The Roboteq® Devices API Documentation

V1.1, July 5, 2017

Visit www.roboteq.com to download the latest revision of this manual

Table of Contents

1. Introduction **1**

2. Deliverables **2**

2.1 Windows 2
2.2 Linux 2

3. How to use **3**

4. API Documentation **5**

4.1 sleepms Function 5
4.2 RoboteqDevice:: Connect 6
4.3 RoboteqDevice:: Disconnect 7
4.4 RoboteqDevice:: IsConnected 8
4.5 RoboteqDevice:: SetConfig 9
4.6 RoboteqDevice:: GetConfig 11
4.7 RoboteqDevice:: SetCommand 12
4.8 RoboteqDevice:: GetValue 14

Introduction

The Roboteq ® Devices API is a set of functions used to control motor controller behavior.

These API's come in form of static libraries that can be used on both Windows and Linux.

Deliverables

1.1 Windows

- **RoborunDevice.lib:** a static library contains the API's, which should be statically linked with your program.
- **RoboteqDevice.h:** contains the API's prototype.
- **ErrorCodes.h:** contains constants representing the error codes that can be returned by the API's.
- **Constants.h:** contains a set of constants representing the commands supported by the device.
- **sample.cpp:** a sample program representing how to use the API.

1.2 Linux

- **RoborunDevice.o:** object file contains the API's, which should be compiled with your program.
- **RoboteqDevice.h:** contains the API's prototype.
- **ErrorCodes.h:** contains constants representing the error codes that can be returned by the API's.
- **Constants.h:** contains a set of constants representing the commands supported by the device.
- **sample.cpp:** a sample program representing how to use the API.

How to use

The following is a sample to show how to use the API in a simple program:

```
#include <iostream>
#include <stdio.h>
#include <string.h>

#include "RoboteqDevice.h"
#include "ErrorCodes.h"
#include "Constants.h"

using namespace std;

int main(int argc, char *argv[])
{
   string response = "";
   //Create an instance of RoboteqDevice.
   RoboteqDevice device;
   //Connect to the device, for windows use "\\\\.\\com1" for com1.
   int status = device.Connect("/dev/ttyS0");

   //Check to see if the connection succeeded.
   if(status != RQ_SUCCESS)
   {
      cout<<"Error connecting to device: "<<status<<"."<<endl;
      return 1;
   }

   cout<<"- SetConfig(_DINA, 1, 1)...";
   if((status = device.SetConfig(_DINA, 1, 1)) != RQ_SUCCESS)
      cout<<"failed --> "<<status<<endl;
   else
      cout<<"succeeded."<<endl;

   //Wait 10 ms before sending another command to device
   sleepms(10);

   int result;
   cout<<"- GetConfig(_DINA, 1)...";
   if((status = device.GetConfig(_DINA, 1, result)) != RQ_SUCCESS)
      cout<<"failed --> "<<status<<endl;
   else
      cout<<"returned --> "<<result<<endl;

   //Wait 10 ms before sending another command to device
   sleepms(10);
```

```
   cout<<"- GetValue(_ANAIN, 1)...";
   if((status = device.GetValue(_ANAIN, 1, result)) != RQ_SUCCESS)
      cout<<"failed --> "<<status<<endl;
   else
      cout<<"returned --> "<<result<<endl;

   //Wait 10 ms before sending another command to device
   sleepms(10);

   cout<<"- SetCommand(_GO, 1, 1)...";
   if((status = device.SetCommand(_GO, 1, 1)) != RQ_SUCCESS)
      cout<<"failed --> "<<status<<endl;
   else
      cout<<"succeeded."<<endl;

   device.Disconnect();
   return 0;
}
```

API Documentation

1.3 sleepms Function

Suspends the execution of the program until the timeout interval elapses.

Syntax:

```
void sleepms(int milliseconds)
```

Parameters:

milliseconds [in]

The time interval for which execution is to be suspended, in milliseconds.

Return Value:

This function does not return a value.

Remarks:

Use this function to obtain pauses between two successive device requests.

Examples:

The following example shows how to use the sleepms function to obtain pauses. The example counts down for 10 seconds.

```
#include <iostream>
#include <stdio.h>
#include <string.h>

#include "RoboteqDevice.h"

using namespace std;

int main(int argc, char *argv[])
{
   for(int i = 0; i < 10; i++)
   {
      cout<<"Wait for: "<<10 - i<<" second(s)."<<endl;
      sleepms(1000);
   }
   return 0;
}
```

1.4 RoboteqDevice:: Connect

Opens a connection to the device connected to a specified port.

Syntax:

```
int Connect(string port)
```

Parameters:

port [in]

The port that device is attached to.

Return Value:

One of the following values determines the operation status:

Value	Constant	Description
0	RQ_SUCCESS	The operation completed successfully.
1	RQ_ERR_OPEN_PORT	Error occurred while trying to open the communication port.
7	RQ_UNRECOGNIZED_DEVICE	The device is not recognized.
8	RQ_UNRECOGNIZED_VERSION	Invalid device version.

Examples:

The following example shows how to obtain a connection to Roboteq device.

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#include "RoboteqDevice.h"
#include "ErrorCodes.h"
#include "Constants.h"
using namespace std;
int main(int argc, char *argv[])
{
   //Create an instance of RoboteqDevice.
   RoboteqDevice device;
   //Connect to the device, for windows use "\\\\.\\com1" for com1.
   int status = device.Connect("/dev/ttyS0");

   //Check to see if the connection succeeded.
   if(status != RQ_SUCCESS)
   {
      cout<<"Error connecting to device: "<<status<<"."<<endl;
      return 1;
   }

   cout<<"Connection to device succeeded."<<endl;
   device.Disconnect();
   return 0;
}
```

1.5 RoboteqDevice:: Disconnect

Closes the connection to the device.

Syntax:

```
void Disconnect()
```

Parameters:

This function does not need any parameters.

Return Value:

This function does not return a value.

Examples:

See **RoboteqDevice:: Connect** example.

1.6 RoboteqDevice:: IsConnected

Checks whether connection to device is opened.

Syntax:

```
bool IsConnected()
```

Parameters:

This function does not need any parameters.

Return Value:

Returns true if the device connection is open, false otherwise.

1.7 RoboteqDevice:: SetConfig

Changes one of the controller's configuration parameters. The changes are made in the controller's RAM and take effect immediately. Configuration changes are not stored in EEPROM.

Syntax:

```
int SetConfig(int configItem, int index, int value)
int SetConfig(int configItem, int value)
```

Parameters:

configItem [in]

The configuration item needs to be changed. See ***Table 1 below*** for constants that can be used with this function.

index [in]

Used to select one of the configuration item in multi-channel configurations. When accessing a configuration parameter that is not part of an array, the index value 1 must be used. Details on the various configurations items, their effects and acceptable values can be found in the Controller's User Manual.

If the index is omitted, it is supposed to be 0.

value [in]

The new parameter configuration value.

Table 1 Configuration Items Constants

Config Item	Description
_ACS	Enable Ana Center Safety
_ACTR	Analog Center
_ADB	Analog Deadband
_AINA	Analog Input Actions
_ALIM	Motor Amps Limit
_ALIN	Analog Linearity
_AMAX	Analog Max
_AMAXA	Action on Analog Input Max
_AMIN	Analog Min
_AMINA	Action on Analog Input Min
_AMOD	Analog Input Mode
_AMS	Enable Ana Min/Max Safety
_APOL	Analog Input Polarity
_ATGA	Amps Trigger Action
_ATGD	Amps Trigger Delay
_ATRIG	Amps Trigger Value
_BHL	Encoder High Limit
_BHLA	Encoder High Limit Action
_BHOME	Brushless Counter Load at Home Position

Config Item	Description
_EMOD	Encoder Operating Mode
_EPPR	Encoder PPR
_ICAP	Motor(s) Int Cap
_KD	Set PID Differential Gain
_KDC1	KD Curve Points for Motor1
_KDC2	KD Curve Points for Motor2
_KI	Set PID Integral Gain
_KIC1	KI Curve Points for Motor1
_KIC2	KI Curve Points for Motor2
_KP	Set PID Proportional Gain
_KPC1	KP Curve Points for Motor1
_KPC2	KP Curve Points for Motor2
_MAC	Motor(s) Desired Acceleration
_MDEC	Motor(s) Desired Deceleration
_MMOD	Motor Operating Mode
_MVEL	Motor(s) Default Position Velocity
_MXPF	Motor Max Power
_MXPR	Motor Max Power
_MXRPM	Motor RPM at 100%

Config Item	Description
_BLFB	Speed and Position sensor feedback
_BLL	Encoder Low Limit
_BLLA	Encoder Low Limit Action
_BLSTD	BL Stall Detection
_BPOL	Number of Poles of BL Motor
_CLERD	Close Loop Error Detection
_CLIN	Command Linearity
_DFC	Default Command value
_DINA	Digital Input Action
_DINL	Read Digital Inputs
_DOA	Digital Output Action
_DOL	Digital Output Action
_ECHOF	Disable/Enabe RS232 & USB Echo
_EHL	Encoder High Limit
_EHLA	Encoder High Limit Action
_EHOME	Encoder Counter Load at Home Position
_ELL	Encoder Low Limit
_ELLA	Encoder Low Limit Action

Config Item	Description
_MXTRN	Number of Motor Turns between Limits
_PCTR	Pulse Center
_PDB	Pulse Deadband
_PIDM	Set PID Options
_PINA	Pulse Input Actions
_PLIN	Pulse Linearity
_PMAX	Pulse Max
_PMAXA	Action on Pulse Input Max
_PMIN	Pulse Min
_PMINA	Action on Pulse Input Min
_PMOD	Pulse Input Mode
_PMS	Enable Pulse Min/Max safety
_PPOL	Pulse Input Polarity
_RWD	RS232 Watchdog (0 to disable)
_SXC	Sepex Curve Points
_SXM	Minimum Field Current

For full list of constants, please refer to your controller's reference manual.

Return Value:

One of the following values determines the operation status:

Value	Constant	Description
0	`RQ_SUCCESS`	The operation completed successfully.
2	`RQ_ERR_NOT_CONNECTED`	The device not connected, you should call the Connect function and insure that the device connection succeeded.
3	`RQ_ERR_TRANSMIT_FAILED`	Error occurred while transmitting data to device.
4	`RQ_ERR_SERIAL_IO`	Error occurred to serial communication.
5	`RQ_ERR_SERIAL_RECEIVE`	Error occurred while transmitting data from device.
6	`RQ_INVALID_RESPONSE`	Invalid response to the issued command.
9	`RQ_INVALID_CONFIG_ITEM`	Invalid configuration item, it should be in the range [0, 255].
12	`RQ_INDEX_OUT_RANGE`	The item index is out of range.
13	`RQ_SET_CONFIG_FAILED`	Failed to set device configuration.

Examples:

See ***sample.cpp***.

1.8 RoboteqDevice:: GetConfig

Reads one of the controller's configuration parameters.

Syntax:

```
int GetConfig(int configItem, int index, int &result)
int GetConfig(int configItem, int &result)
```

Parameters:

configItem [in]

The configuration item needs to be read. See ***Table 1 above*** for constants that can be used with this function.

index [in]

Used to select one of the configuration item in multi-channel configurations. When accessing a configuration parameter that is not part of an array, the index value 1 must be used. Details on the various configurations items, their effects and acceptable values can be found in the Controller's User Manual.

If the index is omitted, it is supposed to be 0.

result [out]

Contains the configuration item value in case of function success.

Return Value:

One of the following values determines the operation status:

Value	Constant	Description
0	RQ_SUCCESS	The operation completed successfully.
2	RQ_ERR_NOT_CONNECTED	The device not connected, you should call the Connect function and insure that the device connection succeeded.
3	RQ_ERR_TRANSMIT_FAILED	Error occurred while transmitting data to device.
4	RQ_ERR_SERIAL_IO	Error occurred to serial communication.
5	RQ_ERR_SERIAL_RECEIVE	Error occurred while transmitting data from device.
6	RQ_INVALID_RESPONSE	Invalid response to the issued command.
9	RQ_INVALID_CONFIG_ITEM	Invalid configuration item, it should be in the range [0, 255].
12	RQ_INDEX_OUT_RANGE	The item index is out of range.
14	RQ_GET_CONFIG_FAILED	Failed to get device configuration.

Examples:

See ***sample.cpp***.

1.9 RoboteqDevice:: SetCommand

This function is used to send operating commands to the controller at runtime. The function requires a Command Item, and a Value as parameters. The Command Item can be any one from the table below. Details on the various commands, their effects and acceptable ranges can be found in the Controller's User Manual.

Syntax:

```
int SetCommand(int commandItem, int index, int value)
int SetCommand(int commandItem, int value)
```

Parameters:

commandItem [in]

The command item needs to be set. See ***Table 2 below*** for constants that can be used with this function.

index [in]

Used to select one of the command channel in multi-channel commands. Details on the various commands, their effects and acceptable ranges can be found in the Controller's User Manual.

If the index is omitted, it is supposed to be 0.

value [in]

The new command value.

Table 2 Command Items Constants

Command Item	Description
ACCEL	Set Acceleration
_DECEL	Set Deceleration
_DOUT	Set all Digital Out bits
_DRES	Reset Individual Digital Out bits
_DSET	Set Individual Digital Out bits
_ESTOP	Emergency Shutdown
_GO	Set Motor1 Command
_HOME	Load Home counter

For full list of constants, please refer to your controller's reference manual.

Return Value:

One of the following values determines the operation status:

Value	Constant	Description
0	RQ_SUCCESS	The operation completed successfully.
2	RQ_ERR_NOT_CONNECTED	The device not connected, you should call the Connect function and insure that the device connection succeeded.
3	RQ_ERR_TRANSMIT_FAILED	Error occurred while transmitting data to device.

4	RQ_ERR_SERIAL_IO	Error occurred to serial communication.
5	RQ_ERR_SERIAL_RECEIVE	Error occurred while transmitting data from device.
6	RQ_INVALID_RESPONSE	Invalid response to the issued command.
11	RQ_INVALID_COMMAND_ITEM	Invalid command item, it should be in the range [0, 255].
12	RQ_INDEX_OUT_RANGE	The item index is out of range.
16	RQ_SET_COMMAND_FAILED	Failed to set device command.

Examples:

See ***sample.cpp***.

1.10 RoboteqDevice:: GetValue

Reads one of the controller's operating parameters.

Syntax:

```
int GetValue(int operatingItem, int index, int &result)
int GetValue(int operatingItem, int &result)
```

Parameters:

operatingItem [in]

The operating item needs to be read. See ***Table 3 below*** for constants that can be used with this function.

index [in]

Used to select one of the operating item in multi-channel configurations. When accessing operating parameter that is not part of an array, the index value 1 must be used. Details on the various operating items, can be found in the Controller's User Manual.

If the index is omitted, it is supposed to be 0.

result [out]

Contains the operating item value in case of function success.

Table 3 Operating Items Constants

Config Item	Description	Config Item	Description
_ABCNTR	Absolute Encoder Count	**_FLTFLAG**	Fault Flags
_ABSPEED	Encoder Motor Speed in RPM	**_LOCKED**	Lock status
_ANAIN	Analog Inputs	**_LPERR**	Closed Loop Error
_BATAMPS	Battery Amps	**_MOTAMPS**	Motor Amps
_BLCNTR	Absolute Brushless Counter	**_MOTCMD**	Actual Motor Command
_BLRCNTR	Brushless Count Relative	**_MOTPWR**	Applied Power Level
_BLRSPEED	BL Motor Speed as 1/1000 of Max	**_PLSIN**	Pulse Inputs
_BLSPEED	BL Motor Speed in RPM	**_RELCNTR**	Encoder Count Relative
_CMDANA	Internal Analog Command	**_RELSPEED**	Encoder Motor Speed as 1/1000 of Max
_CMDPLS	Internal Pulse Command	**_STFLAG**	Status Flags
_CMDSER	Internal Serial Command	**_TEMP**	Case & Internal Temperatures
_DIGIN	All Digital Inputs	**_TIME**	Time
_DIGOUT	Current Digital Outputs	**_VAR**	User Variable
_DIN	Individual Digital Inputs	**_VOLTS**	Internal Voltages
_FEEDBK	Feedback		

For full list of constants, please refer to your controller's reference manual.

Return Value:

One of the following values determines the operation status:

Value	Constant	Description
0	RQ_SUCCESS	The operation completed successfully.
2	RQ_ERR_NOT_CONNECTED	The device not connected, you should call the Connect function and insure that the device connection succeeded.
3	RQ_ERR_TRANSMIT_FAILED	Error occurred while transmitting data to device.
4	RQ_ERR_SERIAL_IO	Error occurred to serial communication.
5	RQ_ERR_SERIAL_RECEIVE	Error occurred while transmitting data from device.
6	RQ_INVALID_RESPONSE	Invalid response to the issued command.
10	RQ_INVALID_OPER_ITEM	Invalid operating item, it should be in the range [0, 255].
12	RQ_INDEX_OUT_RANGE	The item index is out of range.
15	RQ_GET_VALUE_FAILED	Failed to get operating item value.

Examples:

See ***sample.cpp***.