Capture Systems

CAP-LIB API

CAPTURE
precise tracking in its best

T. +972-3-3038108 | Fax +972-3-6200621
sales@capture-sys.com | www.capture-sys.com
6 Ravnitzky st. Beit Shaham Petach Tikva 4900617 Israel

Capture Systems

CAP-LIB API

Capture Systems LTD.

6 Ravnitzky St.,
Petach-Tikva 4900617
Israel
Office: +972-(0)3-3038108
Fax: +972-(0)3-6200621
support@capture-sys.com
www.capture-sys.com

Revision: 1.1

CAPTURE
precise tracking at its best

Table of contents

T. +972-3-3038108 | Fax +972-3-6200621
sales@capture-sys.com | www.capture-sys.com
6 Ravnitzky st. Beit Shoham Petach Tikva 4900617 Israel

Revision History

Revision	Date
1.0	5/18
1.1	5/18

T. +972-3-3038108 | Fax +972-3-6200621
sales@capture-sys.com | www.capture-sys.com
6 Ravnitzky st. Beit Shoham Petach Tikva 4900617 Israel

1 Introduction

The purpose of this document is to explain how to use the CAP-LIB API in the easiest way.
The CAP-LIB library is a collection of functions allowing the user to implement the control on the system over a PC application. The library contains several .DLL files that can be integrated in a host application written in C#.
The communication link between the PC and the system can be done via Ethernet or Serial link.
Realized as a collection of high-level functions, the library allows you to focus on the main aspects related to your application specific implementation, and to simply use the system and execute motion commands by calling appropriate functions from the library.

This manual describes how to install and use the components of CAP-LIB library.

2 Let's start

CAP-LIB is a collection of high level functions, grouped in several categories.
Most of these functions are Boolean type and return a 'True' value if the execution of the function performed without any error. If the function returns a 'False' value, you can retrieve the error description by calling the function CAP_GetLastError().
The CAP-LIB contains two .DLL files that handle the communication between the user application and the system.
CAPLIB.DLL – This file is the main library file.
Throw this file the user call the library functions according to the API.
RS232LIB.DLL – This file handle the RS-232 communication line between the user application and the system.
This feature is currently under development.
In order to write an application that will use CAP-LIB library you need to:

- Create a PC application project using your IDE.
- Make sure that your system supports your desired communication type.
  You can your supported communication types in your system order documents.
  Capture's default communication method is Ethernet – TCP.
- Set a reference to the 'CAPLIB.DLL' in your project.

3 CAP-LIB API

Capture's systems can be controlled using the Capture Communication Protocol or the CAP-LIB library.
The communication protocol contains a set of low-level commands, and by using those commands the user can achieve all of the system capabilities.
The CAP-LIB is a tool that helps you to handle the process of motion control application implemented on a PC application, at a high level, without the need to write complicated low-level code.

CAP-LIB allows to:

- Configure the motion profile (position or speed).
- Execute homing sequences.
- Read online data from each sensor that connected to the system.
- Save data to the controller.
- etc.

The main element of the library is the communication element, which is responsible of correct opening of the communication channel (serial RS-232 or Ethernet), as well as of CAP-LIB messages handling. You can communicate with one communication channel at a time even if your system supports multiple communication channels.

Consequently, each application you'll develop starts with opening of the communication channel by calling the CAP_Connect() function. The application must end with the CAP_Disconnect() function call.

Axes:
Mostly, Capture systems will have more than one axis. In order to define the desired axis, most of the CAP-LIB functions have an 'axis' parameter field. In this field the user specifies the desired axis.
CAP-LIB library supports the following axes numbers:

- Yaw – 0x01.
- Pitch – 0x02.
- Roll – 0x03.

Data types:
Library functions can be executed with or without data. The data formats for the library functions are:

- Double precision 64-bit (8 bytes).
- Floating point 32-bit (4 bytes).
- Unsinged Int 32-bit (UInt32) (4 bytes)
- Unsinged Int 16-bit (UInt16) (2 bytes)
- Unsinged/signed Int 8-bit (UInt8/Int8) (1 byte)
- ASCII string (Data length varies according to the string).

CAP-LIB infrastructure:
CAP-LIB contains an internal static class called 'CapWords' that holds the system keywords. Each non-data value that the library expected to receive from the user application is defined in this class for your easy access and usage.

4 Functions description

This section presents the functions implemented in the CAP-LIB library. The functions are grouped as follows:

- **Communication** – functions that manage the PC communication with the system.
- **Error handling** – functions that manage the system errors.
- **General System Commands** - functions to enable/disable the power stage of an axis, start/stop the motion, etc.
- **Motion commands** – functions for motion programming for the selected axis.
- **Scanning commands** – functions for scanning programming.
- **Presets** – functions for preset (points of interest) handling.
- **Stabilization commands** – functions that handle the stabilization mechanism.
- **External sensors** – functions that handle the communication with the external sensors that connected to the system.

For each function you will find the following information:

- C# prototype.
- Description of the arguments.
- A functional description.

4.1 Communication

4.1.1 CAP_Connect

Prototype:

```
bool CAP_Connect(byte channelType, string portNumber, int baudrates)
```

Arguments:

	Name	Data type	Description
Input	channelType	byte	The type of the communication channel to be opened
	portName	ASCII string	The system COM port or IP address (channel dependent).
	baudrates	UInt-32	Communication baudrate of port (channel dependent)
Output	Return	bool	True – The communication channel is open. False – Otherwise.

Description:

The function opens the communication channel specified with parameter **channelType**.
The CAP-LIB library supports the following types of communication channels:

- **Ethernet-TCP**
  channelType = CapWords.ProtocolEthernet for Ethernet-TCP communication**.**
- **Serial RS-232**
  channelType = CapWords.ProtocolRs232 for serial RS-232 communication.
- **Serial RS-422**
  channelType = CapWords.ProtocolRs422 for serial RS-422 communication.
  Remark: The system assumes that the user uses a communication converter between the PC RS-232 lines to the controller RS-422 lines. The .DLL uses the RS-232 line as the serial line for all serial protocols.

Depending on your communication channel type, the parameters **portName & baudrate** can be:

- For Ethernet communication:
  portName holds the system IP address, for example: '192.168.10.120'.
  baudrates holds the system port, for example: 4949.
- For Serial communications:
  portName holds the com port name, for example: 'COM1', 'COM2', 'COM3'…
  baudrates holds the communication speed, for example: 115200.
  The default baudrate value is 115200 bps.

4.1.2 CAP_Disconnect

Prototype:

```
bool CAP_Disconnect()
```

Arguments:

	Name	Data type	Description
Input	-		
Output	Return	bool	True – The communication channel is closed. False – Otherwise.

Description:
The function closes the active communication channel between the user application and the system.

4.1.3 CAP_IsConnected

Prototype

```
bool CAP_IsConnected()
```

Arguments:

	Name	Data type	Description
Input	-		
Output	Return	bool	True – The communication channel is active. False – Otherwise.

Description:
The function checks and returns the current communication state with the system**.**

4.1.4 CAP_SetComType

Prototype

```
bool CAP_SetComType(byte channelType)
```

Arguments:

	Name	Data type	Description
Input	channelType	byte	The new communication channel number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function changes the active communication channel to the specified channel with parameter **channelType**.
The CAP-LIB library supports the following types of communication channels:

- Ethernet-TCP
  channelType = CapWords.ProtocolEthernet for Ethernet-TCP communication**.**
- **Serial RS-232**
  channelType = CapWords.ProtocolRs232 for serial RS-232 communication.
- **Serial RS-422**
  channelType = CapWords.ProtocolRs422 for serial RS-422 communication.

After sending this command the system will restart itself in order the changes will take place. After this restart you will be able to communicate with the system using your new communication channel type. The system restart process can take about 30 seconds on order to complete.

4.1.5 CAP_SetIp

Prototype

```
bool CAP_SetIp(string ipAddress, ushort port)
```

Arguments:

	Name	Data type	Description
Input	ipAddress	ASCII string	The new IP address value
	port	UInt-16	The new port value
Output	Return	bool	True – The IP address changed successfully. False – Otherwise.

Description:
This function is valid only while the active communication channel is Ethernet-TCP. In other cases the function will return false immediately.
The function changes the system IP address and port according to the user input. The new IP address is specifies with parameter **ipAddress**.
CAP-LIB supports the standard IPv4 format for the system IP addresses, for example: '192.168.10.120'.
The new system port is specifies with the parameter **port**.
After sending this command the system will restart itself in order the changes will take place. After this restart you will be able to communicate with the system using your new IP address.
The system restart process can take about 30 seconds on order to complete.
Remark: In order to connect to the system, your system and PC should have the same subnet.

4.1.6 CAP_GetIpAddress

Prototype

```
bool CAP_GetIpAddress(out string ipAddress)
```

	Name	Data type	Description
Input	-		-
Output	ipAddress	ASCII string	Returns the system IP address in IPv4 format.
	Return	bool	True – The command runs successfully. False – Otherwise.

Arguments:

Description:
The function returns the current system IP address in IPv4 format.

4.1.7 CAP_GetIpPort

Prototype

```
bool CAP_GetIpPort(out ushort port)
```

	Name	Data type	Description
Input	-		-
Output	port	UInt-16	Returns the system IP address in IPv4 format.
	Return	bool	True – The command runs successfully. False – Otherwise.

Arguments:

Description:
The function returns the current system port.

4.2 Error handling

4.2.1 CAP_ResetFault

Prototype

```
bool CAP_ResetFault(byte axis)
```

	Name	Data type	Description
Input	axis	byte	Selected axis number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Arguments:

Description:
The function resets the selected **axis** fault status.

4.2.2 CAP_GetLastError

Prototype

```
string CAP_GetLastError()
```

Arguments:

	Name	Data type	Description
Input	**-**		
Output	Return	ASCII string	Last error value

Description:
The function returns the last system error.

4.3 General System Commands

4.3.1 CAP_Power

Prototype

```
bool CAP_Power(byte axis, byte state)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	state	byte	Enables / Disables the power state to the selected axis
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function enables/disables the power stage of the selected axis.
- state = true: the power stage is enabled.
- state = false: the power stage is disabled.

4.3.2 CAP_Stop

Prototype

```
bool CAP_Stop(byte axis)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function stops the motion of the selected axis.

4.3.3 CAP_ResetAxis

Prototype

```
bool CAP_ResetAxis (byte axis)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function resets the selected axis. During the axis reset the system will be disabled.
This process mostly takes several seconds.

4.3.4 CAP_ResetSystem

Prototype

```
bool CAP_ResetSystem ()
```

Arguments:

	Name	Data type	Description
Input	-		
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function resets entire system. During the system reset process each axis will be restarted manually and then the main system controller will be restarted. During the system reset process the system will be disabled.
This process mostly takes about a minute.

4.3.5 CAP_ReadStatus

Prototype

```
bool CAP_ReadStatus(ushort register, byte axis, out short data)
```

Arguments:

	Name	Data type	Description
Input	register	UInt-16	Selected register to read
	axis	byte	Selected axis number
Output	data	Int-16	Register value
	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function returns the value of the register that specified with parameter **register**.
The CAP-LIB library supports the following types of registers:

- **MER – Motion error register**
  Register = CapWords. RegMer.
- **DER – Detailed error register**
  Register = CapWords. RegDer.
- **MSR – Motion status register**
  Register = CapWords. RegMsr.
- **SRL – Status register low**
  Register = CapWords.RegSrl.
- **SRH – Status register high**
  Register = CapWords.RegSrh.

The registers content are described in a different document called 'Capture Registers'.

4.4 Motion commands

4.4.1 CAP_MoveAbsolute

Prototype

```
bool CAP_MoveAbsolute(byte axis, float absPosition, float speed, float acceleration,
byte referenceBase)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	absPosition	float 32-bit	Position to reach expressed in [deg]
	speed	float 32-bit	Slew speed expressed in [deg/sec]. The speed must be larger than zero in order to perform a motion.
	acceleration	float 32-bit	Acceleration rate expressed in [deg/sec^2]. The acceleration must be larger than zero in order to perform a motion.
	referenceBase	byte	Specifies how the motion reference is computed: from actual values of position and speed reference or from actual values of load/motor position and speed.
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function creates an absolute positioning with trapezoidal profile.
The motion is described through **absPosition** parameter for position to reach, **speed** for slew speed and **acceleration** for acceleration/deceleration rate. The position to reach can be positive or negative. The **speed** and **acceleration** can be **only** positive.
Set **referenceBase = CapWords.FromReference** if you want the reference generator to compute the motion profile starting from the actual values of the position and speed reference.
Set **ReferenceBase = CapWords. FromMeasure** if you want the reference generator to compute the motion profile starting from the actual values of the load/motor position and speed. When this option is used, at the beginning of each new motion profile, the position and speed reference are updated with the actual values of the load/motor position and speed.

Remark: In systems without any feedback, this option is ignored because there is no position and/or speed feedback.

4.4.2 CAP_MoveRelative

Prototype

```
bool CAP_MoveRelative(byte axis, float relPosition, float speed, float acceleration,
byte referenceBase)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	relPosition	float 32-bit	Position increment expressed in [deg]
	speed	float 32-bit	Slew speed expressed in [deg/sec]. The speed must be larger than zero in order to perform a motion.
	acceleration	float 32-bit	Acceleration rate expressed in [deg/sec^2]. The acceleration must be larger than zero in order to perform a motion.
	referenceBase	byte	Specifies how the motion reference is computed: from actual values of position and speed reference or from actual values of load/motor position and speed.
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function creates a relative positioning with trapezoidal profile.
The motion is described through **relPosition** parameter for position increment, **speed** for slew speed and **acceleration** for acceleration/deceleration rate. The position to reach can be positive or negative. The **speed** and **acceleration** can be **only** positive.
Set **referenceBase = CapWords. FromReference** if you want the reference generator to compute the motion profile starting from the actual values of the position and speed reference.
Set **ReferenceBase = CapWords. FromMeasure** if you want the reference generator to compute the motion profile starting from the actual values of the load/motor position and speed. When this option is used, at the beginning of each new motion profile, the position and speed reference are updated with the actual values of the load/motor position and speed.

Remark: In systems without any feedback, this option is ignored because there is no position and/or speed feedback.

4.4.3 CAP_MoveVelocity

Prototype

```
bool CAP_MoveRelative(byte axis, float speed, float acceleration, byte
referenceBase)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	speed	float 32-bit	Slew speed expressed in [deg/sec]. The speed must be larger than zero in order to perform a motion.
	acceleration	float 32-bit	Acceleration rate expressed in [deg/sec^2]. The acceleration must be larger than zero in order to perform a motion.
	referenceBase	byte	Specifies how the motion reference is computed: from actual values of position and speed reference or from actual values of load/motor position and speed.
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function creates a trapezoidal speed profile.
The motion is described through **speed** parameter for jog speed. The motor accelerates until the desired jog speed is reached.
The jog speed can be both positive and negative. The **acceleration** can be only positive.

Set **referenceBase = CapWords.FromReference** if you want the reference generator to compute the motion profile starting from the actual values of the position and speed reference.
Set **ReferenceBase = CapWords.FromMeasure** if you want the reference generator to compute the motion profile starting from the actual values of the load/motor position and speed. When this option is used, at the beginning of each new motion profile, the position and speed reference are updated with the actual values of the load/motor position and speed.

Remark: In systems without any feedback, this option is ignored because there is no position and/or speed feedback.

4.4.4 CAP_RunHoming

Prototype

```
bool CAP_RunHoming(byte axis)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function runs the homing sequence of the selected axis.
Remark: in order to activate a homing sequence you need to make sure that such sequence is defined in your system configurations.

4.4.5 CAP_GetFloatVariable

Prototype

```
bool CAP_GetFloatVariabe(byte axis, byte variable, out float data)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	variable	byte	The desired variable name
Output	data	float 32-bit	Returned data
	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function reads the value of **variable** from the selected axis. The data type is 4 byte floating point value. The value read is converted to float and saved in the variable pointed by **data**.
A list of Capture variables you can get from the CapWords class.

4.5 Scanning Commands

4.5.1 CAP_StartScan

Prototype

```
bool CAP_StartScan(byte scanType, float yawMin, float yawMax, float pitchMin, byte numOfSteps, float stepHeight, float yawSpeed, bool shortPath)
```

Arguments:

	Name		Description
Input	scanType	byte	Scanning type
	yawMin	float 32-bit	Yaw starting point value represented as absolute degree
	yawMax	float 32-bit	Yaw ending point value represented as absolute degree
	pitchMin	float 32-bit	Pitch starting point value represented as absolute degree
	numOfSteps	float 32-bit	Number of scan steps according to the scan type
	stepHeight	byte	Each step height represented in [deg]
	yawSpeed	float 32-bit	Yaw jog speed represented in [deg/sec]
	shortPath	bool	True if the scan performs the shortest path between the Yaw values
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function activates the scan algorithm that described through the **scanType** parameter.
The CAPLIB library supports the following types of scan:

- **Zig-Zag**
  scanType = CapWords.ScanZigZag for zig-zag scanning mode.
- **Snake**
  scanType = CapWords.ScanSnake for snake scanning mode.
- **Square**
  scanType =CapWords. ScanSquare for square scanning mode.

Each scan starts from a defined point. This point represented by absolute degrees values of **yawMin** & **pitchMin** parameters. The edge point of the Yaw axis is represented thorough the **yawMax** parameter. The Pitch max value is calculated by the scanning algorithm according to the selected scanning type. The number of scanning steps should be defined through the **numOfSteps** parameter and each step height through the **stepHeight** parameter.
Shortest path parameter determines if the movement over the Yaw axis will be the shortest path between the **yawMin** & **yawMax** values. Setting the **shortPath** parameter to True will activate the short path calculation.

Remark: Scan mode can be activated for two axes system only.
Description of each scan mode can be found in the 'Capture Systems ,Command and Control API' document.

4.5.2 CAP_StopScan

Prototype

```
bool CAP_StopScan()
```

Arguments:

	Name	Data type	Description
Input	-		
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function stops the current active scan.

4.5.3 CAP_IsScanOn

Prototype

```
bool CAP_IsScanOn(out bool data)
```

Arguments:

	Name	Data type	Description
Input	-		
Output	data	bool	Scan state
	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function sets the current scanning state in the **data** parameter.
data = True indicates that a scanning is currently active while **data=False** indicates that the scan isn't active.

4.6 Presets

4.6.1 CAP_SetPreset

Prototype

```
bool CAP_SetPreset(float azimuth, float elevation, byte number)
```

Arguments:

	Name	Data type	Description
Input	azimuth	float 32-bit	Preset azimuth (Yaw) value represented as absolute degree
	elevation	float 32-bit	Preset elevation (Pitch) value represented as absolute degree
	number	float 32-bit	Preset number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function sets a preset (point of interest) in the system memory. The preset point values are defined by the **azimuth** and **elevation** parameters. The preset number is defined through the **number** parameter.
Remark: Preset number range is 1 - 15.

4.6.2 CAP_GetPreset

Prototype

```
bool CAP_GetPreset(byte number, out float azimuth, out float elevation)
```

Arguments:

	Name	Data type	Description
Input	number	byte	Preset number
Output	azimuth	float 32-bit	Preset azimuth value represented as absolute degree
	elevation	float 32-bit	Preset elevation value represented as absolute degree
	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function asks for a preset point that is stored in the system memory. The preset point values returned to the **azimuth** and **elevation** parameters. The point number is defined through the **number** parameter.
Remark: Preset number range is 1 - 15.

4.6.3 CAP_GoToPreset

Prototype

```
bool CAP_GoToPreset(byte number)
```

	Name	Data type	Description
Input	number	byte	Preset number
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Arguments:

Description:
The function sends the system to a preset that defined through the **number** parameter.
Remark: Preset number range is 1 - 15.

4.7 Stabilization commands

4.7.1 CAP_StabilizationOn

Prototype

```
bool CAP_StabilizationOn()
```

	Name	Data type	Description
Input	-		
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Arguments:

Description:
The function turns on the stabilization mechanism.
Remark: For supported systems only.

4.7.2 CAP_StabilizationOff

Prototype

```
bool CAP_StabilizationOff()
```

	Name	Data type	Description
Input	-		
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Arguments:

Description:
The function turns off the stabilization mechanism.
Remark: For supported systems only.

4.7.3 CAP_StabMoveAbsolute

Prototype

```
bool CAP_StabMoveAbsolute(byte axis, float absPosition, float speed)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	absPosition	float 32-bit	Position to reach expressed in [deg]
	speed	float 32-bit	Slew speed expressed in [deg/sec]. The speed must be larger than zero in order to perform a motion.
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
This function creates an absolute positioning profile while the stabilization mechanism is on.
The motion is described through **absPosition** parameter for position to reach and **speed** for slew speed. The position to reach can be positive or negative. The **speed** can be **only** positive.
Remark: For supported systems only.

4.7.4 CAP_StabMoveRelative

Prototype

```
bool CAP_StabMoveRelative(byte axis, float relPosition, float speed)
```

Arguments:

	Name		Description
Input	axis	byte	Selected axis number
	relPosition	float 32-bit	Position increment expressed in [deg]
	speed	float 32-bit	Slew speed expressed in [deg/sec]. The speed need to be larger than zero in order to perform a motion.
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function creates a relative positioning profile while the stabilization mechanism is on.
The motion is described through **relPosition** parameter for position increment and **speed** for slew speed.
The position increment can be positive or negative. The **speed** can be **only** positive.
Remark: For supported systems only.

4.7.5 CAP_StabMoveVelocity

Prototype

```
bool CAP_StabMoveVelocity (byte axis, float speed)
```

Arguments:

	Name	Data type	Description
Input	axis	byte	Selected axis number
	speed	float 32-bit	Slew speed expressed in [deg/sec]. The speed need to be larger than zero in order to perform a motion.
Output	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function creates a trapezoidal speed profile while the stabilization mechanism is on.
The motion is described through **speed** parameter for jog speed. The jog speed can be both positive and negative.
Remark: For supported systems only.

4.8 External sensors

4.8.1 CAP_GetImuData

Prototype

```
bool CAP_GetImuData(out float roll, out float pitch, out float yaw)
```

Arguments:

	Name	Data type	Description
Input	-		
Output	roll	float 32-bit	Current Roll value
	pitch	float 32-bit	Current Pitch value
	yaw	float 32-bit	Current Yaw value
	Return	bool	True – The command runs successfully. False – Otherwise.

Description:
The function returns the current IMU readings to the **roll**, **pitch** and **yaw** parameters.
Remark: For supported systems only.

5 Contact us

For any questions, assistance and troubleshooting regarding the CAP-LIB API please contact us at:
Web: capture-sys.com
Sales: sales@capture-sys.com
Support: support@capture-sys.com
Phone: +972-3-3038108

We welcome all feedback from our customers and will appreciate any comments regarding this manual. If you find any errors while reading this manual or parts which are not clear please send us an email and we will fix the problem.