    }
    else
    {
        QDomDocument documentAppareils;
        documentAppareils.setContent(&fichierAppareils, false);
```

```
        QDomElement root = documentAppareils.documentElement();

        QDomNode noeudAppareils = root.firstChild();

        while(!noeudAppareils.isNull())
        {

            QDomElement appareil = noeudAppareils.toElement();
            qDebug() << "Suppression" << appareil.attribute("nom");

            if(noeudAppareils.nextSibling().isNull())
            {
                root.removeChild(noeudAppareils);
                noeudAppareils = noeudAppareils.nextSibling();
            }
            else
            {
                noeudAppareils = noeudAppareils.nextSibling();
                root.removeChild(noeudAppareils.previousSibling());
            }
        }
        root.setAttribute("nbAppareils", 0);
        fichierAppareils.resize(0);
        fichierAppareils.write(documentAppareils.toByteArray());
        fichierAppareils.close();

        return true;
    }

}
```

8.27.3.21 bool XMLUtilitaire : :supprimerScene (QString *uuid*)

Paramètres

uuid	QString l'uuid de la scène à supprimer

Renvoie

bool true si la modification a été réalisée sinon false

Références fichierScenes.

Référencé par IHM : :supprimerScene().

```
{
    qDebug() << Q_FUNC_INFO;
    if(!fichierScenes.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierScenes.fileName
      ();
        return false;
    }
    else
    {
        QDomDocument documentScenes;
        documentScenes.setContent(&fichierScenes, false);
        QDomElement root = documentScenes.documentElement();
        QDomNode noeudScenes = root.firstChild();
        while(!noeudScenes.isNull())
        {
            QDomElement scene = noeudScenes.toElement();
            qDebug() << "uuid XML:" << scene.attribute("uuid") ;
            qDebug() << "uuid argument:" << uuid ;
            if(scene.attribute("uuid") == uuid)
            {
                root.removeChild(noeudScenes);
                QTextStream out(&fichierScenes);
```

```
                fichierScenes.resize(0);
                documentScenes.save(out, 2);
                fichierScenes.close();
                return true;
            }
            noeudScenes = noeudScenes.nextSibling();
        }
        qDebug() << Q_FUNC_INFO << "Erreur supression !" ;
        fichierScenes.close();

        return false;
    }
}
```

8.27.3.22 bool XMLUtilitaire : :supprimerSequence (QString *uuid*)

Paramètres

uuid	QString uuid de la séquence concernée

Renvoie

bool true si la modification a été réalisée sinon false

Références fichierSequences.

Référencé par IHM : :supprimerSequence().

```
{
    qDebug() << Q_FUNC_INFO;
    if(!fichierSequences.open(QIODevice::ReadWrite))
    {
        qDebug() << Q_FUNC_INFO << "Erreur ouverture" << fichierSequences.
      fileName();
        return false;
    }
    else
    {
        QDomDocument documentSequences;
        documentSequences.setContent(&fichierSequences, false);
        QDomElement root = documentSequences.documentElement();
        QDomNode noeudSequences = root.firstChild();
        while(!noeudSequences.isNull())
        {
            QDomElement sequence = noeudSequences.toElement();
            qDebug() << "uuid XML:" << sequence.attribute("uuid") ;
            qDebug() << "uuid argument:" << uuid ;
            if(sequence.attribute("uuid") == uuid)
            {
                root.removeChild(noeudSequences);
                QTextStream out(&fichierSequences);
                fichierSequences.resize(0);
                documentSequences.save(out, 2);
                fichierSequences.close();
                return true;
            }
            noeudSequences = noeudSequences.nextSibling();
        }
        qDebug() << Q_FUNC_INFO << "Erreur supression !" ;
        fichierSequences.close();

        return false;
    }
}
```

8.27.4 Documentation des données membres

8.27.4.1 QFile XMLUtilitaire : :fichierAppareils [private]

Référencé par afficherProjecteursEnregistres(), enregistrerAppareil(), getNbAppareils(), lireAppareils(), modifierAppareil(), supprimerAppareil(), supprimerProjecteursEnregistres(), XMLUtilitaire(), et ∼XMLUtilitaire().

8.27.4.2 QFile XMLUtilitaire : :fichierConsoles [private]

Référencé par ecrireConsole(), lireConsoles(), XMLUtilitaire(), et ∼XMLUtilitaire().

8.27.4.3 QFile XMLUtilitaire : :fichierInterfaces [private]

Référencé par ecrireInterfaces(), lireInterfaces(), XMLUtilitaire(), et ∼XMLUtilitaire().

8.27.4.4 QFile XMLUtilitaire : :fichierScenes [private]

Référencé par enregistrerScene(), lireScenes(), lireSceneSequence(), supprimerScene(), XMLUtilitaire(), et ∼XMLUtilitaire().

8.27.4.5 QFile XMLUtilitaire : :fichierSequences [private]

Référencé par enregistrerSequence(), lireSequences(), lireSequenceSpectacle(), supprimerSequence(), XMLUtilitaire(), et ∼XMLUtilitaire().

8.27.4.6 QFile XMLUtilitaire : :fichierSpectacles [private]

Référencé par ecrireSpectacles(), lireSpectacles(), XMLUtilitaire(), et ∼XMLUtilitaire().

La documentation de cette classe a été générée à partir des fichiers suivants :
– xmlutilitaire.h
– xmlutilitaire.cpp

9 Documentation des fichiers

9.1 Référence du fichier Changelog.dox

9.2 Référence du fichier console.h

```
#include <QString>
```

Classes

– struct Console
Structure comprenant les diférents paramètre d'une interface.

9.2.1 Description détaillée

9.3 Référence du fichier DMXLaser.cpp

Definition de la classe DMXLaser.

```
#include "DMXLaser.h" #include <QDebug>
```

9.3.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.4 Référence du fichier DMXLaser.h

```
#include "DMXProjecteur.h"
```

Classes

– class DMXLaser

9.5 Référence du fichier DMXLyre.cpp

Definition de la classe DMXLyre.

```
#include "DMXLyre.h" #include <QDebug>
```

9.5.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.6 Référence du fichier DMXLyre.h

```
#include "DMXProjecteur.h"
```

Classes

– class DMXLyre

9.7 Référence du fichier DMXPAR.cpp

Definition de la classe DMXPAR.

```
#include "DMXPAR.h" #include <QDebug>
```

9.7.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.8 Référence du fichier DMXPAR.h

```
#include "DMXProjecteur.h"
```

Classes

– class DMXPAR

9.9 Référence du fichier DMXProjecteur.cpp

Definition de la classe DMXProjecteur.

```
#include "DMXProjecteur.h" #include <QDebug>
```

9.9.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.10 Référence du fichier DMXProjecteur.h

```
#include <QtCore>
```

Classes

– class DMXProjecteur
Definition de la classe DMXProjecteur.

9.11 Référence du fichier DMXScanner.cpp

Definition de la classe DMXScanner.

```
#include "DMXScanner.h" #include <QDebug>
```

9.11.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.12 Référence du fichier DMXScanner.h

```
#include "DMXProjecteur.h"
```

Classes

– class DMXScanner

9.13 Référence du fichier DMXSpecial.cpp

Definition de la classe DMXSpecial.

```
#include "DMXSpecial.h" #include <QDebug>
```

9.13.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.14 Référence du fichier DMXSpecial.h

```
#include "DMXProjecteur.h"
```

Classes

– class DMXSpecial

9.15 Référence du fichier enttecdmxusb.cpp

Definition de la classe EnttecDMXUSB.

```
#include "enttecdmxusb.h" #include <QDebug>
```

9.15.1 Description détaillée

Auteur

Thierry Vaira

Version

1.0

9.16 Référence du fichier enttecdmxusb.h

Declaration de la classe EnttecDMXUSB.

```
#include <iostream> #include <iomanip> #include <string> ×
#include <stdio.h> #include <unistd.h> #include <stdlib.-
h> #include <math.h> #include <string.h> #include "qextserialport.-
h"
```

Classes

- class EnttecDMXUSB
  Classe permettant la communication avec l'interface Enttec.

Macros

- #define NB_CANAUX_MAX 512
- #define NB_INTERFACES 2
- #define DEBUG_DMX_USB

Définitions de type

- typedef unsigned char byte
- typedef byte TDmxArray [NB_CANAUX_MAX+1]

Énumérations

- enum EnttecInterfaces { OPEN_DMX_USB, DMX_USB_PRO }

Variables

- static const char nomInterfaces [][20]
- const int PKT_SOM = 0x7E
- const int PKT_EOM = 0xE7

- const int PKT_GETCFG = 3
- const int PKT_SETCFG = 4
- const int PKT_GETSERIAL = 10
- const int PKT_DMXOUT = 6
- const int PKT_DMXIN = 5
- const int PKT_DMX_RDM = 7
- const int PKT_DMXIN_MODE = 8
- const int PKT_DMXIN_UPDATE = 9
- const int USER_DATA_SIZE = 508
- const int D_TIMEOUT = 100

9.16.1 Description détaillée

Auteur

Thierry Vaira

Version

1.0

9.16.2 Documentation des macros

9.16.2.1 #define DEBUG_DMX_USB

9.16.2.2 #define NB_CANAUX_MAX 512

Référencé par EnttecDMXUSB : :EnttecDMXUSB(), EnttecDMXUSB : :openPort(), EnttecDMXUSB : :ResetCanauxDMX(), EnttecDMXUSB : :SendDatasDMX(), EnttecDMXUSB : :SetCanalDMX(), et EnttecDMXUSB : :SetNbCanauxDMX().

9.16.2.3 #define NB_INTERFACES 2

9.16.3 Documentation des définitions de type

9.16.3.1 typedef unsigned char byte

9.16.3.2 typedef byte TDmxArray[NB_CANAUX_MAX+1]

9.16.4 Documentation du type de l'énumération

9.16.4.1 enum EnttecInterfaces

Valeurs énumérées :

OPEN_DMX_USB

DMX_USB_PRO

```
{
    OPEN_DMX_USB, // 0
    DMX_USB_PRO // 1
};
```

9.16.5 Documentation des variables

9.16.5.1 const int D_TIMEOUT = 100

9.16.5.2 const char nomInterfaces[][20] [static]

Valeur initiale :

```
{
   "Enttec OPEN DMX USB",
   "Enttec DMX USB PRO"
}
```

Référencé par EnttecDMXUSB : :GetNomInterface(), et EnttecDMXUSB : :sendPacket().

9.16.5.3 const int PKT_DMX_RDM = 7

9.16.5.4 const int PKT_DMXIN = 5

Référencé par EnttecDMXUSB : :openPort(), EnttecDMXUSB : :recieve(), EnttecDMXUSB : :widgetRecieveAllMode(), et EnttecDMXUSB : :widgetRecieveOnChangeMode().

9.16.5.5 const int PKT_DMXIN_MODE = 8

Référencé par EnttecDMXUSB : :widgetRecieveAllMode(), et EnttecDMXUSB : :widgetRecieveOnChangeMode().

9.16.5.6 const int PKT_DMXIN_UPDATE = 9

Référencé par EnttecDMXUSB : :recieve().

9.16.5.7 const int PKT_DMXOUT = 6

Référencé par EnttecDMXUSB : :SendDatasDMX(), et EnttecDMXUSB : :SendDMX().

9.16.5.8 const int PKT_EOM = 0xE7

Référencé par EnttecDMXUSB : :recieve(), et EnttecDMXUSB : :sendPacket().

9.16.5.9 const int PKT_GETCFG = 3

Référencé par EnttecDMXUSB : :recieve(), et EnttecDMXUSB : :widgetRequestConfig().

9.16.5.10 const int PKT_GETSERIAL = 10

Référencé par EnttecDMXUSB : :recieve(), et EnttecDMXUSB : :widgetRequestSerial().

9.16.5.11 const int PKT_SETCFG = 4

9.16.5.12 const int PKT_SOM = 0x7E

Référencé par EnttecDMXUSB : :recieve(), et EnttecDMXUSB : :sendPacket().

9.16.5.13 const int USER_DATA_SIZE = 508

9.17 Référence du fichier errcode.h

Déclaration des constantes d'erreur pour la classe CRS232.

Macros

- #define OK 0
- #define BADPARAM -2
- #define FILENOTFOUND -3
- #define CREATEERROR -4
- #define BADFILETYPE -5
- #define READERROR -6
- #define WRITEERROR -7
- #define NOTHINGTOSAVE -8
- #define NOTSUPPORTED -9
- #define E2P_TIMEOUT -10
- #define IICERR_BUSBUSY -11
- #define IICERR_NOTACK -12
- #define IICERR_NOADDRACK -13
- #define IICERR_SDACONFLICT -14
- #define IICERR_SCLCONFLICT -15
- #define E2ERR_OPENFAILED -16
- #define E2ERR_ACCESSDENIED -17
- #define E2ERR_NOTINSTALLED -18
- #define IICERR_TIMEOUT -19
- #define IICERR_STOP -20
- #define E2ERR_WRITEFAILED -21
- #define DEVICE_BADTYPE -23
- #define DEVICE_UNKNOWN -24
- #define DEVICE_LOCKED -25
- #define OP_ABORTED -26
- #define BUFFEROVERFLOW -30
- #define OUTOFMEMORY -31
- #define BUFFERUNDERFLOW -32
- #define CMD_BUFFEREMPTY -39
- #define CMD_NOTHINGTOWRITE -40
- #define CMD_NOTHINGTOVERIFY -41
- #define CMD_NOTHINGTOLOAD -42
- #define CMD_NOTHINGTOSAVE -43
- #define CMD_WRITEFAILED -44
- #define CMD_VERIFYFAILED -45
- #define CMD_ROLLOVERFAILED -46

9.17.1 Description détaillée

Auteur

Thierry Vaira

Version

0.9

9.17.2 Documentation des macros

9.17.2.1 #define BADFILETYPE -5

9.17.2.2 #define BADPARAM -2

9.17.2.3 #define BUFFEROVERFLOW -30

9.17.2.4 #define BUFFERUNDERFLOW -32

9.17.2.5 #define CMD_BUFFEREMPTY -39

9.17.2.6 #define CMD_NOTHINGTOLOAD -42

9.17.2.7 #define CMD_NOTHINGTOSAVE -43

9.17.2.8 #define CMD_NOTHINGTOVERIFY -41

9.17.2.9 #define CMD_NOTHINGTOWRITE -40

9.17.2.10 #define CMD_ROLLOVERFAILED -46

9.17.2.11 #define CMD_VERIFYFAILED -45

9.17.2.12 #define CMD_WRITEFAILED -44

9.17.2.13 #define CREATEERROR -4

9.17.2.14 #define DEVICE_BADTYPE -23

9.17.2.15 #define DEVICE_LOCKED -25

9.17.2.16 #define DEVICE_UNKNOWN -24

9.17.2.17 #define E2ERR_ACCESSDENIED -17

9.17.2.18 #define E2ERR_NOTINSTALLED -18

9.17.2.19 #define E2ERR_OPENFAILED -16

9.17.2.20 #define E2ERR_WRITEFAILED -21

9.17.2.21 #define E2P_TIMEOUT -10

9.17.2.22 #define FILENOTFOUND -3

9.17.2.23 #define IICERR_BUSBUSY -11

9.17.2.24 #define IICERR_NOADDRACK -13

9.17.2.25 #define IICERR_NOTACK -12

9.17.2.26 #define IICERR_SCLCONFLICT -15

9.17.2.27 #define IICERR_SDACONFLICT -14

9.17.2.28 #define IICERR_STOP -20

9.17.2.29 #define IICERR_TIMEOUT -19

9.17.2.30 #define NOTHINGTOSAVE -8

9.17.2.31 #define NOTSUPPORTED -9

9.17.2.32 #define OK 0

9.17.2.33 #define OP_ABORTED -26

9.17.2.34 #define OUTOFMEMORY -31

9.17.2.35 #define READERROR -6

9.17.2.36 #define WRITEERROR -7

9.18 Référence du fichier idProjecteurSauvegarde.cpp

Définition de la classe PlaybackWing.

```
#include "idProjecteurSauvegarde.h" #include <QDebug>
```

9.18.1 Description détaillée

Auteur

Reynier Tony

Version

1.0

9.19 Référence du fichier idProjecteurSauvegarde.h

```
#include <QtGlobal> #include <QtGui>
```

Classes

– class IDProjecteurSauvegarde

9.20 Référence du fichier IHM.cpp

```
#include "IHM.h"  #include "enttecdmxusb.h"  #include "D-
MXProjecteur.h"  #include "xmlutilitaire.h"  #include "-
PlaybackWing.h" #include <QDebug>
```

9.20.1 Description détaillée

9.21 Référence du fichier IHM.h

Déclaration de la classe IHM.

```
#include <QtGlobal> #include <QtGui> #include <QTimer>×
#include "interface.h" #include "PlaybackWing.h" #include
"myslider.h" #include "idProjecteurSauvegarde.h" #include
"scene.h"  #include "sequence.h"  #include "spectacle.h"×
#include "ThreadAttente.h"
```

Classes

– class IHM
Définition de la classe IHM.

Macros

– #define NB_TAB_MAIN 4
– #define NB_SLIDERS 10
– #define NB_CANAUX 512
– #define NB_PROJECTEURS 32

9.21.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.1.2

9.21.2 Documentation des macros

9.21.2.1 #define NB_CANAUX 512

Référencé par IHM : :changerCanalDepart(), IHM : :decrementerCanal(), IHM-: :incrementerCanal(), MySlider : :MySlider(), et IHM : :setCanal().

9.21.2.2 #define NB_PROJECTEURS 32

9.21.2.3 #define NB_SLIDERS 10

Référencé par IHM : :activerPilotageIHM(), IHM : :ajouterAppareilScene(), IHM : :creer-IHM(), IHM : :decrementerCanal(), IHM : :desactiverPilotageIHM(), IHM : :generer-FenetreCanauxScene(), IHM : :IHM(), IHM : :incrementerCanal(), IHM : :selectionner-ProjecteursPilotage(), et IHM : :setCanal().

9.21.2.4 #define NB_TAB_MAIN 4

9.22 Référence du fichier interface.h

`#include <QString> #include "enttecdmxusb.h"`

Classes

– struct Interface
Structure comprenant les différents paramètre d'une interface USB/DMX.

9.22.1 Description détaillée

9.23 Référence du fichier main.cpp

Programme principal.

`#include <QApplication> #include "IHM.h"`

Fonctions

– int main (int argc, char ∗argv[])
Programme principal.

9.23.1 Description détaillée

Crée et affiche la fenêtre principale de l'application

9.23.2 Documentation des fonctions

9.23.2.1 main (int *argc*, char ∗ *argv[]*)

Paramètres

argc	
argv[]	

Renvoie

int

```
{
    //QTextCodec::setCodecForCStrings(QTextCodec::codecForName("UTF-8"));
    //QTextCodec::setCodecForTr(QTextCodec::codecForName("UTF-8"));
    //QTextCodec::setCodecForLocale(QTextCodec::codecForName("ISO 8859-1"));
    //QTextCodec::setCodecForLocale(QTextCodec::codecForName("Windows-1250"));

        QApplication a(argc, argv);
    IHM w;
    w.show();
```

```
    return a.exec();
}
```

9.24 Référence du fichier myslider.cpp

Définition de la classe myslider permettant d'instancier des widgets de type slider personnalisés.

`#include "myslider.h" #include <QDebug>`

9.24.1 Description détaillée

Version

1.0

9.25 Référence du fichier myslider.h

`#include <QtGlobal> #include <QtGui>`

Classes

– class MySlider
Déclaration de la classe myslider permettant d'instancier des widgets de type slider personnalisés.

Macros

– #define VALEUR_MIN 0
– #define VALEUR_MAX 255
– #define NB_CANAUX 512
– #define NB_DIGIT 3

9.25.1 Description détaillée

9.25.2 Documentation des macros

9.25.2.1 #define NB_CANAUX 512

9.25.2.2 #define NB_DIGIT 3

Référencé par MySlider : :MySlider().

9.25.2.3 #define VALEUR_MAX 255

Référencé par MySlider : :MySlider().

9.25.2.4 #define VALEUR_MIN 0

Référencé par MySlider : :MySlider().

9.26 Référence du fichier PlaybackWing.cpp

Définition de la classe PlaybackWing.

```
#include "PlaybackWing.h"  #include "console.h"  #include
"xmlutilitaire.h" #include <bitset> #include <QUdpSocket> ×
#include <QByteArray>  #include <QBitArray>  #include <-
QHostAddress>  #include <QMessageBox>  #include <QChar> ×
#include <QDebug>
```

9.26.1 Description détaillée

Auteur

Demont Thomas

Version

1.0

9.27 Référence du fichier PlaybackWing.h

Déclaration de la classe PlaybackWing.

```
#include "xmlutilitaire.h"  #include "console.h"  #include
<QUdpSocket>
```

Classes

- struct EtatTouchesControle
  Structure permettant de stocker les valeurs des touches de contrôle (Page Up, Page Back, Back, Go)
- struct EtatFaders
  Structure permettant de stocker les valeurs des faders.
- class PlaybackWing
  la classe gérant la console wing

Macros

- #define PORT_ENTTEC 3330
- #define INDEX_FIRMWARE 4
- #define INDEX_FLAGS 5
- #define INDEX_TOUCHES_CONTROLE 6
- #define LG_MESSAGE_WODD 28
- #define FADER_NUMERO_0 15
- #define FADER_NUMERO_1 16
- #define FADER_NUMERO_2 17

- #define FADER_NUMERO_3 18
- #define FADER_NUMERO_4 19
- #define FADER_NUMERO_5 20
- #define FADER_NUMERO_6 21
- #define FADER_NUMERO_7 22
- #define FADER_NUMERO_8 23
- #define FADER_NUMERO_9 24
- #define KEY_PRESSED 0
- #define KEY_RELEASED 1
- #define PAGE_UP 7
- #define PAGE_DOWN 6
- #define BACK 5
- #define GO 4

9.27.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.27.2 Documentation des macros

9.27.2.1 #define BACK 5

Référencé par PlaybackWing : :DecoderTouchesControle().

9.27.2.2 #define FADER_NUMERO_0 15

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.3 #define FADER_NUMERO_1 16

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.4 #define FADER_NUMERO_2 17

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.5 #define FADER_NUMERO_3 18

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.6 #define FADER_NUMERO_4 19

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.7 #define FADER_NUMERO_5 20

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.8 #define FADER_NUMERO_6 21

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.9 #define FADER_NUMERO_7 22

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.10 #define FADER_NUMERO_8 23

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.11 #define FADER_NUMERO_9 24

Référencé par PlaybackWing : :ExtraireFaders().

9.27.2.12 #define GO 4

Référencé par PlaybackWing : :DecoderTouchesControle().

9.27.2.13 #define INDEX_FIRMWARE 4

Référencé par PlaybackWing : :debuguerDatagramme(), et PlaybackWing : :Verifier-Datagramme().

9.27.2.14 #define INDEX_FLAGS 5

9.27.2.15 #define INDEX_TOUCHES_CONTROLE 6

Référencé par PlaybackWing : :TraiterDatagramme().

9.27.2.16 #define KEY_PRESSED 0

9.27.2.17 #define KEY_RELEASED 1

9.27.2.18 #define LG_MESSAGE_WODD 28

Référencé par PlaybackWing : :debuguerDatagramme().

9.27.2.19 #define PAGE_DOWN 6

Référencé par PlaybackWing : :DecoderTouchesControle().

9.27.2.20 #define PAGE_UP 7

Référencé par PlaybackWing : :DecoderTouchesControle().

9.27.2.21 #define PORT_ENTTEC 3330

Référencé par PlaybackWing : :PlaybackWing().

9.28 Référence du fichier qextserialenumerator.cpp

```
#include "qextserialenumerator.h" #include "qextserialenumerator-
_p.h"  #include <QtCore/QDebug>  #include <QtCore/QMeta-
Type> #include <QtCore/QRegExp> #include "moc_qextserialenumerator.-
cpp"
```

9.29 Référence du fichier qextserialenumerator.h

```
#include <QtCore/QList> #include <QtCore/QObject> #include
"qextserialport_global.h"
```

Classes

- class QextPortInfo
  The QextPortInfo class containing port information.
- class QextSerialEnumerator
  The QextSerialEnumerator class provides list of ports available in the system.

9.30 Référence du fichier qextserialenumerator_linux.cpp

```
#include "qextserialenumerator.h" #include "qextserialenumerator-
_p.h" #include <QtCore/QDebug> #include <QtCore/QString-
List> #include <QtCore/QDir>
```

Fonctions

- static QextPortInfo portInfoFromDevice (struct udev_device *dev)

9.30.1 Documentation des fonctions

9.30.1.1 static QextPortInfo portInfoFromDevice (struct udev_device * *dev*) [static]

Références QextPortInfo : :physName, QextPortInfo : :portName, QextPortInfo-
: :productID, et QextPortInfo : :vendorID.

Référencé par QextSerialEnumeratorPrivate : :getPorts_sys().

```
{
    QString vendor = QString::fromLatin1(udev_device_get_property_value(dev, "
      ID_VENDOR_ID"));
    QString product = QString::fromLatin1(udev_device_get_property_value(dev, "
      ID_MODEL_ID"));

    QextPortInfo pi;
    pi.vendorID = vendor.toInt(0, 16);
    pi.productID = product.toInt(0, 16);
    pi.portName = QString::fromLatin1(udev_device_get_devnode(dev));
    pi.physName = pi.portName;
```

```
    return pi;
}
```

9.31 Référence du fichier qextserialenumerator_osx.cpp

```
#include "qextserialenumerator.h" #include "qextserialenumerator-
_p.h"  #include <QtCore/QDebug>  #include <IOKit/serial/-
IOSerialKeys.h>    #include <CoreFoundation/CFNumber.h> ×
#include <sys/param.h>
```

Fonctions

– void deviceDiscoveredCallbackOSX (void ∗ctxt, io_iterator_t serialPortIterator)
– void deviceTerminatedCallbackOSX (void ∗ctxt, io_iterator_t serialPortIterator)

9.31.1 Documentation des fonctions

9.31.1.1 void deviceDiscoveredCallbackOSX (void ∗ *ctxt,* io_iterator_t *serialPortIterator*)

```
{
    QextSerialEnumeratorPrivate *d = (QextSerialEnumeratorPrivate *)ctxt;
    io_object_t serialService;
    while ((serialService = IOIteratorNext(serialPortIterator)))
        d->onDeviceDiscoveredOSX(serialService);
}
```

9.31.1.2 void deviceTerminatedCallbackOSX (void ∗ *ctxt,* io_iterator_t *serialPortIterator*)

```
{
    QextSerialEnumeratorPrivate *d = (QextSerialEnumeratorPrivate *)ctxt;
    io_object_t serialService;
    while ((serialService = IOIteratorNext(serialPortIterator)))
        d->onDeviceTerminatedOSX(serialService);
}
```

9.32 Référence du fichier qextserialenumerator_p.h

```
#include "qextserialenumerator.h"
```

Classes

– class QextSerialEnumeratorPrivate

9.33 Référence du fichier qextserialenumerator_unix.cpp

```
#include "qextserialenumerator.h" #include "qextserialenumerator-
_p.h" #include <QtCore/QDebug>
```

9.34 Référence du fichier qextserialenumerator_win.cpp

```
#include "qextserialenumerator.h" #include "qextserialenumerator-
_p.h"  #include <QtCore/QDebug>  #include <QtCore/QMeta-
Type>  #include <QtCore/QRegExp>  #include <objbase.h> ×
#include <initguid.h>  #include <setupapi.h>  #include
<dbt.h> #include "qextserialport.h"
```

Macros

- #define QStringToTCHAR(x) x.local8Bit().constData()
- #define PQStringToTCHAR(x) x->local8Bit().constData()
- #define TCHARToQString(x) QString : :fromLocal8Bit((char ∗)(x))
- #define TCHARToQStringN(x, y) QString : :fromLocal8Bit((char ∗)(x),(y))

Fonctions

- static QString getRegKeyValue (HKEY key, LPCTSTR property)
- static QString getDeviceProperty (HDEVINFO devInfo, PSP_DEVINFO_DATA devData, DWORD property)
- static bool getDeviceDetailsWin (QextPortInfo ∗portInfo, HDEVINFO devInfo, PSP_DEVINFO_DATA devData, WPARAM wParam=DBT_DEVICEARRIVAL)
- static void enumerateDevicesWin (const GUID &guid, QList< QextPortInfo > ∗infoList)
- static bool lessThan (const QextPortInfo &s1, const QextPortInfo &s2)

Variables

- const GUID deviceClassGuids []

9.34.1 Documentation des macros

9.34.1.1 #define PQStringToTCHAR(*x*) x->local8Bit().constData()

9.34.1.2 #define QStringToTCHAR(*x*) x.local8Bit().constData()

9.34.1.3 #define TCHARToQString(*x*) QString : :fromLocal8Bit((char ∗)(x))

Référencé par getDeviceProperty(), et getRegKeyValue().

9.34.1.4 #define TCHARToQStringN(*x, y*) QString : :fromLocal8Bit((char ∗)(x),(y))

9.34.2 Documentation des fonctions

9.34.2.1 static void enumerateDevicesWin (const GUID & *guid,* QList< QextPortInfo > ∗ *infoList*) [static]

Références getDeviceDetailsWin(), QextPortInfo : :portName, QextPortInfo : :productID, et QextPortInfo : :vendorID.

```
{
    HDEVINFO devInfo;
    if ((devInfo = ::SetupDiGetClassDevs(&guid, NULL, NULL, DIGCF_PRESENT)) !=
      INVALID_HANDLE_VALUE) {
        SP_DEVINFO_DATA devInfoData;
        devInfoData.cbSize = sizeof(SP_DEVINFO_DATA);
        for(int i = 0; ::SetupDiEnumDeviceInfo(devInfo, i, &devInfoData); i++)
      {
            QextPortInfo info;
            info.productID = info.vendorID = 0;
            getDeviceDetailsWin(&info, devInfo, &devInfoData);
            if (!info.portName.startsWith(QLatin1String("LPT"),
      Qt::CaseInsensitive))
                infoList->append(info);
        }
        ::SetupDiDestroyDeviceInfoList(devInfo);
    }
}
```

9.34.2.2 static bool getDeviceDetailsWin (QextPortInfo * *portInfo,* HDEVINFO *devInfo,* PSP_DEVINFO_DATA *devData,* WPARAM *wParam* = DBT_DEVICEARRIVAL) [static]

Références QextPortInfo : :enumName, QextPortInfo : :friendName, getDeviceProperty(), getRegKeyValue(), QextPortInfo : :physName, QextPortInfo : :portName, QextPortInfo : :productID, et QextPortInfo : :vendorID.

Référencé par enumerateDevicesWin().

```
{
    portInfo->friendName = getDeviceProperty(devInfo, devData,
      SPDRP_FRIENDLYNAME);
    if (wParam == DBT_DEVICEARRIVAL)
        portInfo->physName = getDeviceProperty(devInfo, devData,
      SPDRP_PHYSICAL_DEVICE_OBJECT_NAME);
    portInfo->enumName = getDeviceProperty(devInfo, devData,
      SPDRP_ENUMERATOR_NAME);
    QString hardwareIDs = getDeviceProperty(devInfo, devData, SPDRP_HARDWAREID)
      ;
    HKEY devKey = ::SetupDiOpenDevRegKey(devInfo, devData, DICS_FLAG_GLOBAL, 0,
       DIREG_DEV, KEY_QUERY_VALUE);
    portInfo->portName = getRegKeyValue(devKey, TEXT("PortName"));
    QRegExp idRx(QLatin1String("VID_(\\w+)&PID_(\\w+)"));
    if (hardwareIDs.toUpper().contains(idRx)) {
        bool dummy;
        portInfo->vendorID = idRx.cap(1).toInt(&dummy, 16);
        portInfo->productID = idRx.cap(2).toInt(&dummy, 16);
        //qDebug() << "got vid:" << vid << "pid:" << pid;
    }
    return true;
}
```

9.34.2.3 static QString getDeviceProperty (HDEVINFO *devInfo,* PSP_DEVINFO_DATA *devData,* DWORD *property*) [static]

Références TCHARToQString.

Référencé par getDeviceDetailsWin().

```
{
    DWORD buffSize = 0;
    ::SetupDiGetDeviceRegistryProperty(devInfo, devData, property, NULL, NULL,
      0, &buffSize);
    BYTE *buff = new BYTE[buffSize];
```

```
    ::SetupDiGetDeviceRegistryProperty(devInfo, devData, property, NULL, buff,
      buffSize, NULL);
    QString result = TCHARToQString(buff);
    delete [] buff;
    return result;
}
```

9.34.2.4 static QString getRegKeyValue (HKEY *key*, LPCTSTR *property*) [static]

Références TCHARToQString.

Référencé par getDeviceDetailsWin().

```
{
    DWORD size = 0;
    DWORD type;
    ::RegQueryValueEx(key, property, NULL, NULL, NULL, &size);
    BYTE *buff = new BYTE[size];
    QString result;
    if (::RegQueryValueEx(key, property, NULL, &type, buff, &size) ==
      ERROR_SUCCESS)
        result = TCHARToQString(buff);
    ::RegCloseKey(key);
    delete [] buff;
    return result;
}
```

9.34.2.5 static bool lessThan (const QextPortInfo & *s1*, const QextPortInfo & *s2*) [static]

Références QextPortInfo : :portName.

```
{
    if (s1.portName.startsWith(QLatin1String("COM"))
            && s2.portName.startsWith(QLatin1String("COM"))) {
        return s1.portName.mid(3).toInt()<s2.portName.mid(3).toInt();
    }
    return s1.portName < s2.portName;
}
```

9.34.3 Documentation des variables

9.34.3.1 const GUID deviceClassGuids[]

Valeur initiale :

```
{

    {0x4D36E978, 0xE325, 0x11CE, {0xBF, 0xC1, 0x08, 0x00, 0x2B, 0xE1, 0x03,
      0x18}},

    {0x4D36E96D, 0xE325, 0x11CE, {0xBF, 0xC1, 0x08, 0x00, 0x2B, 0xE1, 0x03,
      0x18}},

    {0xE0CBF06C, 0xCD8B, 0x4647, {0xBB, 0x8A, 0x26, 0x3B, 0x43, 0xF0, 0xF9,
      0x74}},

    {0xDF799E12, 0x3C56, 0x421B, {0xB2, 0x98, 0xB6, 0xD3, 0x64, 0x2B, 0xC8,
      0x78}}
}
```

9.35 Référence du fichier qextserialport.cpp

```
#include "qextserialport.h"  #include "qextserialport_p.-
h" #include <stdio.h> #include <QtCore/QDebug> #include
<QtCore/QReadLocker>     #include <QtCore/QWriteLocker> ×
#include "moc_qextserialport.cpp"
```

9.36 Référence du fichier qextserialport.h

```
#include <QtCore/QIODevice>    #include "qextserialport_-
global.h"
```

Classes

- class PortSettings
  The PortSettings class contain port settings.
- class QextSerialPort
  The QextSerialPort class encapsulates a serial port on both POSIX and Windows systems.

Macros

- #define LS_CTS 0x01
- #define LS_DSR 0x02
- #define LS_DCD 0x04
- #define LS_RI 0x08
- #define LS_RTS 0x10
- #define LS_DTR 0x20
- #define LS_ST 0x40
- #define LS_SR 0x80
- #define E_NO_ERROR 0
- #define E_INVALID_FD 1
- #define E_NO_MEMORY 2
- #define E_CAUGHT_NON_BLOCKED_SIGNAL 3
- #define E_PORT_TIMEOUT 4
- #define E_INVALID_DEVICE 5
- #define E_BREAK_CONDITION 6
- #define E_FRAMING_ERROR 7
- #define E_IO_ERROR 8
- #define E_BUFFER_OVERRUN 9
- #define E_RECEIVE_OVERFLOW 10
- #define E_RECEIVE_PARITY_ERROR 11
- #define E_TRANSMIT_OVERFLOW 12
- #define E_READ_FAILED 13
- #define E_WRITE_FAILED 14
- #define E_FILE_NOT_FOUND 15
- #define E_PERMISSION_DENIED 16
- #define E_AGAIN 17

Énumérations

- enum BaudRateType { BAUD110 = 110, BAUD300 = 300, BAUD600 = 600, BAUD1200 = 1200, BAUD2400 = 2400, BAUD4800 = 4800, BAUD9600 = 9600, BAUD19200 = 19200, BAUD38400 = 38400, BAUD57600 = 57600, BAUD115200 = 115200 }

– enum DataBitsType { DATA_5 = 5, DATA_6 = 6, DATA_7 = 7, DATA_8 = 8 }
– enum ParityType { PAR_NONE, PAR_ODD, PAR_EVEN, PAR_SPACE }
– enum StopBitsType { STOP_1, STOP_2 }
– enum FlowType { FLOW_OFF, FLOW_HARDWARE, FLOW_XONXOFF }

9.36.1 Documentation des macros

9.36.1.1 #define E_AGAIN 17

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :translateError().

9.36.1.2 #define E_BREAK_CONDITION 6

Référencé par QextSerialPort : :errorString().

9.36.1.3 #define E_BUFFER_OVERRUN 9

Référencé par QextSerialPort : :errorString().

9.36.1.4 #define E_CAUGHT_NON_BLOCKED_SIGNAL 3

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :translateError().

9.36.1.5 #define E_FILE_NOT_FOUND 15

Référencé par QextSerialPort : :errorString().

9.36.1.6 #define E_FRAMING_ERROR 7

Référencé par QextSerialPort : :errorString().

9.36.1.7 #define E_INVALID_DEVICE 5

Référencé par QextSerialPort : :errorString().

9.36.1.8 #define E_INVALID_FD 1

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :translateError().

9.36.1.9 #define E_IO_ERROR 8

Référencé par QextSerialPort : :errorString().

9.36.1.10 #define E_NO_ERROR 0

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :QextSerialPortPrivate().

9.36.1.11 #define E_NO_MEMORY 2

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :translateError().

9.36.1.12 #define E_PERMISSION_DENIED 16

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :translateError().

9.36.1.13 #define E_PORT_TIMEOUT 4

Référencé par QextSerialPort : :errorString().

9.36.1.14 #define E_READ_FAILED 13

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :readData_sys().

9.36.1.15 #define E_RECEIVE_OVERFLOW 10

Référencé par QextSerialPort : :errorString().

9.36.1.16 #define E_RECEIVE_PARITY_ERROR 11

Référencé par QextSerialPort : :errorString().

9.36.1.17 #define E_TRANSMIT_OVERFLOW 12

Référencé par QextSerialPort : :errorString().

9.36.1.18 #define E_WRITE_FAILED 14

Référencé par QextSerialPort : :errorString(), et QextSerialPortPrivate : :writeData_sys().

9.36.1.19 #define LS_CTS 0x01

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.20 #define LS_DCD 0x04

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.21 #define LS_DSR 0x02

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.22 #define LS_DTR 0x20

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.23 #define LS_RI 0x08

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.24 #define LS_RTS 0x10

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.25 #define LS_SR 0x80

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.1.26 #define LS_ST 0x40

Référencé par QextSerialPortPrivate : :lineStatus_sys().

9.36.2 Documentation du type de l'énumération

9.36.2.1 enum BaudRateType

Valeurs énumérées :

BAUD110

BAUD300

BAUD600

BAUD1200

BAUD2400

BAUD4800

BAUD9600

BAUD19200

BAUD38400

BAUD57600

BAUD115200

```
{
#if defined(Q_OS_UNIX) || defined(qdoc)

    BAUD50 = 50,                //POSIX ONLY
    BAUD75 = 75,                //POSIX ONLY
    BAUD134 = 134,              //POSIX ONLY
    BAUD150 = 150,              //POSIX ONLY
    BAUD200 = 200,              //POSIX ONLY
    BAUD1800 = 1800,            //POSIX ONLY
#  if defined(B76800) || defined(qdoc)

    BAUD76800 = 76800,          //POSIX ONLY
#  endif

#  if (defined(B230400) && defined(B4000000)) || defined(qdoc)

    BAUD230400 = 230400,        //POSIX ONLY
    BAUD460800 = 460800,        //POSIX ONLY
    BAUD500000 = 500000,        //POSIX ONLY
    BAUD576000 = 576000,        //POSIX ONLY
```

```
    BAUD921600 = 921600,        //POSIX ONLY
    BAUD1000000 = 1000000,      //POSIX ONLY
    BAUD1152000 = 1152000,      //POSIX ONLY
    BAUD1500000 = 1500000,      //POSIX ONLY
    BAUD2000000 = 2000000,      //POSIX ONLY
    BAUD2500000 = 2500000,      //POSIX ONLY
    BAUD3000000 = 3000000,      //POSIX ONLY
    BAUD3500000 = 3500000,      //POSIX ONLY
    BAUD4000000 = 4000000,      //POSIX ONLY
#  endif

#endif //Q_OS_UNIX

#if defined(Q_OS_WIN) || defined(qdoc)

    BAUD14400 = 14400,          //WINDOWS ONLY
    BAUD56000 = 56000,          //WINDOWS ONLY
    BAUD128000 = 128000,        //WINDOWS ONLY
    BAUD256000 = 256000,        //WINDOWS ONLY
#endif  //Q_OS_WIN

    BAUD110 = 110,
    BAUD300 = 300,
    BAUD600 = 600,
    BAUD1200 = 1200,
    BAUD2400 = 2400,
    BAUD4800 = 4800,
    BAUD9600 = 9600,
    BAUD19200 = 19200,
    BAUD38400 = 38400,
    BAUD57600 = 57600,
    BAUD115200 = 115200
};
```

9.36.2.2 enum DataBitsType

Valeurs énumérées :

DATA_5

DATA_6

DATA_7

DATA_8

```
{
    DATA_5 = 5,
    DATA_6 = 6,
    DATA_7 = 7,
    DATA_8 = 8
};
```

9.36.2.3 enum FlowType

Valeurs énumérées :

FLOW_OFF

FLOW_HARDWARE

FLOW_XONXOFF

```
{
    FLOW_OFF,
    FLOW_HARDWARE,
    FLOW_XONXOFF
};
```

9.36.2.4 enum ParityType

Valeurs énumérées :

PAR_NONE

PAR_ODD

PAR_EVEN

PAR_SPACE

```
{
    PAR_NONE,
    PAR_ODD,
    PAR_EVEN,
#if defined(Q_OS_WIN) || defined(qdoc)

    PAR_MARK,               //WINDOWS ONLY
#endif

    PAR_SPACE
};
```

9.36.2.5 enum StopBitsType

Valeurs énumérées :

STOP_1

STOP_2

```
{
    STOP_1,
#if defined(Q_OS_WIN) || defined(qdoc)

    STOP_1_5,               //WINDOWS ONLY
#endif

    STOP_2
};
```

9.37 Référence du fichier qextserialport_global.h

```
#include <QtCore/QtGlobal>
```

Macros

- #define QEXTSERIALPORT_EXPORT
- #define QESP_PORTABILITY_WARNING qWarning
- #define QESP_WARNING qWarning

9.37.1 Documentation des macros

9.37.1.1 #define QESP_PORTABILITY_WARNING qWarning

Référencé par QextSerialPortPrivate : :setBaudRate(), QextSerialPortPrivate : :setParity(), et QextSerialPortPrivate : :setStopBits().

9.37.1.2 #define QESP_WARNING qWarning

Référencé par QextSerialPortPrivate : :setBaudRate(), QextSerialPortPrivate : :setDataBits(), QextSerialPortPrivate : :setParity(), QextSerialPortPrivate : :setStopBits(), et QextSerialEnumerator : :setUpNotifications().

9.37.1.3 #define QEXTSERIALPORT_EXPORT

9.38 Référence du fichier qextserialport_p.h

`#include "qextserialport.h" #include <QtCore/QReadWriteLock> #include <stdlib.h>`

Classes

– class QextReadBuffer
– class QextSerialPortPrivate

9.39 Référence du fichier qextserialport_unix.cpp

`#include "qextserialport.h" #include "qextserialport_p.h" #include <fcntl.h> #include <stdio.h> #include <errno.h> #include <unistd.h> #include <sys/time.h> #include <sys/ioctl.h> #include <sys/select.h> #include <QtCore/QMutexLocker> #include <QtCore/QDebug> #include <QtCore/QSocketNotifier>`

Fonctions

– static QString fullPortName (const QString &name)
– static void setBaudRate2Termios (termios ∗config, int baudRate)

9.39.1 Documentation des fonctions

9.39.1.1 static QString fullPortName (const QString & *name*) [static]

Référencé par QextSerialPortPrivate : :open_sys().

```
{
    if (name.startsWith(QLatin1Char('/')))
        return name;
    return QLatin1String("/dev/")+name;
}
```

9.39.1.2 static void setBaudRate2Termios (termios ∗ *config*, int *baudRate*) [static]

Référencé par QextSerialPortPrivate : :updatePortSettings().

```
{
#ifdef CBAUD

    config->c_cflag &= (~CBAUD);
    config->c_cflag |= baudRate;
#else

    ::cfsetispeed(config, baudRate);
    ::cfsetospeed(config, baudRate);
#endif

}
```

9.40 Référence du fichier qextserialport_win.cpp

```
#include "qextserialport.h"  #include "qextserialport_p.-
h" #include <QtCore/QThread> #include <QtCore/QReadWrite-
Lock> #include <QtCore/QMutexLocker> #include <QtCore/Q-
Debug> #include <QtCore/QRegExp> #include <QtCore/QMeta-
Type> #include <QtCore/QWinEventNotifier>
```

Fonctions

– static QString fullPortNameWin (const QString &name)

9.40.1 Documentation des fonctions

9.40.1.1 static QString fullPortNameWin (const QString & *name*) [static]

```
{
    QRegExp rx(QLatin1String("^COM(\\d+)"));
    QString fullName(name);
    if (fullName.contains(rx))
        fullName.prepend(QLatin1String("\\\\.\\"));
    return fullName;
}
```

9.41 Référence du fichier README.dox

9.42 Référence du fichier scene.h

Déclaration des structures Scene et SceneSequence.

```
#include <QTime>
```

Classes

– struct Scene
  Structure permettant de représenter le contenu d'un plan d'éclairage d'appareils XML.
– struct SceneSequence

9.42.1 Description détaillée

Auteur

Reynier Tony

Version

1.1

9.43 Référence du fichier sequence.h

Déclaration de la structure Sequence.

```
#include "scene.h"
```

Classes

– struct Sequence
Structure permettant de représenter le contenu d'une séquence d'enchainement de scènes temporisées.

9.43.1 Description détaillée

Auteur

Reynier Tony

Version

1.1

9.44 Référence du fichier ThreadAttente.cpp

```
#include "ThreadAttente.h"
```

9.45 Référence du fichier ThreadAttente.h

Déclaration de la classe ThreadAttente.

```
#include <QThread>
```

Classes

– struct ThreadAttente
Classe heritant de QThread et servant à génerer les temps d'attentes entre les scènes d'une séquence.

9.45.1 Description détaillée

Auteur

Reynier Tony

Version

1.1

9.46 Référence du fichier types.h

Macros

- #define HIDDEN static
- #define PACK

Définitions de type

- typedef unsigned long ULONG
- typedef unsigned short UWORD
- typedef unsigned char UBYTE
- typedef unsigned char * STRPTR
- typedef unsigned long DWORD
- typedef unsigned short WORD
- typedef unsigned char BYTE
- typedef unsigned long WORD32
- typedef unsigned short WORD16
- typedef unsigned char WORD8

9.46.1 Documentation des macros

9.46.1.1 #define HIDDEN static

9.46.1.2 #define PACK

9.46.2 Documentation des définitions de type

9.46.2.1 typedef unsigned char BYTE

9.46.2.2 typedef unsigned long DWORD

9.46.2.3 typedef unsigned char* STRPTR

9.46.2.4 typedef unsigned char UBYTE

9.46.2.5 typedef unsigned long ULONG

9.46.2.6 typedef unsigned short UWORD

9.46.2.7 typedef unsigned short WORD

9.46.2.8 typedef unsigned short WORD16

9.46.2.9 typedef unsigned long WORD32

9.46.2.10 typedef unsigned char WORD8

9.47 Référence du fichier xmlutilitaire.cpp

Définition de la classe XMLUtilitaire.

```
#include "xmlutilitaire.h"    #include "DMXProjecteur.h" ×
#include "DMXLaser.h"  #include "DMXLyre.h"  #include "D-
MXPAR.h"  #include "DMXScanner.h"  #include "DMXSpecial.-
h" #include "interface.h" #include <QDebug> #include <Q-
MessageBox>
```

9.47.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.0

9.48 Référence du fichier xmlutilitaire.h

Déclaration de la classe XMLUtilitaire.

```
#include <QtCore>  #include <QtXml>  #include "scene.h" ×
#include "sequence.h" #include "spectacle.h" #include "id-
ProjecteurSauvegarde.h"  #include "interface.h"  #include
"console.h" #include "PlaybackWing.h"
```

Classes

– class XMLUtilitaire
Classe gérant les fichiers XML de l'application.

9.48.1 Description détaillée

Auteur

Demont Thomas, Reynier Tony

Version

1.1