iQ-Trigger API

C++

Version 3.0.0

Documentation

Created on Fri Dec 15 2017

Contents

1 iQ-Trigger API introduction

The iQ-Trigger API offers functions for controlling the iQ-Trigger USB control box. Besides the control of the iQ--Trigger and iQ-Trigger-T the behaviour of the trigger output port of the control box can also be configured. The trigger output switches between short-circuit and open-circuit and may be used for other devices by Image Engineering GmbH & Co. KG, like the LED-Panel. Since verision 3.0.0 creation of sequences is possible.

Sequences: The API offers the possibility to create sequences of iQ-trigger releases. For each connected iQ--Trigger USB box individual sequences with number of iterations, period and release time can be set. There is also a counter and time functionality, which informs the user about number of current iterations, elapsed and total time.

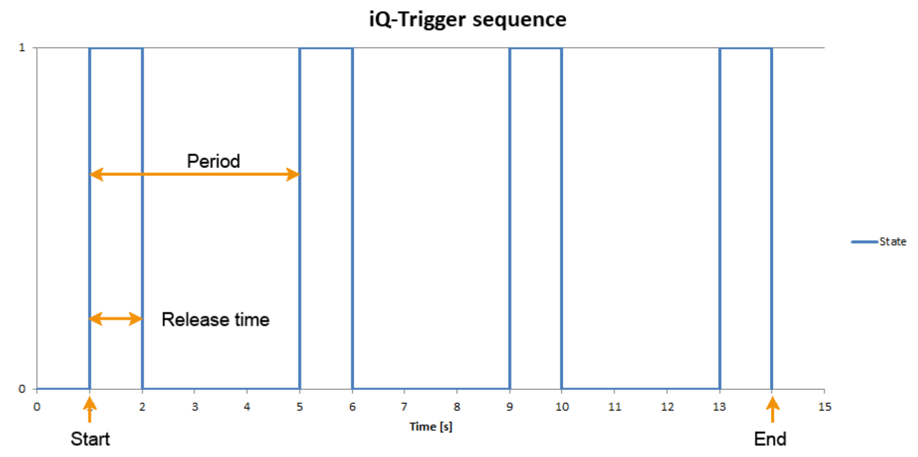

Figure 1: Example of a sequence with a period time of four seconds, one second release time and 4 iterations. The duty cycle is 25%.

Simultaneous usage: In order to control several iQ-Trigger devices simultaneously, it is possible to set the same sequence settings to all controllers.

Attention

The API controls the USB box and has no knowledge about the connected device, iQ-Trigger or iQ-Trigger--T. Both devices have different characteristics concerning continuous operation. While iQ-trigger-T may be switched very fast and may stay in released mode for a long period, the mechanical finger iQ-Trigger is limited due to mechanical construction:

- Maximum time in released state: 60s
- Shortest release time: 100ms
- For continuous operation the duty cycle must not be longer than 50% (duty cycle = the ratio of release time to period) The mechanical finger may heat up or may even be destroyed if operated for a long time in released state without interruption.

Trigger output: The trigger output port of the USB box is directly dependent on the state of the iQ-Trigger. See also function description in iQ::Trigger::TriggerController::setTriggerOutputBehaviour.

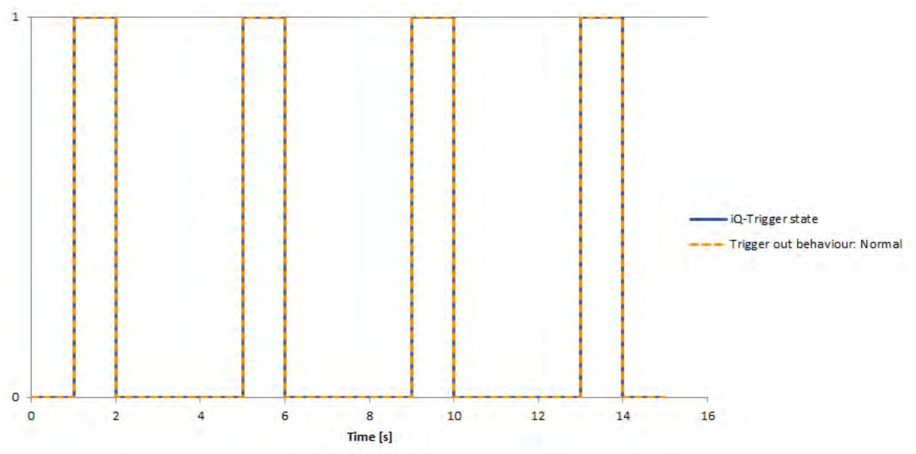

Figure 2: The bahviour of the trigger output with normal behaviour.

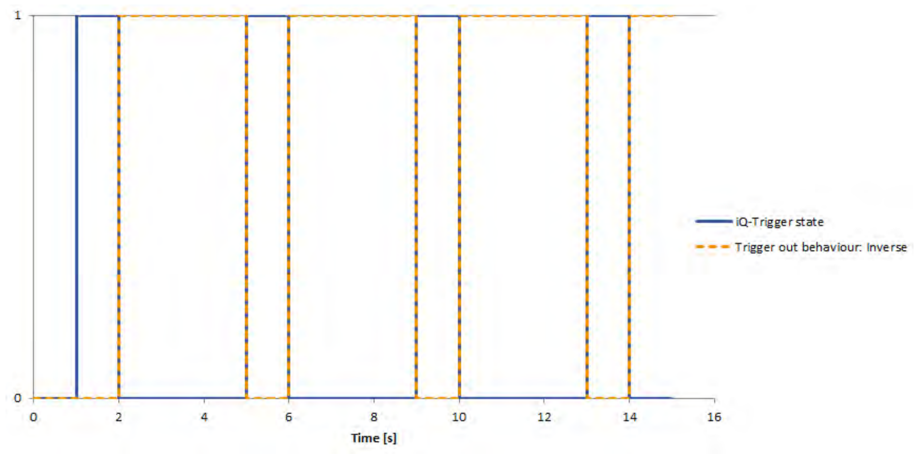

Figure 3: The bahviour of the trigger output with inverse behaviour.

Requirements:

The API is written in C++ and available for the following development environments:

- Microsoft Visual C++ (MSVC) 2012 64Bit
- Microsoft Visual C++ (MSVC) 2012 32Bit
- Microsoft Visual C++ (MSVC) 2015 64Bit
- Microsoft Visual C++ (MSVC) 2015 32Bit

The API uses open source software:

- libusb (www.libusb.org) is an open source project, the code is licensed under the GNU Lesser General Public License version 2.1 or later (www.gnu.org/licenses/old-licenses/lgpl-2.1.html).

2 Code examples for using iQ-Trigger API

The example page contain code examples for the usage of the iQ-Trigger API.

Single use:

```
#include "triggercontroller.h"
#include "error.h"

using namespace iQ::Trigger;
int error;
std::map<std::string, TriggerController *> allTriggerDevices =
      TriggerController::createObjects(error);
if(error != Error::None)
 return 0;
std::vector<std::string> serials = TriggerController::getAllSerialNumbers
      (allTriggerDevices);
std::string serial1 = serials[0];
TriggerController* trigger = allTriggerDevices[serial1];
trigger->setTriggerOutputBehaviour(NORMAL, error);
trigger->setState(RELEASED, error);

// Deleting objects closes the USB connection
delete trigger;
```

Sequential use:

```
// Consider same implementation as above
trigger->setSequence(1000, 4, 1, error);

// Calculate total amount of time for sequence:
double totalTime = trigger->getTotalTime();

trigger->startSequence(error);
// Ask for some information during play back:
int counter = trigger->getCounter();
double time = trigger->getElapsedTime();
```

3 Changelog

Version 3.0.0 (released 15-Dec-2017)

- Added sequence functionality to API and GUI. Includes release counting and elapsed time recording.
- Added functionality for grouping multiple iQ-Trigger USB boxes in GUI.
- API changes in detail:
  - Added new functions:

```
std::string getSerial();
setSequence(int nIterations,
                double periodTimeSec,
                double releaseTimeSec,
                int& errorCode = default_error);
startSequence(int& errorCode = default_error);
stopSequence();
resetCounter();
getCounter();
getNIterations();
getPeriodTimeSec();
getReleaseTimeSec();
getTotalTime();
getCurrentTime();
getElapsedTime();
iQ::Trigger::SequenceState getSequenceState();
setTriggerOutputBehaviour(iQ::Trigger::TriggerOutputBehaviour state,
                            int& errorCode = default_error);
iQ::Trigger::TriggerOutputBehaviour getTriggerOutputBehaviour();
```

  - Renamed enumerations:

```
TriggerState::RELEASED -> TriggerState::IDLE
TriggerState::PRESSED -> TriggerState::RELEASED
```

  - Added new enumeration TriggerOutputBehaviour for the behaviour of the trigger out port.

Version 2.0.0 (released 23-Jan-2017)

- Major change: Renamed Digitus API to iQ-Trigger API due to changes in product naming.

Version 1.0.4 (released 10-May-2016)

- Bug fix: Unstable behaviour in GUI application when calling "Find Digitus devices" from menu.

Version 1.0.3 (released 14-Dec-2015)

- Bug fixes.
- Changed USB controller driver to libusb (WinUSB).
- Drivers, DLL libraries and executables are now signed with an Image Engineering security certificate.

Version 1.0.2 (released 9-Oct-2015)

- Bug fixes

Version 1.0.1 (released 17-Jun-2015)

- Bug fixes.

Version 1.0.0 (released 30-Mar-2015)

- Initial release of Digitus API

4 Namespace Index

4.1 Namespace List

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

5 Class Index

5.1 Class List

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

6 Namespace Documentation

6.1 iQ::Trigger Namespace Reference

Classes

- struct TriggerError
- class TriggerController
- class Version

Enumerations

- enum ErrorCodes {
  NONE = 0, INVALID_PARAMETER = 4000, DEVICE_DISCONNECTED = 4001, USB_CONNECTION_FAILED = 4002,
  USB_COMMUNICATION_ERROR = 4003 }
- enum TriggerState { IDLE = 0, RELEASED = 1, UNDEFINED = 2 }
- enum TriggerOutputBehaviour { NORMAL = 0, INVERSE = 1 }
- enum SequenceState { STOP = 0, PLAY = 1 }

6.1.1 Detailed Description

This namespace is used in the Image Engineering iQ-Trigger API.

6.1.2 Enumeration Type Documentation

6.1.2.1 enum iQ::Trigger::ErrorCodes

The enumeration values are used to address the error codes used in this API.

Enumerator

NONE No error occured.
INVALID_PARAMETER Invalid input parameter.
DEVICE_DISCONNECTED No iQ-Trigger box connected to PC.
USB_CONNECTION_FAILED The connection to USB box could not be established.
USB_COMMUNICATION_ERROR Communication with USB box failed.

6.1.2.2 enum iQ::Trigger::SequenceState

This enumeration defines the play back state of a sequence. It is returned by getSequenceState();

Enumerator

STOP The sequence is stopped. Counter and elapsed time of a sequence are stored and continued if sequence is started again.
PLAY The sequence is played back.

6.1.2.3 enum iQ::Trigger::TriggerOutputBehaviour

This enumeration defines values for the behaviour of the trigger output port of the USB box. The state of the output directly depends on the iQ-Trigger state.

Enumerator

NORMAL If the trigger state of the device is set to released state, the iQ-Trigger output port is switched to active and vice versa. Thus both ports for iQ-trigger and trigger output have the same state, this is the normal behaviour.
INVERSE If the trigger state of the device is set to released state, the iQ-Trigger output port is switched to inactive and vice versa. Thus both ports for iQ-trigger and trigger output have the opposite state, this is the inverse behaviour.

6.1.2.4 enum iQ::Trigger::TriggerState

This enumeration defines values for the state of a iQ-Trigger device conncted to the iQ-Trigger port of the USB box.

Enumerator

IDLE If state is set to idle, the iQ-Trigger is switched to inactive.
RELEASED If state is set to released, the iQ-Trigger is switched to active.
UNDEFINED This state is returned by getState() in case of communication error. Using this variable in setState() has no influence.

7 Class Documentation

7.1 iQ::Trigger::TriggerController Class Reference

Public Member Functions

- ∼TriggerController ()

  iQ::Trigger::TriggerController::∼TriggerController The destructor closes the USB connection to the connected iQ--Trigger box.

- void setState (iQ::Trigger::TriggerState state, int &errorCode=default_error)

  iQ::Trigger::TriggerController::setState
  Use iQ::Trigger::RELEASE or iQ::Trigger::IDLE to set the iQ-Trigger to the desired state.

- iQ::Trigger::TriggerState getState (int &errorCode=default_error)

iQ::Trigger::TriggerController::getState
Returns the current state of the iQ-Trigger.
If iQ::Trigger::UNDEFINED is returned, an error occurred during USB communication.

- void toggleState (iQ::Trigger::TriggerState state, int msec, int &errorCode=default_error)

  iQ::Trigger::TriggerController::toggleState
  Toggles the state of the iQ-Trigger for a specified amount of time. The iQ-Trigger is set to given state regardless of the current state. The time can not be less than 100 milliseconds. Otherwise function does nothing and errorCode is iQ::Trigger::Error::InvalidParameter.

- std::string getSerial ()

  iQ::Trigger::TriggerController::getSerial

- void setSequence (int nIterations, double periodTimeSec, double releaseTimeSec, int &errorCode=default_-error)

  iQ::Trigger::TriggerController::setSequence Sets the required parameters for the sequence.

- void startSequence (int &errorCode=default_error)

  iQ::Trigger::TriggerController::startSequence Starts the sequence.

- void stopSequence ()

  iQ::Trigger::TriggerController::stopSequence Stops the sequence and pauses counter and elapsed time measure-ment.

- void resetCounter ()

  iQ::Trigger::TriggerController::resetCounter Resets the counter to 0 and also resets the elapsed time.

- int getCounter ()

  iQ::Trigger::TriggerController::getCounter

- int getNIterations ()

  iQ::Trigger::TriggerController::getNIterations

- double getPeriodTimeSec ()

  iQ::Trigger::TriggerController::getPeriodTimeSec

- double getReleaseTimeSec ()

  iQ::Trigger::TriggerController::getReleaseTimeSec

- double getTotalTime ()

  iQ::Trigger::TriggerController::getTotalTime

- int getCurrentTime ()

  iQ::Trigger::TriggerController::getCurrentTime

- int getElapsedTime ()

  iQ::Trigger::TriggerController::getElapsedtime

- iQ::Trigger::SequenceState getSequenceState ()

  iQ::Trigger::TriggerController::getSequenceState Returns the current state of the sequence.

- void setTriggerOutputBehaviour (iQ::Trigger::TriggerOutputBehaviour state, int &errorCode=default_error)

  iQ::Trigger::TriggerController::setTriggerOutputBehaviour
  Sets the behaviour for the trigger output port of the USB box. The trigger output depends on the state of the iQ-Trigger. May be iQ::Trigger::NORMAL or iQ::Trigger::INVERSE.

- iQ::Trigger::TriggerOutputBehaviour getTriggerOutputBehaviour ()

  iQ::Trigger::TriggerController::getTriggerOutputBehaviour
  Returns the current trigger output behaviour.

Static Public Member Functions

- static std::map< std::string,
  TriggerController ∗ > createObjects (int &errorCode=default_error)

  iQ::Trigger::TriggerController::createObjects
  Creates controller objects for any iQ-Trigger box connected to PC. A controller object contained in the map is adressed with the serial number of the iQ-Trigger box. E.g.:
- static std::vector< std::string > getAllSerialNumbers (std::map< std::string, TriggerController ∗ > objects)

  iQ::Trigger::TriggerController::getAllSerialNumbers
  Returns the serial numbers of all connected iQ-Trigger USB boxes. A serial number can be used to address the corresponding controller object:
```
using namespace iQ::Trigger;
std::map<std::string, TriggerController *> allTriggerDevices =
      TriggerController::createObjects(error);
if(error != iQ::Trigger::None)
 return;
std::vector<std::string> serials = TriggerController::getAllSerialNumbers
      (allTriggerDevices);
std::string serial1 = serials[0];
TriggerController* dig1 = allTriggerDevices[serial1];
```
- static std::string getErrorMessage (int &error)

  iQ::Trigger::TriggerController::getErrorMessage
  Returns a corresponding message for the error code.

7.1.1 Member Function Documentation

7.1.1.1 std::map< std::string, iQ::Trigger::TriggerController ∗ > iQ::Trigger::TriggerController::createObjects (int & *errorCode* = default_error) [static]

iQ::Trigger::TriggerController::createObjects

Creates controller objects for any iQ-Trigger box connected to PC. A controller object contained in the map is adressed with the serial number of the iQ-Trigger box. E.g.:

```
std::map<std::string, iQ::Trigger::TriggerController *> allTriggers =
      iQ::Trigger::TriggerController::createObjects(error);
iQ::Trigger::TriggerController* dig1 = allTriggers["DG30001"];
```

See also iQ::Trigger::TriggerController::getAllSerialNumbers(). Here the serial numbers of all conncted USB boxes is returned.

Parameters

errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::get-ErrorMessage().

Returns

A std::map with iQ-Trigger serial numbers as keys and pointers to controller objects as values. In case of an error an empty map is returned.

7.1.1.2 std::vector< std::string > iQ::Trigger::TriggerController::getAllSerialNumbers (std::map< std::string, TriggerController * > *objects*) [static]

iQ::Trigger::TriggerController::getAllSerialNumbers

Returns the serial numbers of all connected iQ-Trigger USB boxes. A serial number can be used to address the corresponding controller object:

```
using namespace iQ::Trigger;
std::map<std::string, TriggerController *> allTriggerDevices =
      TriggerController::createObjects(error);
if(error != iQ::Trigger::None)
 return;
std::vector<std::string> serials = TriggerController::getAllSerialNumbers
      (allTriggerDevices);
std::string serial1 = serials[0];
TriggerController* dig1 = allTriggerDevices[serial1];
```

Parameters

objects	A std::map with the controller objects received from iQ::Trigger::TriggerController::create-Objects().

Returns

A std::vector with the serial numbers of all connected iQ-Trigger boxes.

7.1.1.3 int iQ::Trigger::TriggerController::getCounter ( )

iQ::Trigger::TriggerController::getCounter

Returns

The current number of iQ-Trigger periods.

7.1.1.4 int iQ::Trigger::TriggerController::getCurrentTime ( )

iQ::Trigger::TriggerController::getCurrentTime

Returns

The current time in seconds since start of sequence.

7.1.1.5 int iQ::Trigger::TriggerController::getElapsedTime ( )

iQ::Trigger::TriggerController::getElapsedtime

Returns

The elapsed time in seconds after stopping the sequence.

7.1.1.6 std::string iQ::Trigger::TriggerController::getErrorMessage (int & *error*) [static]

iQ::Trigger::TriggerController::getErrorMessage

Returns a corresponding message for the error code.

Parameters

error	The error code received from function call.

Returns

A string representation of the error code.

7.1.1.7 int iQ::Trigger::TriggerController::getNIterations ( )

iQ::Trigger::TriggerController::getNIterations

Returns

The number of iterations.

7.1.1.8 double iQ::Trigger::TriggerController::getPeriodTimeSec ( )

iQ::Trigger::TriggerController::getPeriodTimeSec

Returns

periodTime The time for a period.

7.1.1.9 double iQ::Trigger::TriggerController::getReleaseTimeSec ( )

iQ::Trigger::TriggerController::getReleaseTimeSec

Returns

releaseTime The time for an iQ-Trigger release.

7.1.1.10 iQ::Trigger::SequenceState iQ::Trigger::TriggerController::getSequenceState ( )

iQ::Trigger::TriggerController::getSequenceState Returns the current state of the sequence.

Returns

- iQ::Trigger::PLAY or
- iQ::Trigger::STOP

7.1.1.11 std::string iQ::Trigger::TriggerController::getSerial ( )

iQ::Trigger::TriggerController::getSerial

Returns

The serial number of the corresponding USB box.

7.1.1.12 iQ::Trigger::TriggerState iQ::Trigger::TriggerController::getState (int & *errorCode* = `default_error`)

iQ::Trigger::TriggerController::getState

Returns the current state of the iQ-Trigger.

If iQ::Trigger::UNDEFINED is returned, an error occurred during USB communication.

Parameters

errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::getErrorMessage().

Returns

- iQ::Trigger::RELEASED,
- iQ::Trigger::IDLE or
- iQ::Trigger::UNDEFINED

7.1.1.13 double iQ::Trigger::TriggerController::getTotalTime ( )

iQ::Trigger::TriggerController::getTotalTime

Returns

The total time for the sequence in seconds calculated as number of iterations times period time.

7.1.1.14 iQ::Trigger::TriggerOutputBehaviour iQ::Trigger::TriggerController::getTriggerOutputBehaviour ( )

iQ::Trigger::TriggerController::getTriggerOutputBehaviour

Returns the current trigger output behaviour.

Returns

- iQ::Trigger::NORMAL or
- iQ::Trigger::INVERSE

7.1.1.15 void iQ::Trigger::TriggerController::setSequence (int *nIterations*, double *periodTimeSec*, double *releaseTimeSec*, int & *errorCode* = `default_error`)

iQ::Trigger::TriggerController::setSequence Sets the required parameters for the sequence.

See Also

iQ::Trigger::TriggerController::stopSequence()
iQ::Trigger::TriggerController::startSequence()

Parameters

nIterations	The number of iterations of a period. Valid range is [1,n] and -1, -1 denotes infinite number of iterations.
periodTimeSec	The time for a period. Minimum value is 0.6 sec, which includes 0.5 sec as minimum idle time.
releaseTimeSec	The minimum value for the release time is 0.1 sec. Period and release time depend on each other. The time for a period must not be shorter than the release time + 0.5 seconds. See also image on main page.
errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::getErrorMessage().

7.1.1.16 void iQ::Trigger::TriggerController::setState (iQ::Trigger::TriggerState *state*, int & *errorCode* = `default_error`)

iQ::Trigger::TriggerController::setState

Use iQ::Trigger::RELEASE or iQ::Trigger::IDLE to set the iQ-Trigger to the desired state.

Parameters

state	Use only iQ::Trigger::RELEASE or iQ::Trigger::IDLE
errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::getErrorMessage().

7.1.1.17 void iQ::Trigger::TriggerController::setTriggerOutputBehaviour (iQ::Trigger::TriggerOutputBehaviour *state*, int & *errorCode* = `default_error`)

iQ::Trigger::TriggerController::setTriggerOutputBehaviour

Sets the behaviour for the trigger output port of the USB box. The trigger output depends on the state of the iQ-Trigger. May be iQ::Trigger::NORMAL or iQ::Trigger::INVERSE.

The trigger output has two physical states:

- Short-circuit
- Open-circuit

Table of possible combinations:

iQ-Trigger state	Trigger output behaviour	Trigger output port
IDLE	NORMAL	Open-circuit
RELEASED	NORMAL	Short-circuit
IDLE	INVERSE	Short-circuit
RELEASED	INVERSE	Open-circuit

Depending on the iQ-Trigger state, the trigger output is set to the aforementioned states.

Parameters

state	Use only iQ::Trigger::NORMAL or iQ::Trigger::INVERSE
errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::getErrorMessage().

7.1.1.18 void iQ::Trigger::TriggerController::startSequence (int & *errorCode* = `default_error`)

iQ::Trigger::TriggerController::startSequence Starts the sequence.

Parameters

errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::getErrorMessage().

7.1.1.19 void iQ::Trigger::TriggerController::toggleState (iQ::Trigger::TriggerState *state*, int *msec*, int & *errorCode* = `default_error`)

iQ::Trigger::TriggerController::toggleState

Toggles the state of the iQ-Trigger for a specified amount of time. The iQ-Trigger is set to given state regardless of the current state. The time can not be less than 100 milliseconds. Otherwise function does nothing and errorCode is iQ::Trigger::Error::InvalidParameter.

Parameters

state	Use only iQ::Trigger::RELEASED or iQ::Trigger::IDLE
msec	Time in milliseconds, minimum value is 100 ms.
errorCode	Describes the error that occurred during processing. 0 denotes no error. You can receive a corresponding error message by calling iQ::Trigger::TriggerController::getErrorMessage().

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

- triggercontroller.h
- triggercontroller.cpp

7.2 iQ::Trigger::TriggerError Struct Reference

The documentation for this struct was generated from the following file:

- error.h

7.3 iQ::Trigger::Version Class Reference

Static Public Member Functions

- static int getMajor ()

  iQ::Trigger::Version::getMajor Returns the major version number.
- static int getMinor ()

  iQ::Trigger::Version::getMajor Returns the minor version number.
- static int getPatch ()

  iQ::Trigger::Version::getMajor Returns the patch version number.
- static std::string getVersion ()

  iQ::Trigger::Version::getVersion Returns the API version number.
  Major, minor and patch version number are concatenated to a string separated by a dot(.).

7.3.1 Member Function Documentation

7.3.1.1 int iQ::Trigger::Version::getMajor ( ) `[static]`

iQ::Trigger::Version::getMajor Returns the major version number.

For example API version 2.0.5 denotes:

- 2: major
- 0: minor
- 5: patch

Returns

The major version number of the iQ-Trigger API.

7.3.1.2 int iQ::Trigger::Version::getMinor ( ) `[static]`

iQ::Trigger::Version::getMajor Returns the minor version number.

For example API version 2.0.5 denotes:

- 2: major
- 0: minor
- 5: patch

Returns

The minor version number of the iQ-Trigger API.

7.3.1.3 int iQ::Trigger::Version::getPatch ( ) `[static]`

iQ::Trigger::Version::getMajor Returns the patch version number.

For example API version 2.0.5 denotes:

- 2: major
- 0: minor
- 5: patch

Returns

The patch version number of the iQ-Trigger API.

7.3.1.4 std::string iQ::Trigger::Version::getVersion ( ) `[static]`

iQ::Trigger::Version::getVersion Returns the API version number.

Major, minor and patch version number are concatenated to a string separated by a dot(.).

Returns

The API version number.

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

- version.h
- version.cpp

Index