

ScStw shared libraries

Generated by Doxygen 1.9.4

1 ScStw Libraries documentation	1
1.1 Introduction	1
1.2 Installation	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 ScStw Class Reference	9
5.2 ScStwLibraries Class Reference	9
5.3 ScStwRace Class Reference	9
5.3.1 Detailed Description	11
5.3.2 Member Function Documentation	11
5.3.2.1 reset	12
5.3.2.2 start	12
5.3.2.3 stop	12
5.4 ScStwSetting Class Reference	12
5.5 ScStwSettings Class Reference	13
5.5.1 Member Enumeration Documentation	14
5.5.1.1 BaseStationSetting	15
5.6 ScStwSoundPlayer Class Reference	15
5.6.1 Detailed Description	16
5.6.2 Constructor & Destructor Documentation	16
5.6.2.1 ScStwSoundPlayer()	16
5.6.3 Member Function Documentation	16
5.6.3.1 cancel	16
5.6.3.2 play	17
5.6.3.3 waitForSoundFinish	17
5.7 ScStwStartSoundPlayer Class Reference	18
5.8 ScStwTimer Class Reference	18
5.8.1 Detailed Description	20
5.8.2 Member Enumeration Documentation	21
5.8.2.1 ReadyState	21
5.8.2.2 TimerState	21
5.8.3 Constructor & Destructor Documentation	21
5.8.3.1 ScStwTimer()	22
5.8.4 Member Function Documentation	22
5.8.4.1 cancel	22

5.8.4.2	getCurrentTime	22
5.8.4.3	getLetter	23
5.8.4.4	getReactionTime	23
5.8.4.5	getReadyState	23
5.8.4.6	getState	23
5.8.4.7	getText	24
5.8.4.8	getWantsToBeDisabled	24
5.8.4.9	handleClimberStart	24
5.8.4.10	readyStateChanged	24
5.8.4.11	reset	25
5.8.4.12	setDisabled	25
5.8.4.13	setResult	25
5.8.4.14	setState	26
5.8.4.15	setWantsToBeDisabled	26
5.8.4.16	start [1/2]	26
5.8.4.17	start [2/2]	26
5.8.4.18	stop [1/2]	27
5.8.4.19	stop [2/2]	27
5.8.4.20	technicalIncident	27
5.8.4.21	wantsToBeDisabledChanged	28
5.8.4.22	wildcard	28
6	File Documentation	29
6.1	ScStw.hpp	29
6.2	scstwlibraries.h	31
6.3	ScStwLibraries_global.h	32
6.4	scstwrace.h	32
6.5	scstwsetting.h	34
6.6	scstwsettings.h	35
6.7	scstwsoundplayer.h	36
6.8	scstwstartsoundplayer.h	37
6.9	scstwtimer.h	38
Index		41

Chapter 1

ScStw Libraries documentation

1.1 Introduction

This library is meant for usage with the Speed climbing stopwatch project. It contains some helper classes to build a client application for the [ScStw](#) basestation with Qt.

1.2 Installation

```
cd yourRepo
git submodule add https://git.itsblue.de/ScStw/shared-libraries/
git submodule update --init --recursive
```

And in your MyProject.pro include the .pri file:

```
include($PWD/shared-libraries/ScStwLibraries/ScStwLibraries.pri)
```


Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

QObject	
ScStw	9
ScStwLibraries	9
ScStwRace	9
ScStwSetting	12
ScStwSettings	13
ScStwSoundPlayer	15
ScStwStartSoundPlayer	18
ScStwTimer	18

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ScStw	Some shared functions and enums for use in the ScStw project	9
ScStwLibraries	9
ScStwRace	Can be used to measure timings of climbing races with multiple lanes at once	9
ScStwSetting	12
ScStwSettings	13
ScStwSoundPlayer	Used for ultra low latency sound playback of the speed climbing start tones and commands	15
ScStwStartSoundPlayer	18
ScStwTimer	Used for advanced time measurement	18

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

/drone/src/ScStwLibraries/headers/ScStw.hpp	29
/drone/src/ScStwLibraries/headers/scstwlibraries.h	31
/drone/src/ScStwLibraries/headers/ScStwLibraries_global.h	32
/drone/src/ScStwLibraries/headers/scstwrace.h	32
/drone/src/ScStwLibraries/headers/scstwsetting.h	34
/drone/src/ScStwLibraries/headers/scstwsettings.h	35
/drone/src/ScStwLibraries/headers/scstwsoundplayer.h	36
/drone/src/ScStwLibraries/headers/scstwstartsoundplayer.h	37
/drone/src/ScStwLibraries/headers/scstwtimer.h	38

Chapter 5

Class Documentation

5.1 ScStw Class Reference

The [ScStw](#) class provides some shared functions and enums for use in the [ScStw](#) project.

```
#include <ScStw.hpp>
```

Inheritance diagram for ScStw:

5.2 ScStwLibraries Class Reference

Inheritance diagram for ScStwLibraries:

Collaboration diagram for ScStwLibraries:

Static Public Member Functions

- static void **init** ()

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwlibraries.h
- /drone/src/ScStwLibraries/sources/scstwlibraries.cpp

5.3 ScStwRace Class Reference

The [ScStwRace](#) class can be used to measure timings of climbing races with multiple lanes at once.

```
#include <scstwrace.h>
```

Inheritance diagram for ScStwRace:

Collaboration diagram for ScStwRace:

Public Types

- enum **RaceState** {
IDLE , **PREPAIRING** , **WAITING** , **STARTING** ,
RUNNING , **STOPPED** , **INCIDENT** }

Public Slots

- virtual [ScStw::StatusCode](#) **start** (bool asynchronous=true)
Function to start the race.
- virtual [ScStw::StatusCode](#) **stop** ()
Function to stop the currently running race.
- virtual [ScStw::StatusCode](#) **reset** ()
Function to reset a stopped race.
- virtual [ScStw::StatusCode](#) **cancel** ()
- virtual [ScStw::StatusCode](#) **setTimerDisabled** (int id, bool disabled)
- virtual Q_INVOKABLE bool **addTimer** ([ScStwTimer](#) *timer)
- RaceState **getState** ()
- virtual QVariantMap **getCurrentStartDelay** ()
- QList< [ScStwTimer](#) * > **getTimers** ()
- QVariantList **getTimerDetailList** ()
- QVariantMap **getDetails** ()
- bool **getCompetitionMode** ()
- virtual bool **getReadySoundEnabled** ()
- [ScStwSettings](#) * **getSettings** ()
- void **setSettings** ([ScStwSettings](#) *settings)
- bool **getAutoRefreshTimerText** ()
- void **setAutoRefreshTimerText** (bool autoRefresh)

Signals

- void **startTimers** ()
- void **stopTimers** (int type)
- void **resetTimers** ()
- void **stateChanged** (RaceState state)
- void **currentStartDelayChanged** ()
- void **timersChanged** ()
- void **isReadyForNextStateChanged** ()
- void **detailsChanged** ()
- void **competitionModeChanged** ()
- void **readySoundEnabledChanged** ()
- void **settingsChanged** ()
- void **autoRefreshTimerTextChanged** ()

Public Member Functions

- **ScStwRace** (QObject *parent=nullptr)
- **ScStwRace** ([ScStwSettings](#) *settings, QObject *parent=nullptr)

Protected Member Functions

- void **setState** (RaceState newState)

Protected Attributes

- QList< [ScStwTimer](#) * > **timers**

Properties

- RaceState **state**
- QVariantList **timers**
- QVariantMap **currentStartDelay**
- bool **isReadyForNextState**
- bool **competitionMode**
- bool **readySoundEnabled**
- QVariantMap **details**
- [ScStwSettings](#) * **settings**
- bool **autoRefreshTimerText**

Friends

- class **ScStwRemoteRace**

5.3.1 Detailed Description

The [ScStwRace](#) class can be used to measure timings of climbing races with multiple lanes at once.

The [ScStwRace](#) is a container to manage multiple timers at a time and introduces a proper start sequence with start commands ('At your Marks' and 'Ready') and the official IFSC start signal.

Basic usage:

```
ScStwRace race;
// add two timers
race.addTimer(new ScStwTimer());
race.addTimer(new ScStwTimer());
// start a race
race.start();
```

5.3.2 Member Function Documentation

5.3.2.1 reset

```
ScStw::StatusCode ScStwRace::reset ( ) [virtual], [slot]
```

Function to reset a stopped race.

Returns

5.3.2.2 start

```
ScStw::StatusCode ScStwRace::start (
    bool asynchronous = true ) [virtual], [slot]
```

Function to start the race.

Parameters

<i>asynchronous</i>	if the function should just start the start sequence and then quit (true) or if should wait until the start sequence is over and quit after that (false)
---------------------	--

Returns

200: OK; 904: state not matching

5.3.2.3 stop

```
ScStw::StatusCode ScStwRace::stop ( ) [virtual], [slot]
```

Function to stop the currently running race.

Returns

200: OK; 904: state not matching

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwrace.h
- /drone/src/ScStwLibraries/sources/scstwrace.cpp

5.4 ScStwSetting Class Reference

Inheritance diagram for ScStwSetting:

Collaboration diagram for ScStwSetting:

Public Slots

- QVariant **getValue** ()
- void **setValue** (QVariant value)

Signals

- void **valueChanged** ()

Protected Slots

- void **handleSettingChange** (int key, int keyLevel, QVariant value)

Protected Member Functions

- **ScStwSetting** (int key, int keyLevel, [ScStwSettings](#) *scStwSettings, QObject *parent)

Protected Attributes

- int **key**
- int **keyLevel**
- bool **hasToReload**

Properties

- QVariant **value**
- QVariant **readonlyValue**

Friends

- class **ScStwSettings**

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwsetting.h
- /drone/src/ScStwLibraries/sources/scstwsetting.cpp

5.5 ScStwSettings Class Reference

Inheritance diagram for ScStwSettings:

Collaboration diagram for ScStwSettings:

Public Types

- enum [BaseStationSetting](#) {
InvalidSetting = -1 , **ReadySoundEnableSetting** , **ReadySoundDelaySetting** , **AtYourMarksSoundEnableSetting** ,
AtYourMarksSoundDelaySetting , **SoundVolumeSetting** , **CompetitionModeSetting** }
The BaseStationSetting enum contains all settings of the base station that can be changed by a client.
- enum **KeyLevelEnum** { **KeyLevel** = 0 }
- typedef QString(* **keyToStringConverter**) (int)
- typedef QVariant::Type(* **keyToTypeConverter**) (int)

Signals

- void **settingChanged** (int key, int keyLevel, QVariant value)

Public Member Functions

- **ScStwSettings** (QObject *parent=nullptr, bool overwriteFileOnErrors=true)
- virtual QVariant **readSetting** ([BaseStationSetting](#) key)
- virtual Q_INVOKABLE QVariant **readSetting** (int key, int keyLevel)
- virtual bool **writeSetting** ([BaseStationSetting](#) key, QVariant value)
- virtual Q_INVOKABLE bool **writeSetting** (int key, int keyLevel, QVariant value)
- virtual bool **setDefaultSetting** ([BaseStationSetting](#) key, QVariant defaultValue)
- virtual Q_INVOKABLE bool **setDefaultSetting** (int key, int keyLevel, QVariant defaultValue)
- Q_INVOKABLE [ScStwSetting](#) * **getSetting** (int key, int keyLevel)

Static Public Member Functions

- static [BaseStationSetting](#) **keyFromInt** (int i)
- static QString **keyToString** (int key)
- static QVariant::Type **keyToType** (int key)

Protected Member Functions

- virtual QVariant **readSetting** (QString key, int keyInt=-1, int keyLevel=-1)
- virtual bool **writeSetting** (QString key, QVariant value, int keyInt=-1, int keyLevel=-1)
- virtual bool **setDefaultSetting** (QString key, QVariant defaultValue, int keyInt, int keyLevel=-1)
- bool **registerKeyLevelConverters** (int keyLevel, keyToStringConverter, keyToTypeConverter)

5.5.1 Member Enumeration Documentation

5.5.1.1 BaseStationSetting

```
enum ScStwSettings::BaseStationSetting
```

The BaseStationSetting enum contains all settings of the base station that can be changed by a client.

See also

```
ScStw::baseStationSettingFromInt()
ScStw::baseStationSettingToString()
ScStw::baseStationSettingFromString()
ScStw::baseStationSettings
```

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwsettings.h
- /drone/src/ScStwLibraries/sources/scstwsettings.cpp

5.6 ScStwSoundPlayer Class Reference

The [ScStwSoundPlayer](#) class is used for ultra low latency sound playback of the speed climbing start tones and commands.

```
#include <scstwsoundplayer.h>
```

Inheritance diagram for ScStwSoundPlayer:

Collaboration diagram for ScStwSoundPlayer:

Public Types

- enum **StartSound** {
None = -1 , **AtYourMarks** = 0 , **Ready** = 1 , **Start** = 2 ,
FalseStart = 3 }
- enum **PlayResult** { **Success** = 0 , **Cancelled** = -1 , **Error** = -2 }

Public Slots

- ScStwSoundPlayer::PlayResult [play](#) (StartSound sound, double volume, double *timeOfStart=nullptr)
Function to begin playing the sound of a certain state.
- ScStwSoundPlayer::PlayResult [waitForSoundFinish](#) (double *timeOfStop=nullptr)
Function to wait for the playback to finish.
- bool [cancel](#) ()
Function to cancel the current playback.
- bool [isPlaying](#) ()

Signals

- void **playbackStarted** ()
Emitted whenever a playback started.

Public Member Functions

- [ScStwSoundPlayer](#) (QObject *parent=nullptr)
ScStwSoundPlayer constructor.

5.6.1 Detailed Description

The [ScStwSoundPlayer](#) class is used for ultra low latency sound playback of the speed climbing start tones and commands.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ScStwSoundPlayer()

```
ScStwSoundPlayer::ScStwSoundPlayer (
    QObject * parent = nullptr ) [explicit]
```

[ScStwSoundPlayer](#) constructor.

Parameters

<i>parent</i>	
---------------	--

5.6.3 Member Function Documentation

5.6.3.1 cancel

```
bool ScStwSoundPlayer::cancel ( ) [slot]
```

Function to cancel the current playback.

Note that this function will automatically play the false start tone if the currently playing action is 2

Parameters

<i>volume</i>	the volume to play the false start sound at
---------------	---

Returns

true if the playback was successfully stopped, false otherwise

5.6.3.2 play

```
ScStwSoundPlayer::PlayResult ScStwSoundPlayer::play (
    ScStwSoundPlayer::StartSound sound,
    double volume,
    double * timeOfStart = nullptr ) [slot]
```

Function to begin playing the sound of a certain state.

Parameters

<i>action</i>	The action to play (0: AtYourMarks, 1:Ready, 2:Start)
<i>volume</i>	The volume to play at
<i>timeOfStop</i>	The time the playback actually started (msecs since epoch)

Returns

TODO true if the playback was successfully started, false otherwise

5.6.3.3 waitForSoundFinish

```
ScStwSoundPlayer::PlayResult ScStwSoundPlayer::waitForSoundFinish (
    double * timeOfStop = nullptr ) [slot]
```

Function to wait for the playback to finish.

Parameters

<i>timeOfStop</i>	the point in time when the playback actually stopped (msecs since epoch)
-------------------	--

Returns

false if there was any error (eg. there was no playback currently), true otherwise

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwsoundplayer.h
- /drone/src/ScStwLibraries/sources/scstwsoundplayer.cpp

5.7 ScStwStartSoundPlayer Class Reference

Inheritance diagram for ScStwStartSoundPlayer:

Collaboration diagram for ScStwStartSoundPlayer:

Public Slots

- `bool play` (double volume, double *timeOfStop=nullptr)

Public Member Functions

- `ScStwStartSoundPlayer` (QObject *parent=nullptr)

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwstartsoundplayer.h
- /drone/src/ScStwLibraries/sources/scstwstartsoundplayer.cpp

5.8 ScStwTimer Class Reference

The [ScStwTimer](#) class is used for advanced time measurement.

```
#include <scstwtimer.h>
```

Inheritance diagram for ScStwTimer:

Collaboration diagram for ScStwTimer:

Public Types

- enum [TimerState](#) {
[IDLE](#) , [STARTING](#) , [RUNNING](#) , [WAITING](#) ,
[WON](#) , [LOST](#) , [FAILING](#) , [WILDCARD](#) ,
[FAILED](#) , [CANCELLED](#) , [INCIDENT](#) , [DISABLED](#) }

The TimerState enum contains all state the timer can be in.

- enum [ReadyState](#) {
[IsReady](#) = 0 , [NotInIdleState](#) , [IsDisabled](#) , [ExtensionIsNotConnected](#) ,
[ExtensionBatteryIsCritical](#) , [ClimberIsNotReady](#) }

The ReadyStatus enum contains all possible reasons for a timer not to be ready (>0) and the case that it is ready (0)

Public Slots

- bool [start](#) ()
Function to start the timer.
- virtual bool [start](#) (double timeOfStart)
Function to start the timer at a given point in time (present or future)
- bool [cancel](#) ()
Function to cancel the timer.
- bool [stop](#) ()
Function to stop the timer.
- bool [stop](#) (double timeOfStop)
Function to stop the timer at a given point in time (past or future)
- bool [setResult](#) (TimerState)
Function to assing the result of the race to the timer.
- virtual bool [reset](#) ()
Function to reset the timer.
- [TimerState](#) [getState](#) ()
Function to get the current state of the timer.
- double [getCurrentTime](#) ()
Function to get the current time of the timer.
- double [getReactionTime](#) ()
Function to get the reaction time of the climber.
- QString [getText](#) ()
Function to get the text, a timer display is supposed to show.
- QString [getLetter](#) ()
Function to get the letter of the timer.
- void [setDisabled](#) (bool disabled)
Function to set if the timer is supposed to be disabled.
- bool [getWantsToBeDisabled](#) ()
Function to check if the timer currently wants to be disabled.
- virtual [ScStwTimer::ReadyState](#) [getReadyState](#) ()
Function to get the current ready status of a timer.
- bool [isRunning](#) ()
- bool [isDisabled](#) ()

Signals

- void [stateChanged](#) ([TimerState](#) state)
Emitted when the state of the timer changed.
- void [reactionTimeChanged](#) ()
Emitted when the reaction time changed.
- void [wantsToBeDisabledChanged](#) ([ScStwTimer](#) *timer, bool [wantsToBeDisabled](#))
Emitted when the timer wants its state to be changed by the external handler.
- void [readyStateChanged](#) ([ReadyState](#) readyState)
Emitted when the ready state of the timer changes.

Public Member Functions

- [ScStwTimer](#) (QObject *parent=nullptr)
- [ScStwTimer](#) (QString [letter](#), QObject *parent=nullptr)
ScStwTimer constructor.
- [Q_ENUM](#) ([TimerState](#))

Protected Slots

- void `handleClimberStart` (double `timeOfStart`)
slot to call when the climber has started
- void `setState` (`TimerState` `newState`)
Function to change the state of the timer.
- void `setWantsToBeDisabled` (bool `wantsToBeDisabled`)
Function to set whether the timer currently wants to be disabled.
- void `technicalIncident` ()
Function to set the timer into INCIDENT state immediately.
- bool `wildcard` ()
Function to set the timer into WILDCARD state.

Protected Attributes

- `TimerState` **state**
The current state of the timer.
- double **startTime**
The time the timer was started at.
- double **stopTime**
The time the timer was stopped at.
- double **reactionTime**
the reaction time of the climber
- QString **letter**
The letter (eg. "A" or "B") of the Timer (only one char)
- bool **wantsToBeDisabled**
Defines if the timer currently wants to be disabled or not.

Friends

- class `ScStwRace`

5.8.1 Detailed Description

The `ScStwTimer` class is used for advanced time measurement.

It does not work on its own though. It is recommended to use it in combination with the `ScStwRace` class.

When using standalone:

```
ScStwTimer timer;
// start the timer
timer.start();
// stop the timer
timer.stop();
```

The timer will now go into `ScStw::WAITING` state. That indicates that the timer has stopped and the final result has to be assigned by an external handler.

```
// assign result 'won'
timer.setResult(ScStwTimer::WON);
```

The timer is now in `ScStwTimer::WON` state.

```
// reset the timer
timer.reset();
```

The timer is not in `ScStwTimer::IDLE` state again.

5.8.2 Member Enumeration Documentation

5.8.2.1 ReadyState

enum `ScStwTimer::ReadyState`

The ReadyStatus enum contains all possible reasons for a timer not to be ready (>0) and the case that it is ready (0)

Enumerator

<code>IsReady</code>	Timer is ready for start
<code>NotInIdleState</code>	Timer is not in IDLE state
<code>IsDisabled</code>	Timer is disabled
<code>ExtensionIsNotConnected</code>	Not all extension of the timer are conneted
<code>ExtensionBatteryIsCritical</code>	The battery level of one or more extension is critical or unknown
<code>ClimberIsNotReady</code>	The startpad of the timer is not triggered

5.8.2.2 TimerState

enum `ScStwTimer::TimerState`

The TimerState enum contains all state the timer can be in.

Enumerator

<code>IDLE</code>	Timer is waiting to be started
<code>STARTING</code>	Timer is starting and will react with a false start if the climber starts
<code>RUNNING</code>	Timer is running
<code>WAITING</code>	Timer was stopped and is waiting to be set to either WON or LOST
<code>WON</code>	Timer has won
<code>LOST</code>	Timer has lost
<code>FAILING</code>	Timer encountered a false start and is waiting to be set to either FAILED or WILDCARD
<code>WILDCARD</code>	The opponent has done a false start
<code>FAILED</code>	A false start occured
<code>CANCELLED</code>	Timer was cancelled
<code>INCIDENT</code>	There was a technical incident (most likely a disconnected extension)
<code>DISABLED</code>	Timer is disabled

5.8.3 Constructor & Destructor Documentation

5.8.3.1 ScStwTimer()

```
ScStwTimer::ScStwTimer (
    QString letter,
    QObject * parent = nullptr ) [explicit]
```

[ScStwTimer](#) constructor.

Parameters

<i>parent</i>	the parent object
<i>directControlEnabled</i>	Defines if protected properties (startTimer, stopTime, reactionTime and state) can be directly set from outside.
<i>letter</i>	the letter of the timer (only first char will be used!)

5.8.4 Member Function Documentation

5.8.4.1 cancel

```
bool ScStwTimer::cancel ( ) [slot]
```

Function to cancel the timer.

To do this, the timer has to be in [ScStwTimer::IDLE](#), [ScStwTimer::STARTING](#) or [ScStwTimer::RUNNING](#) state!

Returns

false if the timer was not in the required state and therefore not cancelled, true otherwise

5.8.4.2 getCurrentTime

```
double ScStwTimer::getCurrentTime ( ) [slot]
```

Function to get the current time of the timer.

To do this, the timer has to be in [ScStwTimer::RUNNING](#), [ScStwTimer::WAITING](#), [ScStwTimer::WON](#) or [ScSTw::↔LOST](#) state!

Returns

The current / final time of the timer or -1 if it is not in the required state

5.8.4.3 getLetter

```
QString ScStwTimer::getLetter ( ) [slot]
```

Function to get the letter of the timer.

Returns

The letter of the timer or ""

5.8.4.4 getReactionTime

```
double ScStwTimer::getReactionTime ( ) [slot]
```

Function to get the reaction time of the climber.

Returns

The climbers reaction time

5.8.4.5 getReadyState

```
ScStwTimer::ReadyState ScStwTimer::getReadyState ( ) [virtual], [slot]
```

Function to get the current ready status of a timer.

Returns

The current ready status

5.8.4.6 getState

```
ScStwTimer::TimerState ScStwTimer::getState ( ) [slot]
```

Function to get the current state of the timer.

Returns

current state of the timer

See also

[ScStwTimer::TimerState](#)

5.8.4.7 getText

```
QString ScStwTimer::getText ( ) [slot]
```

Function to get the text, a timer display is supposed to show.

Returns

The text to show

5.8.4.8 getWantsToBeDisabled

```
bool ScStwTimer::getWantsToBeDisabled ( ) [slot]
```

Function to check if the timer currently wants to be disabled.

Returns

true or false

5.8.4.9 handleClimberStart

```
void ScStwTimer::handleClimberStart (
    double timeOfStart ) [protected], [slot]
```

slot to call when the climber has started

Parameters

<i>timeOfStart</i>	time (msecs since epoch) when the climber started
--------------------	---

5.8.4.10 readyStateChanged

```
void ScStwTimer::readyStateChanged (
    ReadyState readyState ) [signal]
```

Emitted when the ready state of the timer changes.

Parameters

<i>readyState</i>	the new ReadyState
-------------------	--------------------

5.8.4.11 reset

```
bool ScStwTimer::reset ( ) [virtual], [slot]
```

Function to reset the timer.

To do this, the timer has to be in [ScStwTimer::WON](#) or [ScSTw::LOST](#) state!

Returns

false if the timer was not in the required state and therefore not reset, true otherwise

5.8.4.12 setDisabled

```
void ScStwTimer::setDisabled (
    bool disabled ) [slot]
```

Function to set if the timer is supposed to be disabled.

!!! CAUTION use this function with care, it immediately changes the state of the timer !!! It is recommended to only use this function to change the timers state after the [ScStwTimer::requestTimerEnableChange\(\)](#) signal was called, during the race, the timer is used in, is in IDLE state.

Parameters

<i>disabled</i>	if the timer is supposed to be disabled
-----------------	---

5.8.4.13 setResult

```
bool ScStwTimer::setResult (
    TimerState result ) [slot]
```

Function to assing the result of the race to the timer.

To do this, the timer has to be in [ScStwTimer::WAITING](#) state!

Returns

false if the timer was not in the required state and the result therefore not set, true otherwise

5.8.4.14 `setState`

```
void ScStwTimer::setState (
    TimerState newState ) [protected], [slot]
```

Function to change the state of the timer.

Doing this will emit the `ScStwTimer::stateChanged()` signal (only if the new state differs from the current one)

Parameters

<code><i>newState</i></code>	The new state
------------------------------	---------------

5.8.4.15 `setWantsToBeDisabled`

```
void ScStwTimer::setWantsToBeDisabled (
    bool wantsToBeDisabled ) [protected], [slot]
```

Function to set whether the timer currently wants to be disabled.

Parameters

<code><i>wantsToBeDisabled</i></code>	true or false
---------------------------------------	---------------

5.8.4.16 `start` [1/2]

```
bool ScStwTimer::start ( ) [slot]
```

Function to start the timer.

To do this, the timer has to be in `ScStwTimer::STARTING` state!

Returns

false if the timer was not in the required state and therefore not started, true otherwise

5.8.4.17 `start` [2/2]

```
bool ScStwTimer::start (
    double timeOfStart ) [virtual], [slot]
```

Function to start the timer at a given point in time (present or future)

To do this, the timer has to be in `ScStwTimer::STARTING` state!

Parameters

<i>timeOfStart</i>	the point in time (msecs since epoch) when the timer is supposed to be started
--------------------	--

Returns

false if the timer was not in the required state and therefore not started, true otherwise

5.8.4.18 stop [1/2]

```
bool ScStwTimer::stop ( ) [slot]
```

Function to stop the timer.

To do this, the timer has to be in [ScStwTimer::RUNNING](#) state!

Returns

false if the timer was not in the required state and therefore not stopped, true otherwise

5.8.4.19 stop [2/2]

```
bool ScStwTimer::stop (
    double timeOfStop ) [slot]
```

Function to stop the timer at a given point in time (past or future)

To do this, the timer has to be in [ScStwTimer::RUNNING](#) state!

Parameters

<i>timeOfStop</i>	the point in time (msecs since epoch) when the timer is supposed to be stopped
-------------------	--

Returns

false if the timer was not in the required state and therefore not stopped, true otherwise

5.8.4.20 technicalIncident

```
void ScStwTimer::technicalIncident ( ) [protected], [slot]
```

Function to set the timer into INCIDENT state immediately.

The current state of the timer will be ignored! It can only get out of this state by calling [ScStwTimer::reset](#)

See also

[reset](#)

5.8.4.21 wantsToBeDisabledChanged

```
void ScStwTimer::wantsToBeDisabledChanged (
    ScStwTimer * timer,
    bool wantsToBeDisabled ) [signal]
```

Emitted when the timer wants its state to be changed by the external handler.

Parameters

<i>timer</i>	the timer object
--------------	------------------

5.8.4.22 wildcard

```
bool ScStwTimer::wildcard ( ) [protected], [slot]
```

Function to set the timer into WILDCARD state.

Only works when the timer is in STARTING state.

Returns

false if not in STARTING state

The documentation for this class was generated from the following files:

- /drone/src/ScStwLibraries/headers/scstwtimer.h
- /drone/src/ScStwLibraries/sources/scstwtimer.cpp


```

92     WriteSettingCommand = 3000,
93     ReadSettingCommand = 3001,
94
95     LoginAthleteCommand = 4000,
96     CreateAthleteCommand = 4001,
97     DeleteAthleteCommand = 4002,
98     GetAtheletesCommand = 4003,
99     GetAthleteResultsCommand = 4004,
100
101     UpdateFirmwareCommand = 5000,
102     UpdateSystemTimeCommand = 5001,
103     PairExtensionsCommand = 5002
104 };
105 Q_ENUM(SocketCommand);
106
107 enum StatusCode {
108     Success = 200,
109
110     FirmwareAlreadyUpToDateInfo = 304,
111
112     AccessDeniedError = 401,
113     UpdateSignatureInvalidError = 402,
114     CurrentStateNotVaildForOperationError = 403,
115     CommandNotFoundError = 404,
116     RequiredParameterNotGivenError = 405,
117     TimestampTooSmallError = 406,
118     ClientSessionAlreadyActiveError = 407,
119     NoSessionActiveError = 408,
120     ItemNotFoundError = 409,
121     LastTimerCannotBeDisabledError = 410,
122
123     UpdateFailedError = 500,
124
125     Error = 900,
126     NotConnectedError = 910,
127     TimeoutError = 911,
128     SettingNotAccessibleError = 901,
129     InternalError = 950,
130     InternalErrorTimerOperationFailed = 951,
131     ApiVersionNotSupportedError = 952,
132     CompetitionModeProhibitsThisError = 953,
133     FirmwareUpdateFormatInvalidError = 954,
134     TimersNotReadyError = 501
135 };
136 Q_ENUM(ScStw::StatusCode)
137
138 enum ExtensionType {
139     StartPad,
140     TopPad
141 };
142 Q_ENUM(ExtensionType);
143
144 enum ExtensionState {
145     ExtensionDisconnected = 0,
146     ExtensionConnecting = 1,
147     ExtensionInitialising = 2,
148     ExtensionConnected = 3
149 };
150 Q_ENUM(ExtensionState);
151
152 enum ExtensionBatteryState {
153     BatteryUnknown = -1,
154     BatteryCritical = 0,
155     BatteryWarning = 1,
156     BatteryFine = 2,
157     BatteryCharging = 3,
158     BatteryNotCharging = 4
159 };
160 Q_ENUM(ExtensionBatteryState);
161
162 enum PadState {
163     PadNotPressed = 0,
164     PadPressed = 1
165 };
166 Q_ENUM(PadState);
167
168 static const char* SOCKET_MESSAGE_START_KEY;
169
170 static const char* SOCKET_MESSAGE_END_KEY;
171
172 static SignalKey signalKeyFromInt(int i);
173
174 static SocketCommand socketCommandFromInt(int i);

```

```

213
221     static QString extensionTypeToString(ExtensionType t);
222
223     static int firmwareCompare(QString a, QString b);
224
225     template <typename Enum>
226     static Enum toEnumValue(const int &value, bool *ok)
227     {
228         QMetaEnum enumeration = QMetaEnum::fromType<Enum>();
229         return static_cast<Enum>(enumeration.keyToValue(enumeration.valueToKey(value), ok));
230     }
231
232     ScStw() : QObject(nullptr) {};
233 private:
234 };
235
236 #endif // SCSTW_HPP

```

6.2 scstwlibraries.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWLIBRARIES_H
20 #define SCSTWLIBRARIES_H
21
22 #include <QObject>
23 #ifdef ScStwLibraries_QML
24 #include <QQmlApplicationEngine>
25
26 #ifdef ScStwLibraries_Styling
27 #include "scstwappthememanager.h"
28 #include "scstwapptheme.h"
29 #endif
30
31 #endif
32 #include "scstwtimer.h"
33 #include "scstwtrace.h"
34 #include "scstwsettings.h"
35 #ifdef ScStwLibraries_ClientLibs
36 #include "scstwsetting.h"
37 #include "scstwremoterace.h"
38 #include "scstwclient.h"
39 #include "scstwremotesettings.h"
40 #endif
41
42 class ScStwLibraries : public QObject
43 {
44     Q_OBJECT
45
46 public:
47     static void init();
48
49 #ifdef ScStwLibraries_QML
50 #ifdef ScStwLibraries_Styling
51     static void initStyling(QQmlApplicationEngine *engine);
52 #endif
53 #endif
54
55 private:
56     explicit ScStwLibraries(QObject *parent = nullptr);
57
58 signals:
59
60 };
61
62 #endif // SCSTWLIBRARIES_H

```

6.3 ScStwLibraries_global.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWLIBRARIES_GLOBAL_H
20 #define SCSTWLIBRARIES_GLOBAL_H
21
22 #include <QtCore/qglobal.h>
23
24 #if defined(SCSTWLIBRARIES_LIBRARY)
25 # define SCSTWLIBRARIES_EXPORT Q_DECL_EXPORT
26 #else
27 # define SCSTWLIBRARIES_EXPORT Q_DECL_IMPORT
28 #endif
29
30 #endif // SCSTWCLIENT_GLOBAL_H

```

6.4 scstwrace.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWRACE_H
20 #define SCSTWRACE_H
21
22 #include <QObject>
23 #include <QDebug>
24 #include <QTimer>
25 #include <QEventLoop>
26 #include "scstwtimer.h"
27 #include "scstwsoundplayer.h"
28 #include "scstwsettings.h"
29
30 class ScStwRemoteRace;
31
32 class ScStwRace : public QObject
33 {
34     Q_OBJECT
35     Q_PROPERTY(RaceState state READ getState NOTIFY stateChanged)
36     Q_PROPERTY(QVariantList timers READ getTimerDetailList NOTIFY timersChanged)
37     Q_PROPERTY(QVariantMap currentStartDelay READ getCurrentStartDelay NOTIFY currentStartDelayChanged)
38     Q_PROPERTY(bool isReadyForNextState READ getIsReadyForNextState NOTIFY isReadyForNextStateChanged)
39     Q_PROPERTY(bool competitionMode READ getCompetitionMode NOTIFY competitionModeChanged)
40     Q_PROPERTY(bool readySoundEnabled READ getReadySoundEnabled NOTIFY readySoundEnabledChanged)
41     Q_PROPERTY(QVariantMap details READ getDetails NOTIFY detailsChanged)
42     Q_PROPERTY(ScStwSettings* settings READ getSettings WRITE setSettings NOTIFY settingsChanged)
43     Q_PROPERTY(bool autoRefreshTimerText READ getAutoRefreshTimerText WRITE setAutoRefreshTimerText
44     NOTIFY autoRefreshTimerTextChanged)
45
46 public:
47     explicit ScStwRace(QObject *parent = nullptr);
48     explicit ScStwRace(ScStwSettings *settings, QObject *parent = nullptr);

```

```

68
69     friend class ScStwRemoteRace;
70
71     enum RaceState { IDLE, PREPAIRING, WAITING, STARTING, RUNNING, STOPPED, INCIDENT };
72     Q_ENUM(RaceState)
73
74 protected:
75     QList<ScStwTimer *> timers;
76     void setState(RaceState newState);
77
78 private:
79     RaceState state;
80
81     QTimer *startDelayTimer;
82     QTimer *timerTextRefreshTimer;
83     QEventLoop *startWaitLoop;
84
85     // sounds
86     ScStwSoundPlayer * soundPlayer;
87
88     // settings
89     ScStwSettings *settings;
90     bool competitionMode;
91     bool autoRefreshTimerText;
92
93     enum LoopExitTypes {
94         LoopAutomaticExit = 0,
95         LoopReadyStateChangeExit = 1,
96         LoopManualExit = 2,
97         LoopCancelExit = 3
98     };
99
100
101 public slots:
102     virtual ScStw::StatusCode start (bool asynchronous = true);
103
104     virtual ScStw::StatusCode stop();
105
106     virtual ScStw::StatusCode reset();
107     virtual ScStw::StatusCode cancel();
108
109     virtual ScStw::StatusCode setTimerDisabled(int id, bool disabled);
110
111     Q_INVOKABLE virtual bool addTimer(ScStwTimer *timer);
112
113     // getters
114     RaceState getState();
115     virtual QVariantMap getCurrentStartDelay();
116     QList<ScStwTimer*> getTimers();
117     QVariantList getTimerDetailList();
118     QVariantMap getDetails();
119     bool getCompetitionMode();
120     virtual bool getReadySoundEnabled();
121
122     ScStwSettings* getSettings();
123     void setSettings(ScStwSettings* settings);
124
125     bool getAutoRefreshTimerText();
126     void setAutoRefreshTimerText(bool autoRefresh);
127
128 protected slots:
129
130 private slots:
131     void handleTimerStateChange(ScStwTimer::TimerState newState);
132
133     void handleTimerStop();
134     void handleFalseStart();
135
136     void handleTimerWantsToBeDisabledChange(ScStwTimer* timer, bool wantsToBeDisabled);
137     bool playSoundsAndStartTimers();
138     ScStwSoundPlayer::PlayResult doDelayAndSoundOfCurrentStartState(double *timeOfSoundPlaybackStart =
139     nullptr);
140     void technicalIncident();
141     ScStw::StatusCode setTimerDisabled(ScStwTimer* timer, bool disabled);
142
143     virtual void refreshCompetitionMode();
144
145     double getSoundVolume();
146     ScStwSoundPlayer::StartSound getSoundForState(ScStwRace::RaceState state);
147     bool getSoundEnabledSetting(ScStwSoundPlayer::StartSound sound);
148     int getSoundDelaySetting(ScStwSoundPlayer::StartSound sound);
149
150     bool isStarting();
151     virtual bool getIsReadyForNextState();
152     void handleTimerReadyStateChange(ScStwTimer::ReadyState readyState);

```

```

173
174 signals:
175     void startTimers();
176     void stopTimers(int type);
177     void resetTimers();
178     void stateChanged(RaceState state);
179     void currentStartDelayChanged();
180     void timersChanged();
181     void isReadyForNextStateChanged();
182     void detailsChanged();
183     void competitionModeChanged();
184     void readySoundEnabledChanged();
185     void settingsChanged();
186     void autoRefreshTimerTextChanged();
187
188 };
189
190 #endif // SCSTWRACE_H

```

6.5 scstwsetting.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWQMLSETTING_H
20 #define SCSTWQMLSETTING_H
21
22 #include <QObject>
23 #include <QVariant>
24
25 class ScStwSettings;
26
27 class ScStwSetting : public QObject
28 {
29     Q_OBJECT
30     Q_PROPERTY(QVariant value READ getValue WRITE setValue NOTIFY valueChanged)
31     Q_PROPERTY(QVariant readonlyValue READ getValue NOTIFY valueChanged)
32
33 protected:
34     explicit ScStwSetting(int key, int keyLevel, ScStwSettings*scStwSettings, QObject *parent);
35
36     friend class ScStwSettings;
37
38     int key;
39     int keyLevel;
40     bool hasToReload;
41
42 private:
43     QVariant valueCache;
44     ScStwSettings *scStwSettings;
45
46 public slots:
47     QVariant getValue();
48     void setValue(QVariant value);
49
50 protected slots:
51     void handleSettingChange(int key, int keyLevel, QVariant value);
52
53 signals:
54     void valueChanged();
55
56 };
57
58 #endif // SCSTWQMLSETTING_H

```

6.6 scstwsettings.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWSETTINGS_H
20 #define SCSTWSETTINGS_H
21
22 #include <QObject>
23 #include <QVariant>
24 #include <QMetaEnum>
25 #include <QtDebug>
26 #include <QFile>
27 #include <QStandardPaths>
28 #include <QJsonDocument>
29 #include <ScStw.hpp>
30 #include <QDir>
31 #include <scstwsetting.h>
32
33 class ScStwSettings : public QObject
34 {
35     Q_OBJECT
36 public:
37     explicit ScStwSettings(QObject *parent = nullptr, bool overwriteFileOnErrors = true);
38
39     typedef QString(*keyToStringConverter)(int);
40     typedef QVariant::Type(*keyToTypeConverter)(int);
41
42     enum BaseStationSetting {
43         InvalidSetting = -1,
44         ReadySoundEnableSetting,
45         ReadySoundDelaySetting,
46         AtYourMarksSoundEnableSetting,
47         AtYourMarksSoundDelaySetting,
48         SoundVolumeSetting,
49         CompetitionModeSetting
50     };
51     Q_ENUM(BaseStationSetting)
52
53     enum KeyLevelEnum {
54         KeyLevel = 0
55     };
56     Q_ENUM(KeyLevelEnum)
57
58     virtual QVariant readSetting(BaseStationSetting key);
59     Q_INVOKABLE virtual QVariant readSetting(int key, int keyLevel);
60     virtual bool writeSetting(BaseStationSetting key, QVariant value);
61     Q_INVOKABLE virtual bool writeSetting(int key, int keyLevel, QVariant value);
62     virtual bool setDefaultSetting(BaseStationSetting key, QVariant defaultValue);
63     Q_INVOKABLE virtual bool setDefaultSetting(int key, int keyLevel, QVariant defaultValue);
64
65     Q_INVOKABLE ScStwSetting * getSetting(int key, int keyLevel);
66
67     static BaseStationSetting keyFromInt(int i) {
68         QMetaEnum enumeration = QMetaEnum::fromType<BaseStationSetting>();
69         return static_cast<BaseStationSetting>(enumeration.keyToValue(enumeration.valueToKey(i)));
70     }
71
72     static QString keyToString(int key) {
73         return QMetaEnum::fromType<BaseStationSetting>().valueToKey(key);
74     }
75
76     static QVariant::Type keyToType(int key) {
77         QMap<BaseStationSetting, QVariant::Type> types = {
78             {ReadySoundEnableSetting, QVariant::Bool},
79             {ReadySoundDelaySetting, QVariant::Double},
80             {AtYourMarksSoundEnableSetting, QVariant::Bool},
81             {AtYourMarksSoundDelaySetting, QVariant::Double},
82             {SoundVolumeSetting, QVariant::Double},
83             {CompetitionModeSetting, QVariant::Bool}
84         };
85     }
86
87 };
88
89 #endif

```

```

94     if(types.contains(BaseStationSetting(key)))
95         return types[BaseStationSetting(key)];
96
97     return QVariant::Invalid;
98 }
99
100 protected:
101     virtual QVariant readSetting(QString key, int keyInt = -1, int keyLevel = -1);
102     virtual bool writeSetting(QString key, QVariant value, int keyInt = -1,int keyLevel = -1);
103     virtual bool setDefaultSetting(QString key, QVariant defaultValue, int keyInt,int keyLevel = -1);
104     bool registerKeyLevelConverters(int keyLevel, keyToStringConverter, keyToTypeConverter);
105
106 private:
107     bool fileIsReadOnly;
108
109     QFile * settingsFile;
110     QVariantMap settingsCache;
111
112     bool loadSettingsFromFile();
113
114     QMap<int, keyToStringConverter> keyToStringConverters;
115     QMap<int, keyToTypeConverter> keyToTypeConverters;
116     QMap<int, QMap<int, ScStwSetting*>> internalSettingHandlers;
117
118 private slots:
119     bool writeSettingsToFile();
120
121 signals:
122     void settingChanged(int key, int keyLevel, QVariant value);
123
124 };
125
126 #endif // SCSTWSETTINGS_H

```

6.7 scstwsoundplayer.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWSTARTSOUNDPLAYER_H
20 #define SCSTWSTARTSOUNDPLAYER_H
21
22 #include <QObject>
23 #include <QFile>
24 #include <QAudioOutput>
25 #include <QDebug>
26 #include <QEventLoop>
27 #include <QTimer>
28 #include <QDateTime>
29 #include <QSoundEffect>
30 #include <QAudioDeviceInfo>
31
32 #ifdef ScStwLibraries_Raspi
33 #include <alsa/asoundlib.h>
34 #endif
35
36
37
38
39 class ScStwSoundPlayer : public QObject
40 {
41     Q_OBJECT
42 public:
43     explicit ScStwSoundPlayer(QObject *parent = nullptr);
44
45     enum StartSound {
46         None = -1,
47         AtYourMarks = 0,
48         Ready = 1,
49         Start = 2,
50         FalseStart = 3

```

```

55     };
56
57     enum PlayResult {
58         Success = 0,
59         Cancelled = -1,
60         Error = -2
61     };
62
63 private:
64     bool _setSoundVolume(double volume);
65
66     void _initializeSondEffect();
67
68     QMap<StartSound, QVariantMap> soundFiles;
69
70     QSoundEffect *soundEffect;
71
72     QAudioDeviceInfo *_audioOutputDevice;
73
74     QEventLoop *waitLoop;
75
76     QTimer *waitTimer;
77
78     StartSound currentlyPlayingSound;
79
80     double playingStartedAt;
81
82 public slots:
83
84     ScStwSoundPlayer::PlayResult play(StartSound sound, double volume, double *timeOfStart = nullptr);
85
86     ScStwSoundPlayer::PlayResult waitForSoundFinish(double *timeOfStop = nullptr);
87
88     bool cancel();
89
90     bool isPlaying();
91
92 private slots:
93
94 signals:
95     void playbackStarted();
96 };
97
98 #endif // SCSTWSTARTSOUNDPLAYER_H

```

6.8 scstwstartsoundplayer.h

```

1 #ifndef SCSTWSTARTSOUNDPLAYER_H
2 #define SCSTWSTARTSOUNDPLAYER_H
3
4 #include <QObject>
5 #include <QFile>
6 #include <QAudioOutput>
7 #include <QDebug>
8 #include <QEventLoop>
9 #include <QTimer>
10 #include <QDateTime>
11
12 class ScStwStartSoundPlayer : public QObject
13 {
14     Q_OBJECT
15 public:
16     explicit ScStwStartSoundPlayer(QObject *parent = nullptr);
17
18 private:
19     QFile *startSoundFile;
20     QAudioOutput *audioOutput;
21     QEventLoop *waitLoop;
22
23 public slots:
24     bool play(double volume, double *timeOfStop = nullptr);
25     //int interrupt();
26
27 private slots:
28     void handleStateChanged(QAudio::State newState);
29
30 signals:
31
32 };
33
34 #endif // SCSTWSTARTSOUNDPLAYER_H

```

6.9 scstwtimer.h

```

1 /*****
2 ** ScStw Libraries
3 ** Copyright (C) 2020 Itsblue development
4 **
5 ** This program is free software: you can redistribute it and/or modify
6 ** it under the terms of the GNU General Public License as published by
7 ** the Free Software Foundation, either version 3 of the License, or
8 ** (at your option) any later version.
9 **
10 ** This program is distributed in the hope that it will be useful,
11 ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12 ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 ** GNU General Public License for more details.
14 **
15 ** You should have received a copy of the GNU General Public License
16 ** along with this program. If not, see <http://www.gnu.org/licenses/>.
17 *****/
18
19 #ifndef SCSTWTIMER_H
20 #define SCSTWTIMER_H
21
22 #include <QObject>
23 #include <QDateTime>
24 #include <QDebug>
25 #include <QTimer>
26 #include "ScStw.hpp"
27
28 class ScStwRace;
29 class ScStwRemoteRace;
30
64 class ScStwTimer : public QObject
65 {
66     Q_OBJECT
67 public:
68
69     explicit ScStwTimer(QObject *parent = nullptr);
70
71
77     explicit ScStwTimer(QString letter, QObject *parent = nullptr);
78
79     friend class ScStwRace;
80
84     enum TimerState {
85         IDLE,
86         STARTING,
87         RUNNING,
88         WAITING,
89         WON,
90         LOST,
91         FAILING,
92         WILDCARD,
93         FAILED,
94         CANCELLED,
95         INCIDENT,
96         DISABLED
97     };
98     Q_ENUM(TimerState);
99
103     enum ReadyState {
104         IsReady = 0,
105         NotInIdleState,
106         IsDisabled,
107         ExtensionIsNotConnected,
108         ExtensionBatteryIsCritical,
109         ClimberIsNotReady
110     };
111     Q_ENUM(ReadyState)
112
113 protected:
117     TimerState state;
118
122     double startTime;
123
127     double stopTime;
128
132     double reactionTime;
133
137     QString letter;
142     bool wantsToBeDisabled;
143
144 public slots:
145
153     bool start();
154
163     virtual bool start(double timeOfStart);

```

```
164
172     bool cancel();
173
181     bool stop();
182
192     bool stop(double timeOfStop);
193
201     bool setResult(TimerState);
202
210     virtual bool reset();
211
212     // -- helper functions --
218     TimerState getState();
219
227     double getCurrentTime();
228
233     double getReactionTime();
234
239     QString getText();
240
245     QString getLetter();
246
257     void setDisabled(bool disabled);
258
263     bool getWantsToBeDisabled();
264
269     virtual ScStwTimer::ReadyState getReadyState();
270
271     bool isRunning();
272
273     bool isDisabled();
274
275 protected slots:
276
281     void handleClimberStart(double timeOfStart);
282
290     void setState(TimerState newState);
291
296     void setWantsToBeDisabled(bool wantsToBeDisabled);
297
305     void technicalIncident();
306
314     bool wildcard();
315
316
317 signals:
321     void stateChanged(TimerState state);
322
326     void reactionTimeChanged();
327
332     void wantsToBeDisabledChanged(ScStwTimer* timer, bool wantsToBeDisabled);
333
338     void readyStateChanged(ReadyState readyState);
339
340 };
341
342 #endif // SCSTWTIMER_H
```


Index

[/drone/src/ScStwLibraries/headers/ScStw.hpp](#), 29
[/drone/src/ScStwLibraries/headers/ScStwLibraries_global.h](#), 32
[/drone/src/ScStwLibraries/headers/scstwlibraries.h](#), 31
[/drone/src/ScStwLibraries/headers/scstwrace.h](#), 32
[/drone/src/ScStwLibraries/headers/scstwsetting.h](#), 34
[/drone/src/ScStwLibraries/headers/scstwsettings.h](#), 35
[/drone/src/ScStwLibraries/headers/scstwsoundplayer.h](#), 36
[/drone/src/ScStwLibraries/headers/scstwstartsoundplayer.h](#), 37
[/drone/src/ScStwLibraries/headers/scstwtimer.h](#), 38

BaseStationSetting
 ScStwSettings, 14

cancel
 ScStwSoundPlayer, 16
 ScStwTimer, 22

CANCELLED
 ScStwTimer, 21

ClimberIsNotReady
 ScStwTimer, 21

DISABLED
 ScStwTimer, 21

ExtensionBatteryIsCritical
 ScStwTimer, 21

ExtensionIsNotConnected
 ScStwTimer, 21

FAILED
 ScStwTimer, 21

FAILING
 ScStwTimer, 21

getCurrentTime
 ScStwTimer, 22

getLetter
 ScStwTimer, 22

getReactionTime
 ScStwTimer, 23

getReadyState
 ScStwTimer, 23

getState
 ScStwTimer, 23

getText
 ScStwTimer, 23

getWantsToBeDisabled
 ScStwTimer, 24

handleClimberStart
 ScStwTimer, 24

IDLE
 ScStwTimer, 21

INCIDENT
 ScStwTimer, 21

IsDisabled
 ScStwTimer, 21

IsReady
 ScStwTimer, 21

LOST
 ScStwTimer, 21

NotInIdleState
 ScStwTimer, 21

play
 ScStwSoundPlayer, 17

ReadyState
 ScStwTimer, 21

readyStateChanged
 ScStwTimer, 24

reset
 ScStwRace, 11
 ScStwTimer, 25

RUNNING
 ScStwTimer, 21

ScStw, 9
ScStwLibraries, 9
ScStwRace, 9
 reset, 11
 start, 12
 stop, 12
ScStwSetting, 12
ScStwSettings, 13
 BaseStationSetting, 14
ScStwSoundPlayer, 15
 cancel, 16
 play, 17
 ScStwSoundPlayer, 16
 waitForSoundFinish, 17
ScStwStartSoundPlayer, 18
ScStwTimer, 18
 cancel, 22
 CANCELLED, 21
 ClimberIsNotReady, 21
 DISABLED, 21

- ExtensionBatteryIsCritical, [21](#)
- ExtensionIsNotConnected, [21](#)
- FAILED, [21](#)
- FAILING, [21](#)
- getCurrentTime, [22](#)
- getLetter, [22](#)
- getReactionTime, [23](#)
- getReadyState, [23](#)
- getState, [23](#)
- getText, [23](#)
- getWantsToBeDisabled, [24](#)
- handleClimberStart, [24](#)
- IDLE, [21](#)
- INCIDENT, [21](#)
- IsDisabled, [21](#)
- IsReady, [21](#)
- LOST, [21](#)
- NotInIdleState, [21](#)
- ReadyState, [21](#)
- readyStateChanged, [24](#)
- reset, [25](#)
- RUNNING, [21](#)
- ScStwTimer, [21](#)
- setDisabled, [25](#)
- setResult, [25](#)
- setState, [25](#)
- setWantsToBeDisabled, [26](#)
- start, [26](#)
- STARTING, [21](#)
- stop, [27](#)
- technicalIncident, [27](#)
- TimerState, [21](#)
- WAITING, [21](#)
- wantsToBeDisabledChanged, [28](#)
- WILDCARD, [21](#)
- wildcard, [28](#)
- WON, [21](#)

- setDisabled
 - ScStwTimer, [25](#)
- setResult
 - ScStwTimer, [25](#)
- setState
 - ScStwTimer, [25](#)
- setWantsToBeDisabled
 - ScStwTimer, [26](#)
- start
 - ScStwRace, [12](#)
 - ScStwTimer, [26](#)
- STARTING
 - ScStwTimer, [21](#)
- stop
 - ScStwRace, [12](#)
 - ScStwTimer, [27](#)
- technicalIncident
 - ScStwTimer, [27](#)
- TimerState
 - ScStwTimer, [21](#)

- waitForSoundFinish
 - ScStwSoundPlayer, [17](#)
- WAITING
 - ScStwTimer, [21](#)
- wantsToBeDisabledChanged
 - ScStwTimer, [28](#)
- WILDCARD
 - ScStwTimer, [21](#)
- wildcard
 - ScStwTimer, [28](#)
- WON
 - ScStwTimer, [21](#)