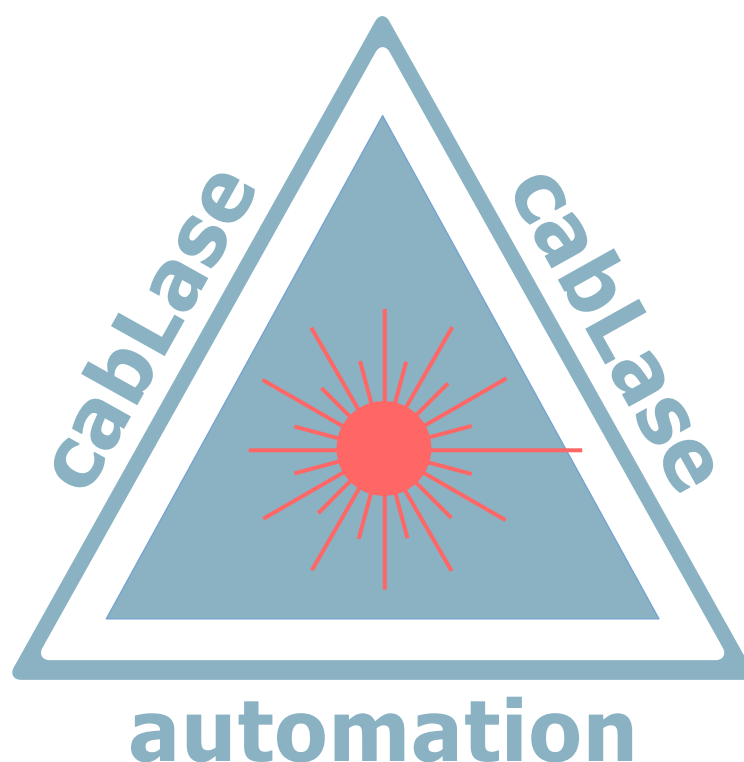


## PLC Programmer's Guide



## **cab FL<sup>+</sup>** **Remote API Interface**

**Art.-Nr.: 9009856**



<b>1</b>	<b>Preface .....</b>	<b>5</b>
<b>2</b>	<b>Introduction .....</b>	<b>7</b>
2.1	Notes .....	7
2.2	Intended Use .....	7
<b>3</b>	<b>Introduction .....</b>	<b>8</b>
3.1	General Functional Description .....	8
3.2	Using the Remote API Interface .....	8
3.3	API Communication Diagram .....	9
3.4	Connecting to the device using TCP/IP .....	10
3.5	Notes on cab PLC Demo Programs .....	11
3.5.1	Hard- and Software .....	11
3.5.2	Information on Sample Programs .....	11
3.5.3	Establishing the Network Connection .....	13
<b>4</b>	<b>Sample Programs .....</b>	<b>14</b>
4.1	Sample 1 .....	14
4.1.1	Description .....	14
4.1.2	Layout .....	14
4.1.3	PLC Sample Program .....	14
4.1.4	Flowchart .....	15
4.2	Sample 2 .....	16
4.2.1	Description .....	16
4.2.2	Layout .....	16
4.2.3	PLC Sample Program .....	16
4.2.4	Flowchart .....	17
4.3	Sample 3 .....	18
4.3.1	Description .....	18
4.3.2	Layout .....	18
4.3.3	PLC Sample Program .....	18
4.3.4	Flowchart .....	19
4.4	Sample Program 4 .....	20
4.4.1	Description .....	20
4.4.2	Layout .....	20
4.4.3	PLC Sample Program .....	20
4.4.4	Flowchart .....	21
<b>5</b>	<b>Establishing a TCP/IP connection with cabLase Editor 5 .....</b>	<b>22</b>
5.1	Selection of Network Adapter .....	22
5.2	Manually Connecting the Laser with cabLase Editor 5 .....	24
5.3	Automatically Connecting the Laser with cabLase Editor 5 .....	26
5.4	Changing the IP Address .....	27
<b>6</b>	<b>Establishing Stand-Alone Operation with cabLase Editor 5 .....</b>	<b>28</b>
6.1	General .....	28
6.2	Storing *.wlj files to the Memory of the Laser .....	28
6.3	Managing *.dat Files .....	29
6.4	Storing Laser Font Files .....	30
<b>7</b>	<b>Remote API Commands .....</b>	<b>31</b>
7.1	API Command Set .....	31
7.1	Abort .....	31
7.2	ClearJobList .....	32
7.3	ConnectNetworkShare .....	32
7.4	Echo .....	32
7.5	EnableObject .....	33
7.6	ExecuteJobContinuous .....	33
7.7	ExecuteJobOnce .....	34
7.8	GetAllIOWords .....	35

## Contents

<b>7</b>	<b>Remote API Commands.....</b>	<b>36</b>
7.9	GetFlashJobFileList .....	36
7.10	GetFontFileList.....	36
7.11	GetKFactor.....	37
7.12	GetNetworkJobFileList .....	37
7.13	GetObjectCenter .....	37
7.14	GetUSBJobFileList.....	38
7.15	HardwareReset .....	38
7.16	LoadFlashJob.....	39
7.17	LoadNetworkJob .....	39
7.18	LoadUSBJob .....	40
7.19	MakeJobActive.....	41
7.20	RemoveJob .....	41
7.21	RemoveObject .....	41
7.22	ReleaseHostControl .....	42
7.23	ResetObject .....	42
7.24	ResetUserTransform .....	42
7.25	SetExternalStartMode .....	43
7.26	SetObjectString .....	43
7.27	TakeHostControl.....	44
7.28	TransformObject.....	45
<b>8</b>	<b>Numerical Listing Remote API Commands .....</b>	<b>46</b>
8.1	Control.....	46
8.2	Objects .....	48
8.3	Marking Job.....	49
8.4	Administration .....	50
<b>9</b>	<b>Remote API Host Response Codes .....</b>	<b>51</b>
<b>10</b>	<b>Remote API System Error Messages .....</b>	<b>56</b>
<b>11</b>	<b>Remote API Object Types.....</b>	<b>57</b>
<b>12</b>	<b>Integration Interfaces.....</b>	<b>58</b>
12.1	External Interface I/O CON2 .....	58
12.2	Remote Interface CON3.....	59
12.3	Interlock / E-Stop Interface CON4.....	61
<b>13</b>	<b>Service .....</b>	<b>62</b>
13.1	Reference Documents .....	62
13.2	Revisio History of Sample Documents.....	62
13.3	Contact.....	62

**Edition: 161121**

**Art.-No. Programmer's Guide: 9009856**

### **Contents of Delivery**

Contents of delivery is specified in the shipping documents of the consignment..

The delivery has to be checked for completeness and flawless condition. For inconsistencies, the supplier has to be informed immediately.

### **General**

This manual contains neither instructions about basic operation of a computer, i.e. programmable control, nor about basic functions of operating systems Windows® or Mac OS®.

For more detailed information how to operate your computer, please refer to the manuals of the computer and its operating system.

### **Manufacturer**

cab Produkttechnik GmbH & Co KG  
Wilhelm-Schickard-Str. 14  
D-76131 Karlsruhe  
Phone: +49 (0)721/66 26 0  
Fax: +49 (0)721/6626 249  
<http://www.cab.de>  
e-mail: [laser@cab.de](mailto:laser@cab.de)

### Copyright

This documentation as well as translation hereof are property of cab Produkttechnik GmbH & Co. KG.

The replication, conversion, duplication or divulgement of the whole manual or parts of it for other intentions than its original intended purpose - in particular the procurement of spare parts for products sold by cab - demand the previous written authorization by cab.

Each possible software that represents part of this system is made available under license and may become use or copies only in agreement with the license conditions.

### Trademark

Microsoft® is a registered trademark of the Microsoft Corporation.

Windows XP®, Vista®, Windows 7®, Windows 8® und Windows 10® are registered trademarks of the Microsoft Corporation.

TrueType™ is a registered trademark of Apple Computer, Inc.

All other itemized company names and product names and its trademark are protected property of the respective proprietors.

### Editor

For any question or comments please contact cab Produkttechnik GmbH & Co. KG, Germany.

### Topicality

Due to further development of our products, discrepancies between documentation and product may occur. Please refer to [www.cab.de](http://www.cab.de) for the current issue.

### Terms and Conditions

Deliveries and performances are effected according to the cab General Conditions of sale.

**cab** Produkttechnik GmbH & Co KG  
 Wilhelm-Schickard-Str. 14  
 D-76131 Karlsruhe  
 Phone +49 721 6626-0  
 Fax +49 721 6626-249  
[www.cab.de](http://www.cab.de)  
[info@cab.de](mailto:info@cab.de)

#### France

cab Technologies S.à.r.l.  
 2a Rue de la Moder  
 Z.A. Nord du Val de Moder  
 67350 Niedermodern  
 phone +33 388 722501  
 fax +33 388 722502  
[info.fr@cab.de](mailto:info.fr@cab.de)  
[www.cab.de/fr](http://www.cab.de/fr)

#### China

cab Technology Co, Ltd.  
 9cab (Shanghai)Trading Co., Ltd.  
 Phone +86 21 6236-3161  
[www.cab.de/cn](http://www.cab.de/cn)  
[info.cn@cab.de](mailto:info.cn@cab.de)

#### USA

cab Technology, Inc.  
 87 Progress Avenue Unit 1  
 Tyngsboro, MA 01879  
 phone +1 978 649 0293  
 fax +1 978 649 0294  
[info.us@cab.de](mailto:info.us@cab.de)  
[www.cab.de/us](http://www.cab.de/us)

#### Asia

Asien  
 cab Technology Co., Ltd.  
 Junghe, Taipei, Taiwan  
 Phone +886 2 8227 3966  
[www.cab.de/tw](http://www.cab.de/tw)  
[info.asia@cab.de](mailto:info.asia@cab.de)  
 China

Other subsidiaries on request

## 2.1 Notes

Important information and instructions are designated as follows:



### **Danger!**

Draws attention to an exceptionally great, imminent danger to your health or life due to hazardous voltages.



### **Danger!**

Draws attention to a danger with high risk which, if not avoided, may result in death or serious injury.



### **Warning!**

Draws attention to a danger with medium risk which, if not avoided, may result in death or serious injury.



### **Caution!**

Draws attention to a danger with low risk which, if not avoided, may result in minor or moderate injury.



### **Attention!**

Draws attention to potential risks of property damage or loss of quality.



### **Note!**

Advices to make work routine easier or on important steps to be carried out.



### **Environment!**

Advices on protecting the environment.



Handling instructions



Reference to chapter, position, picture number or document.



Option (accessories, peripherals, extras).

Time Viewed in the display / monitor.

## 2.2 Intended Use

- The sample programs correspond to the state of the art and recognized safety rules. However, danger to the life and limb of the user or third parties and/or damage to the device or machine and other tangible assets can arise during use.
- The software may only be used for its intended purpose and applications with systems approved by the manufacturer.
- The backup of data, user information, system settings and machine dependent configuration data shall be task of the user. The manufacturer / supplier shall not be liable for any damage resulting from loss or damage of data; the user shall bear the risk alone.
- Any other use or use going beyond this shall be regarded as improper use. The manufacturer / supplier shall not be liable for any damage resulting from unauthorized use; the user shall bear the risk alone.
- Intended Use includes complying with the operating manual and the programmer's guide including the manufacturer's maintenance recommendations and specifications.
- This programmer's guide contains neither instructions about basic operation of a computer nor about basic functions of operating and control systems.

### 3.1 General Functional Description

The LEC-2 control card for cab marking lasers FL<sup>+</sup> was designed to be a powerful stand-alone controller, with the ability to be controlled and monitored externally via Remote API commands. The control card accepts commands received with so-called Response Codes, an answer generated by the control card.

The communication platform enables the cab marking lasers to be integrated into modern Industry 4.0 networks and machine controls.

The Remote API commands provide extended functionality to load marking jobs into the job memory of the laser control, rename and execute jobs or change administration settings.

Communication is divided into the two areas Host and Client. Host means the control of the marking laser. Clients are all higher-level controls at process control level communicating with the laser control via Remote API.

Basically, there are two data interfaces available for interacting. The Remote API may be installed via a TCP/IP socket connection, or a serial interface RS-232.

Main objective of this Programmer's Guide is to provide the PLC programmer with support on how to integrate cab marking lasers in automated inline processes.

Four reference examples from the field show which steps are necessary for integration. Automation processes are transparently visualized by the help of flowcharts and transferred into the Siemens PLC programming language.

All reference examples are provided with the corresponding layouts and PLC sequence programs.

The Remote API commands in this Programmer's Guide are provided in the PLC project as corresponding function blocks and may be combined and interlinked in any order.

The reference examples contain all digital signals of the interfaces CON2 and CON3 for synchronizing the processes and thus show how to control and monitor marking jobs.



**Note!**

If not already available, all example programs and files related to this Programmer's Guide may be downloaded [here](#).



**Note!**

The Remote API interface is available either as TCP/IP socket connection or serial interface RS-232.

### 3.2 Using the Remote API Interface

The LEC Remote API interface uses a message based communication protocol between the „Client“, that means a control at process control level and the „Host“, the laser control. The protocol is bi-directional.

After each command sent, the „Client“ gets a response from the „Host“ indicating the success or failure of the current command. Waiting for the response is obligatory to get a faultless synchronization in interacting. The response messages indicate whether the command has been executed successfully or with a failure.

All commands must end with „Carriage Return“ and „Line Feed“. Response messages from the Host automatically end with „Carriage Return“ and „Line Feed“.

The LEC Remote API interface provides a communication platform for controlling and monitoring marking jobs and the laser control.

The API allows to get and set system parameters as well as load marking jobs locally from the control or from the networks. Each of the marking objects may externally be edited or modified.



**Note!**

Sending new commands without waiting for a response from the Host will result in communication problems and might lead to undefined behavior.



**Note!**

All commands must end with „Carriage Return“ and „Line Feed“ (0X0D & 0X0A oder <CR>&<LF>). All responses must end with „Carriage Return“ and „Line Feed“ (0X0D & 0X0A oder <CR>&<LF>).



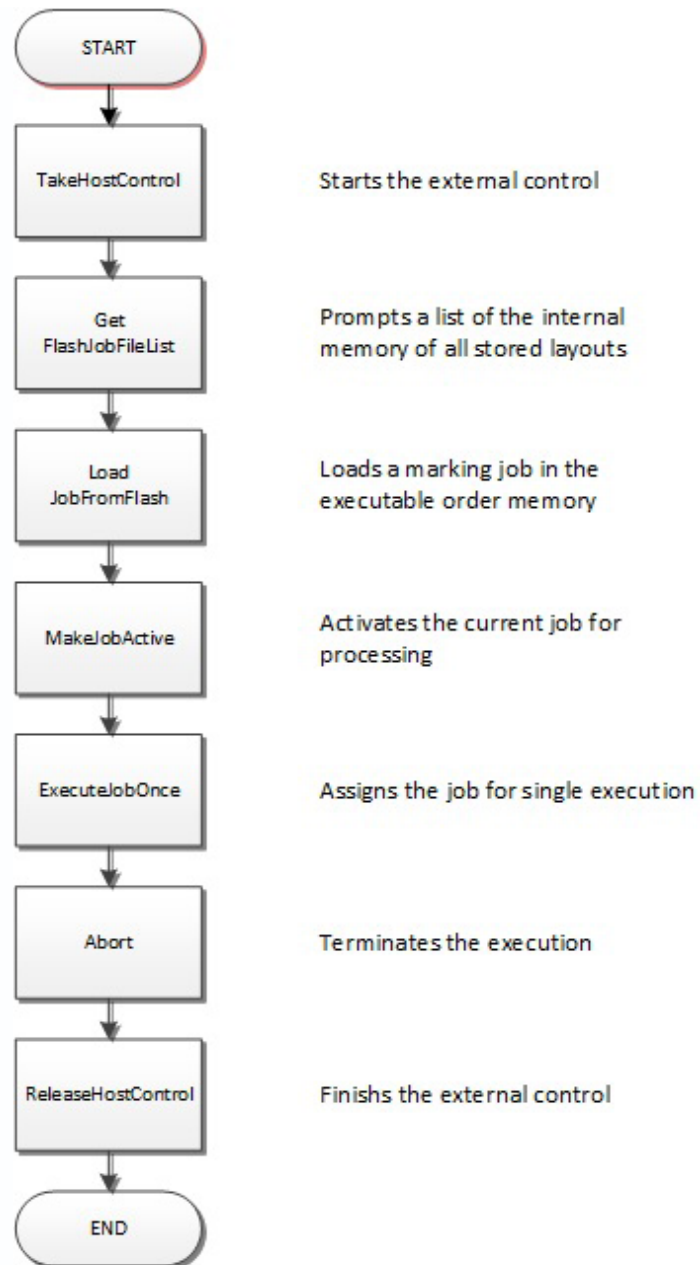
**Note!**

For all commands, character data must be sent using either 7-Bit Encoding ASCII 0-127 or 8-Bit Encoding ASCII0-255.



### 3.3 API Communication Diagram

The figure below shows an exemplary communication between client and host for marking a layout stored on the control board:



Get and set commands enable to check and set status and properties of the host. To be able to change properties, the client needs to be connected to the host via TakeHostControl.



**Note!**

To gain access to the Host use the command TakeHostControl.



**Note!**

To finish access to the Host use the command ReleaseHostControl.

### 3.4 Connecting to the device using TCP/IP

TCP/IP is a networking protocol and has the ability to communicate over local area networks (LAN – Local Area Network) or wide area networks/Internet (WAN – Wide Area Network). In order for programs and computer to communicate with the laser control and the remote server, the remote server needs to make itself available on a specified Port, and the IP address of the remote server has to be known.

It is this IP address and Port that ensures any message sent reaches the proper destination.

The laser control makes the Remote API Interface service available on Port 12500. The laser control can be configured to use a Static IP address, or to request an IP address each time it starts from a DHCP server. It is recommended in situations where the Remote API is used, to configure Static IP addressing for the laser control. Using this approach, the connection settings between Client and Host are statically set and may be used to connect again without any problem, in case of having f. ex. the devices switched off.

Any client computer that supports TCP/IP networking can establish remote control of the laser control. Only one computer or system can simultaneously control the laser. Clients may be installed in all networks running MS Windows or Linux.



**Note!**

When the device is delivered, configured IP address is 192.168.1.11.



**Note!**

The laser control LEC is using Port 12500 for sending and receiving the commands.



**Note!**

Using the Remote API it is recommended to configure a static IP address for the laser control.



**Note!**

Only one computer or system can simultaneously control the laser.



**Note!**

IP address settings or any other settings for communication will not become active until the next restart.

## 3.5 Notes on cab PLC Demo Programs

### 3.5.1 Hard- and Software

The sample programs have been created for a special project on the following hardware and software platform:

Manufacturer: Siemens

Hardware: CPU 1511-1PN

Programming software: Tia Portal V13 SPS1 Update 6

The project is saved under the file name „FL+ Remote API Library\_YYYYMMDD“. The indexing YYYYMMDD informs about the revision date and checks on the currentness.

As the project includes four sample programs in all and these shall be made ready to be selected, the entry condition "Start Sample" in the networks 4 is made available and must be activated to gain access to the laser.



#### Note!

**Only one computer or system can simultaneously control the laser. Using Siemens CPUs of other series, the program can be converted via the TIA portal.**

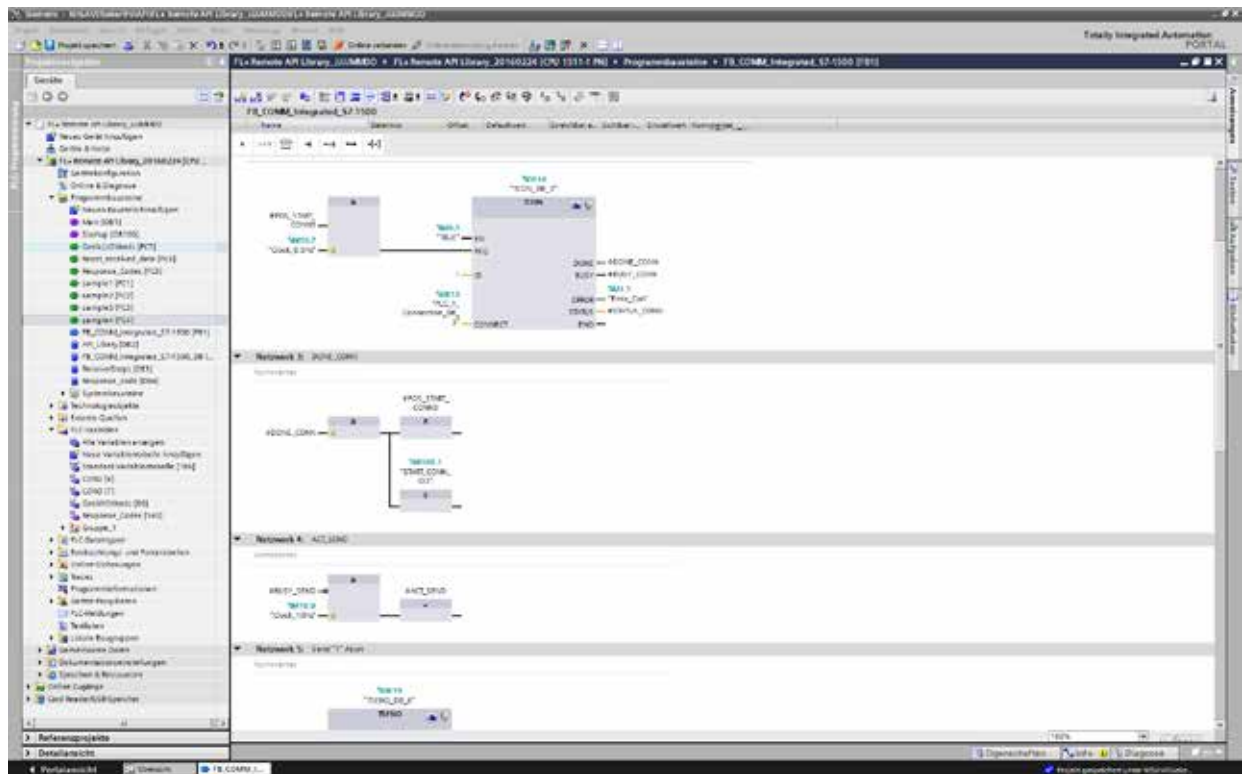


#### Note!

**When activating the entry condition "Start Sample" in the sample program, the laser software cabLase simultaneously has no access to the laser.**

### 3.5.2 Information on Sample Programs

The figure below shows the directory structure in the project folder:



The operating block Main OB1 defines the basic program flow. Using Startup OB100 specifies the behavior of the PLC program when switching the CPU on/off.

The sample programs 1 to 4 dealt with in this programmer's guide can be found in the following function blocks:

Sample1[FC1]

Sample2[FC2]

Sample3[FC3]

Sample4[FC4]

In addition to the actual Remote API command structure, the sample programs provide details on switching on and controlling the laser based on the digital interfaces of the laser CON2, CON3 and CON4.

**Network 1** in the sample is about switching on the laser. The laser can be switched on, as soon as safety circuit CON4 is closed, the "Power" signal at interface CON3 is present and no error messages are issued at the digital interfaces CON2 and CON3.

**Network 5** in the sample program is about analyzing the laser safety circuit at interface CON4. Opening the laser shutter and, consequently, emission of laser radiation is only possible, if the laser circuits are closed.

Each of the sample programs have the entry condition „Start Sample“ in **Network 4** that needs to be activated to gain access to the laser and to enable the sample program to be applied. Cancelling this entry condition enables to establish a connection to the laser with the software cabLase instead of the PLC.

**Response\_Codes [FC5]** Responsible for the evaluation of the API Response Codes received from the laser.

Communication block TCON for establishing a connection between PLC and laser is specified in function block **FB1**. There is also defined how data are sent / received to / from the laser.

Data block **DB[2]** includes a list of function blocks of all Remote API commands dealt with in this programmer's guide. These may be copied for any kind of interlinking for a customer-oriented programming.

The data received from the PLC are stored in the data block **DB[3]** into „Receive\_Data“ geschrieben. Should received data of the data block **DB[3]** be cancelled, the flag 5.4 „Reset\_Receive\_Data“ has temporarily to be set to „high“.

In **DB[3]** under field 8 „Response Code String“ includes a field prepared for issuing potential error messages as description. The issued description may f. ex. be used for the visualization in an user interface.

Conversion of the numerical API Response messages is made in **DB[4]**. This library is used to have clear text messages assigned to the numerical codes that may be adapted, for ex. when choosing a foreign language.

Block **PLCVariablen/CON2** includes a register that has assigned the digital in- and outputs of the interface CON2 to the wiring diagram.

Block **PLCVariablen/CON3** includes a register that has assigned the digital in- and outputs of the interface CON3 to the wiring diagram.



**Note!**

**Sample program 3 has not realized synchronization and control of the laser processing via wiring of interface CON2, but via Remote API. Hardware wiring is not necessary. Switching on and controlling the laser is realized via CON3.**



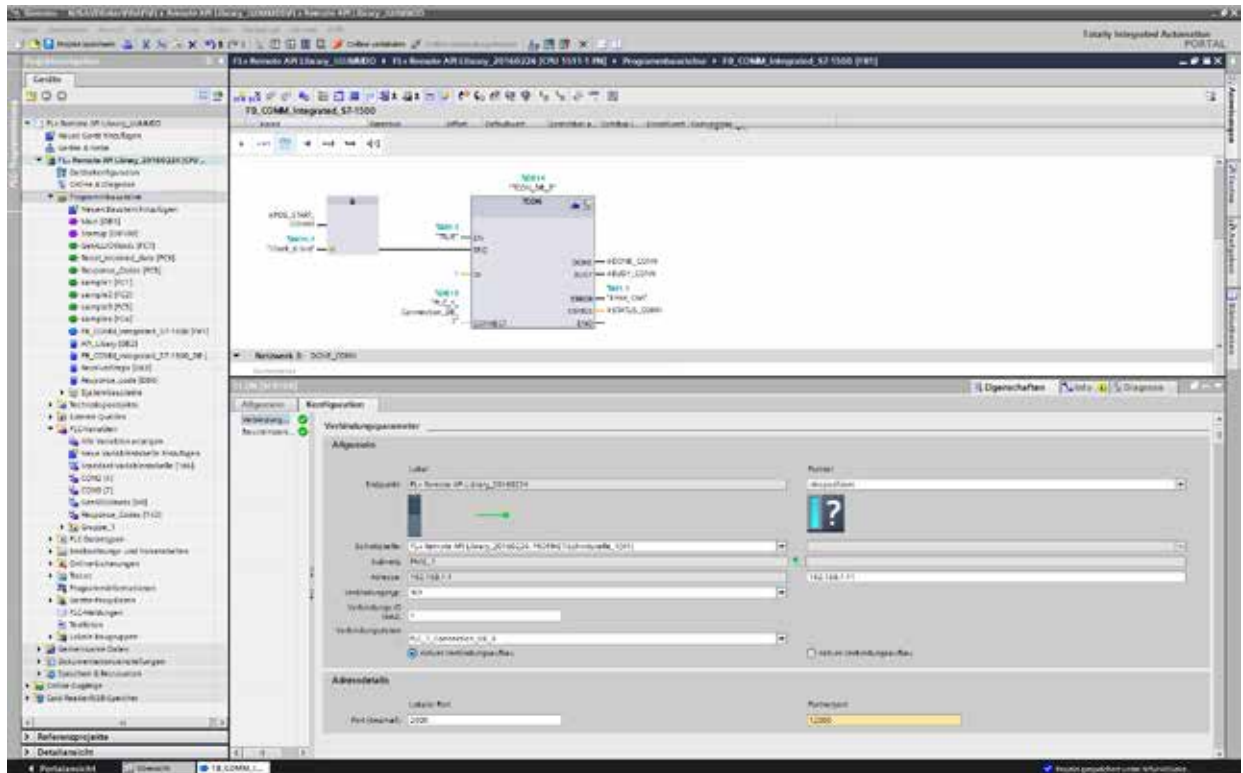
**Note!**

**If a different IP address is configured, this must be adapted in block TCON to establish a connection.**

**If communication block TCON made available by cab, attention should be paid that the data received are correctly referenced to data block DB[3].**

### 3.5.3 Establishing the Network Connection

Function block **FB1** has determined the communication block TCON for establishing the connection between PLC and laser. There is also defined how data are sent / received to / from the laser.



The function block includes the following information:

- Description of local and partner control
- Local and partner IP address
- Definition of memory area to store communication data
- Local port address
- Port address of partner



#### Note!

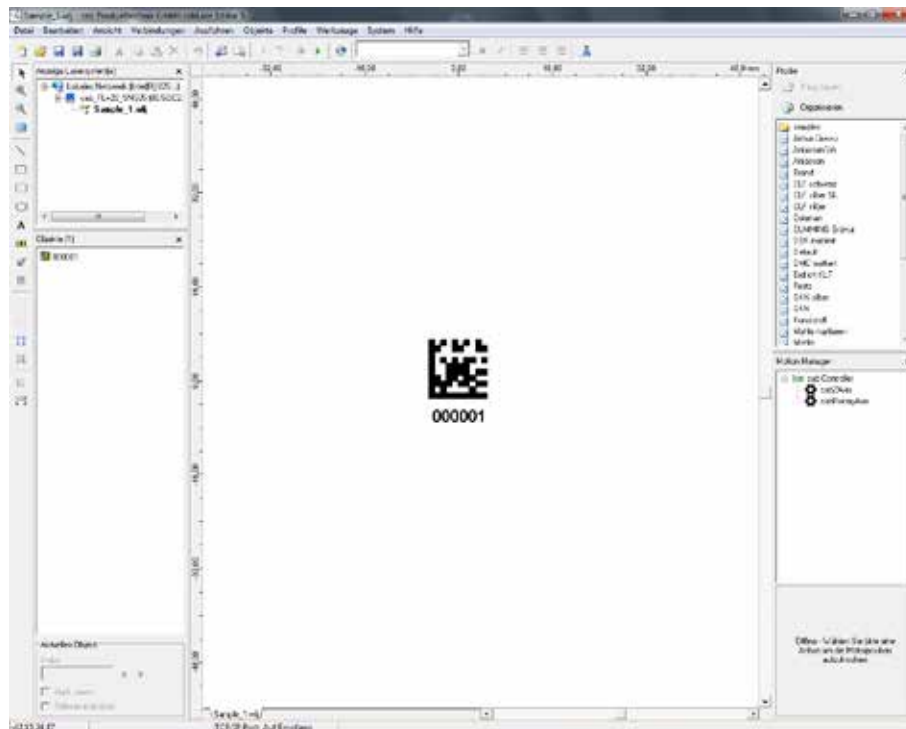
When the device is delivered, configured IP address is 192.168.1.11.

## 4.1 Sample 1

### 4.1.1 Description

Here, task is to realize the marking with a data matrix code, generating a sequential 6 digit serial number. When starting production the starting number shall be handed over and then be incremented by 1. Incrementation is done by the PLC. The layout can be found in the flash memory of the laser control under the file name Sample1.dat. When starting the program the layout is automatically loaded, edited, activated and synchronized with the digital inputs at interface CON2. Once, when starting the process the laser is switched on automatically via interface CON3 and later on controlled via CON3 and CON2.

### 4.1.2 Layout



#### Note!

If the program is working correctly the serial number is counting up during marking.

### 4.1.3 PLC Sample Program

The sample program can be found in the project folder FL+ Remote API Library\_YYYYMMDD in the function Sample1[FC1].



#### Note!

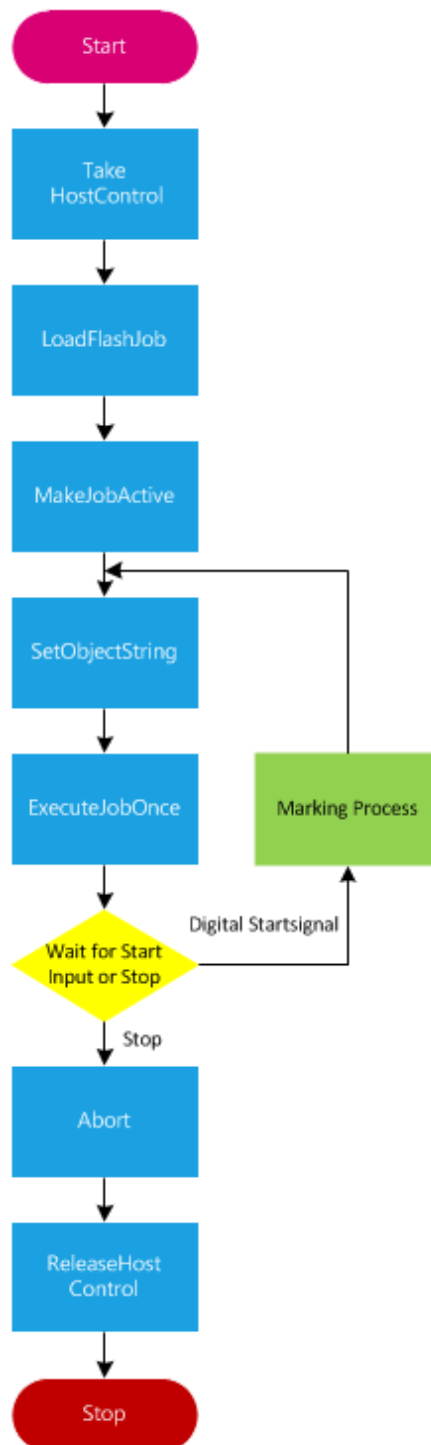
Attention should be paid that the entry condition "Start Sample" is provided in Network4. This condition must be activated to allow access to the laser and to apply the sample program.



#### Note!

After the release of „Start Sample“ access to the laser via cabLase 5 is not possible.

## 4.1.4 Flowchart



**Note!**  
Blue elements equal the Remote API commands.

## 4.2 Sample 2

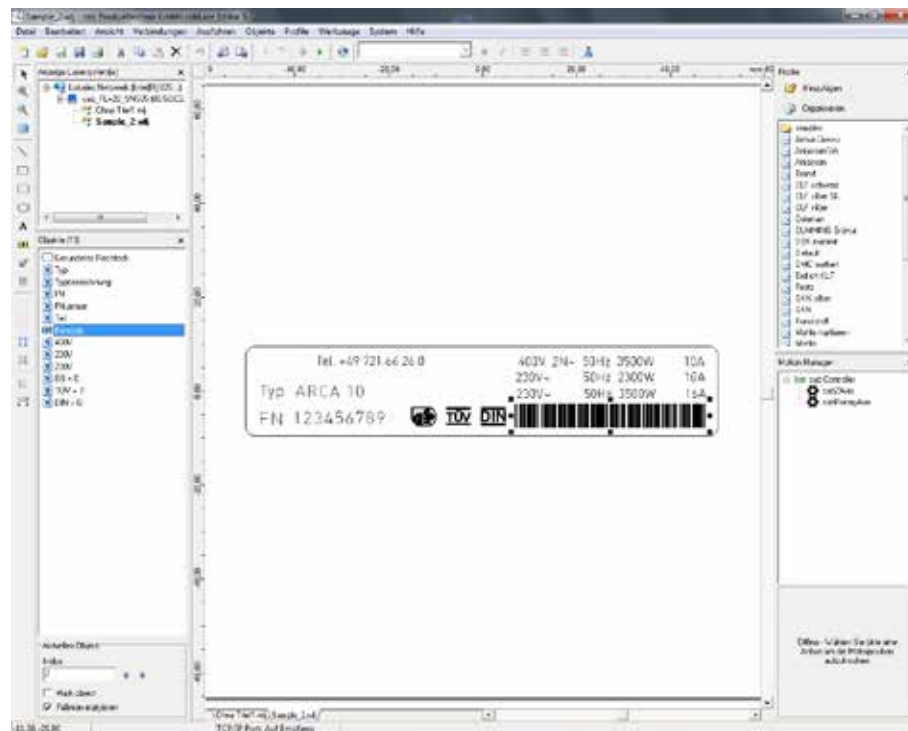
### 4.2.1 Description

Here, task is to realize the marking of a typeplate containing performance data and test certificates GS, TÜV and DIN. Depending on the production control, the various test logos shall be switched on/off.

The layout can be found in the flash memory of the laser control under Sample2.dat. When starting the program the layout is automatically loaded, edited, activated and synchronized with the digital inputs at interface CON2. At the end of the process all marking jobs are deleted from the order memory.

Once, when starting the process the laser is switched on automatically via interface CON3 and later on controlled via CON3 and CON2.

### 4.2.2 Layout



#### Note!

If the program is working correctly, the test symbols are selectively switched on and off.

### 4.2.3 PLC Sample Program

The sample program can be found in the project folder FL+ Remote API Library\_YYYYMMDD in the function Sample2[FC2].



#### Note!

Attention should be paid that the entry condition "Start Sample" is provided in Network4. This condition must be activated to allow access to the laser and to apply the sample program.

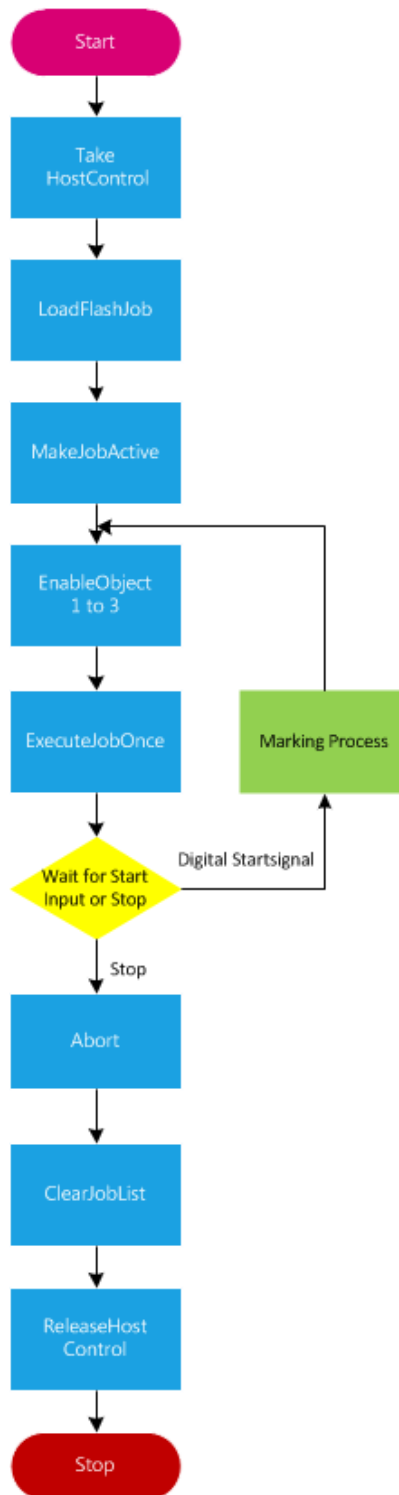


#### Note!

After the release of „Start Sample“ access to the laser via cabLase 5 is not possible.



#### 4.2.4 Flowchart



**Note!**  
Blue elements equal the Remote API commands.

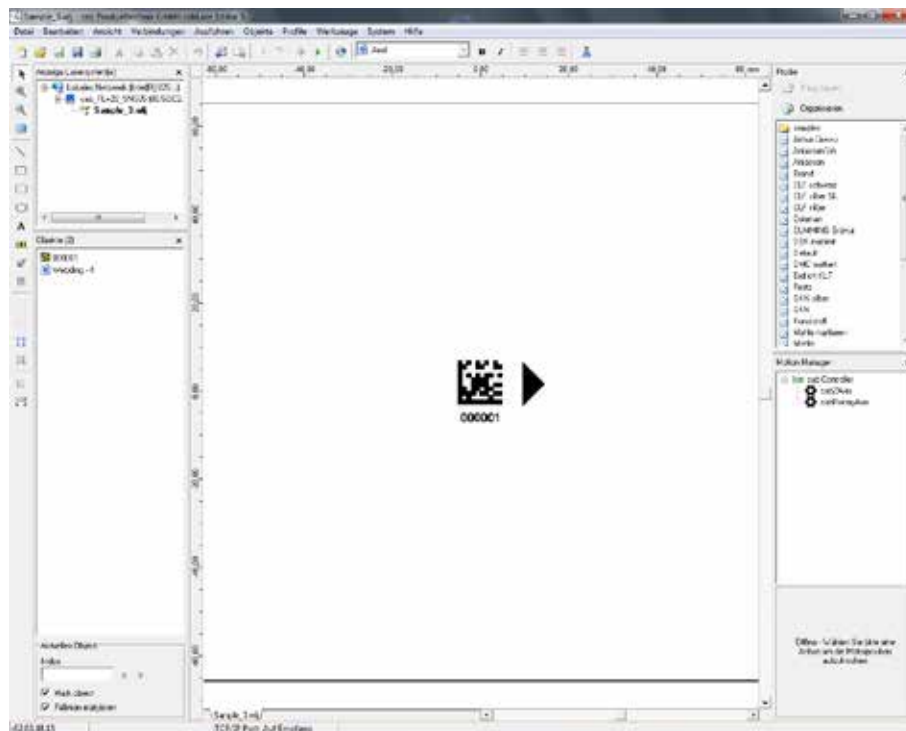
### 4.3 Sample 3

#### 4.3.1 Description

Here, task is to realize the marking of a component of different versions with respect to the positioning of the marking on the part. Depending on the model version marking of the arrow shall be moved and turned to different coordinates in x- and y direction.

The layout can be found in the flash memory of the laser control under Sample3.dat. When starting the program the layout is automatically loaded, edited, activated and synchronized via Remote API by and the command GetAllIOWord. Once, when starting the process the laser is switched on automatically via interface CON3 and later on controlled via CON3 and CON2.

#### 4.3.2 Layout



**Note!**

If the program is working correctly the arrow moves anticlockwise around the code.

#### 4.3.3 PLC Sample Program

The sample program can be found in the project folder FL+ Remote API Library\_YYYYMMDD in the function Sample3[FC3].



**Note!**

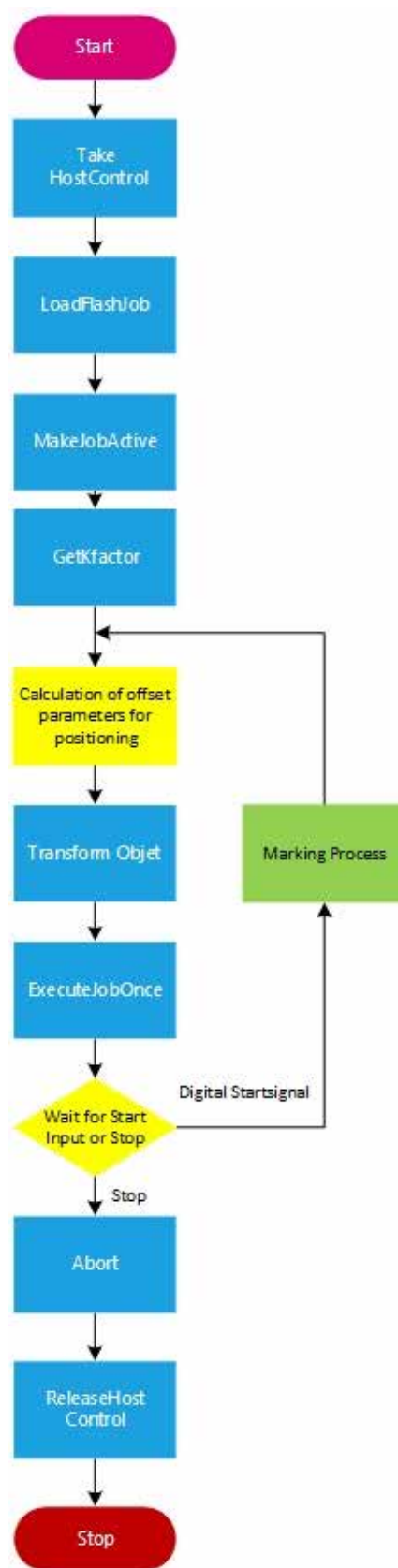
Attention should be paid that the entry condition "Start Sample" is provided in Network4. This condition must be activated to allow access to the laser and to apply the sample program.



**Note!**

After the release of „Start Sample“ access to the laser via cabLase 5 is not possible.

## 4.3.4 Flowchart



**Note!**  
Blue elements equal the Remote API commands.

## 4.4 Sample Program 4

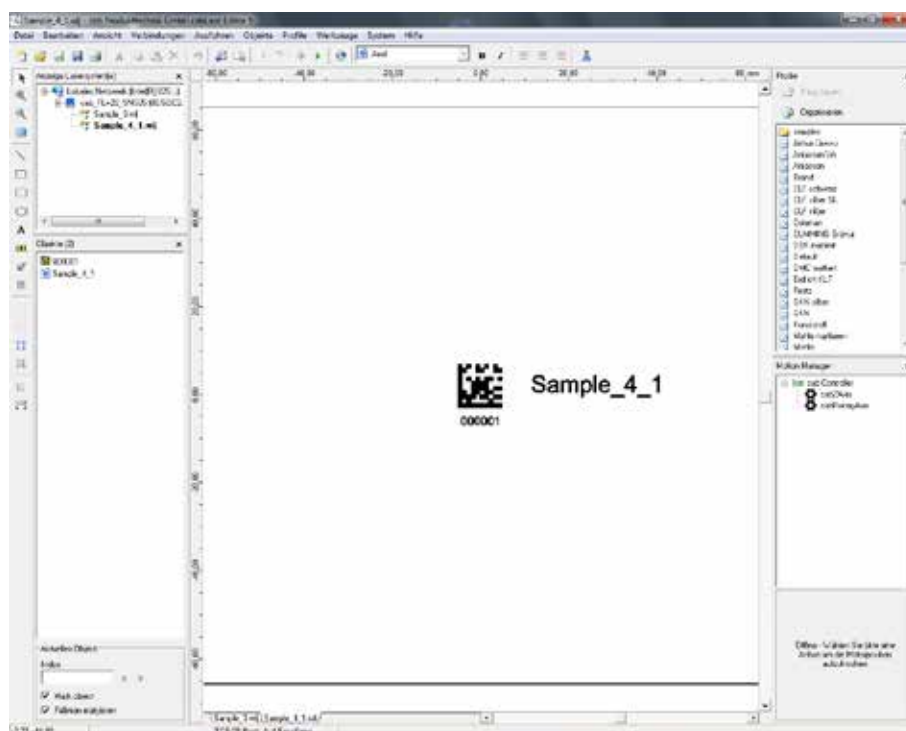
### 4.4.1 Description

Here, task is to realize the marking of different components in a production line. Depending on the model the various marking layouts shall be loaded from the company network.

The different layouts can be found in the network \\Datenserver\\Layouts under Sample4\_1.dat, Sample4\_2.dat and Sample4\_3.dat. Depending on the selection made in the production the layouts are automatically loaded, edited, activated and synchronized with the digital inputs at interface CON2.

Once, when starting the process the laser is switched on automatically via interface CON3 and later on controlled via CON3 and CON2.

### 4.4.2 Layout



#### Note!

If the program is working correctly the text next to the data matrix code changes from Sample\_4\_1 to Sample\_4\_2 and Sample\_4\_3.

### 4.4.3 PLC Sample Program

The sample program can be found in the project folder FL+ Remote API Library\_YYYYMMDD in the function Sample4[FC4].



#### Note!

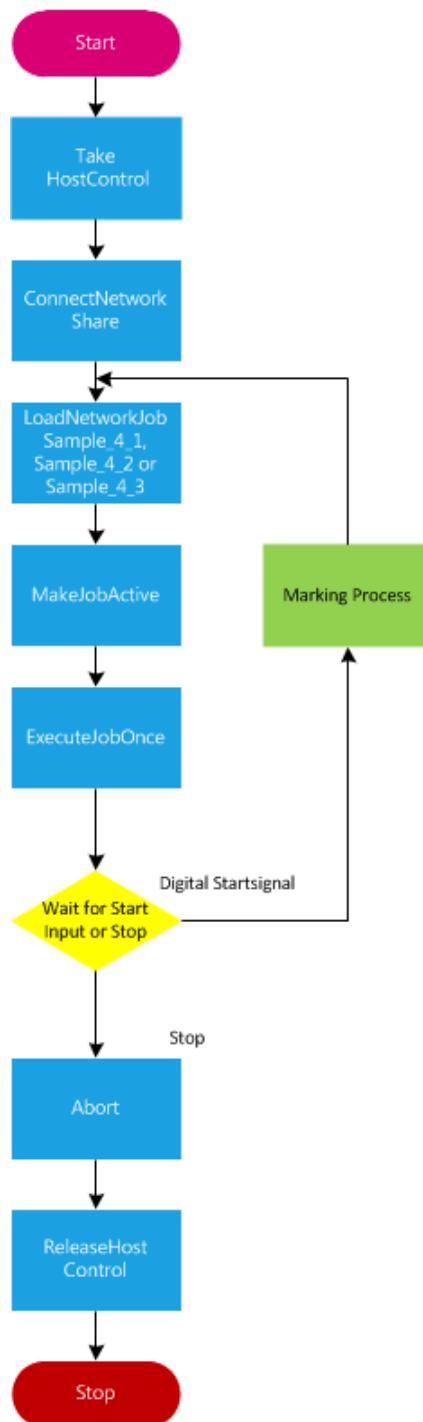
Attention should be paid that the entry condition "Start Sample" is provided in Network4. This condition must be activated to allow access to the laser and to apply the sample program.



#### Note!

After the release of „Start Sample“ access to the laser via cabLase 5 is not possible.

## 4.4.4 Flowchart



**Note!**  
Blue elements equal the Remote API commands.

## 5.1 Selection of Network Adapter

cabLase Editor 5 supports several network adapters installed in the local PC.

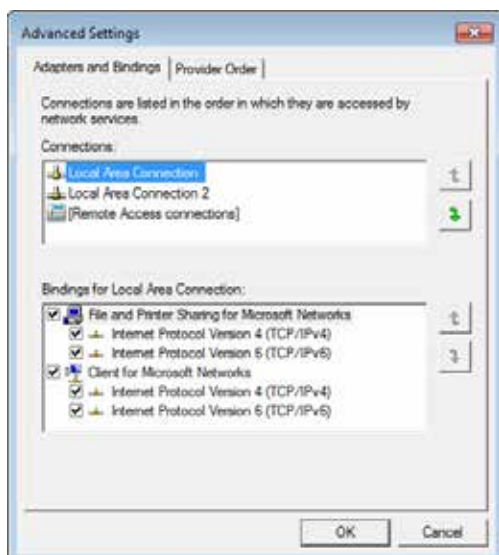


### Note!

In the event that there are more network adapters available, connection sequence for the network adapter under Windows shall be so that the laser marker FL+ is the first to be connected! [▷ Documentation Windows](#)

### Example Windows 7

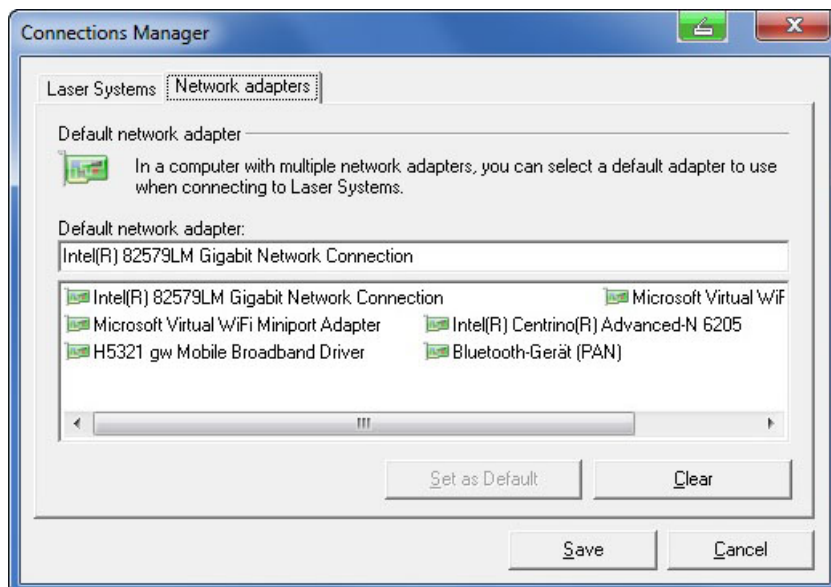
- ▶ Select "Control panel > Network and sharing center > Change adapter settings".
- ▶ Select menu "Advanced > Advanced settings".
- ▶ Select under "Connections" the connection that is used for connecting the FL+.
- ▶ Use the arrow keys next to the window to move the connection selected to the top line.
- ▶ Click on "OK".



Priority setting of the network adapter

The network connection connected to the device must be set as default in cabLase Editor 5.

- ▶ Select "Connections > Manage" in the main menu bar.
- ▶ Select the tab "Network adapters".



Selection of network adapter

- ▶ Select the network adapter to which the FL+ is connected.
- ▶ Click on "Set as default".



**Note!**

If the subnets are incompatible the marking laser FL+ adds a temporary IP address to the default PC network adapter. Once the PC is restarted this temporary address will be deleted.

## 5.2 Manually Connecting the Laser with cabLase Editor 5



### Note!

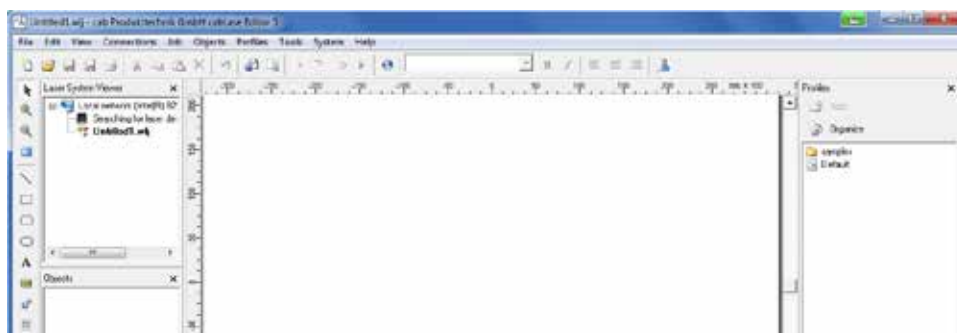
Before connecting cabLase Editor 5 with the marking laser FL+ a network connection must be created.



### Note!

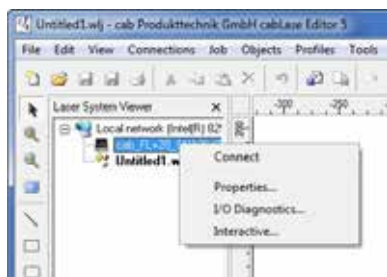
Make sure that the marking laser FL+ has been switched on at the power switch!

After having started the software the message "Searching for laser devices" under the window "Laser System Viewer" is displayed.



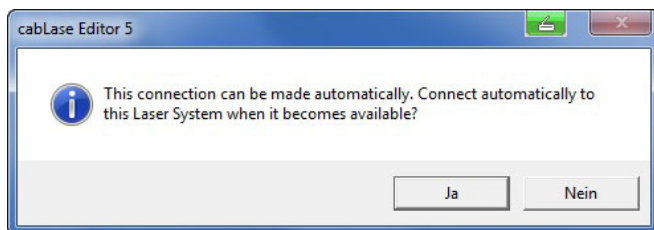
Search for laser devices

- Right-click within the window "Laser System Viewer", select the marking laser FL+ detected in the network and click on "Connect".



Connecting the laser device

This is followed by a query to set up an automatic connection:



Query automatic connection

- Select the method to connect required.

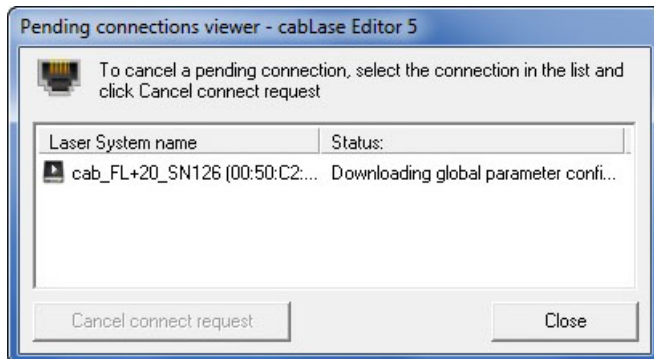


### Note!

Setting up the automatic connection can be carried out even later.



This is followed by transmitting the configuration settings of the marking laser FL+ to the local PC. The downloading status is shown in the "Pending connections viewer".



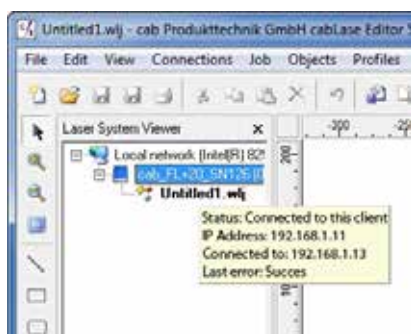
Pending connections viewer

The following icons are used to show different states of connection:

Symbol	Status
	Available in the network
	Not available in the network
	Connected to local installation cabLase Editor 5
	Invalid firmware or license
	PC software incompatible

Tabelle 1 States of connection

Moving the mouse cursor over the laser device detected shows its status of connection:



Display status of connection

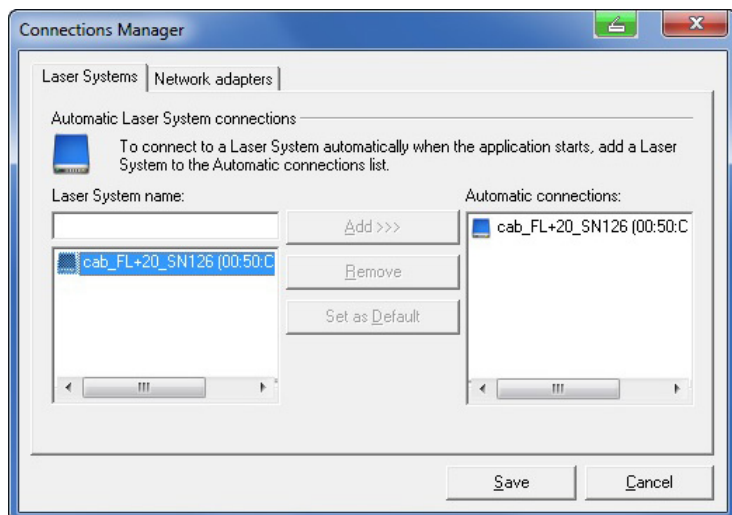


#### Note!

Click on "Disconnect" to close an active connection.

### 5.3 Automatically Connecting the Laser with cabLase Editor 5

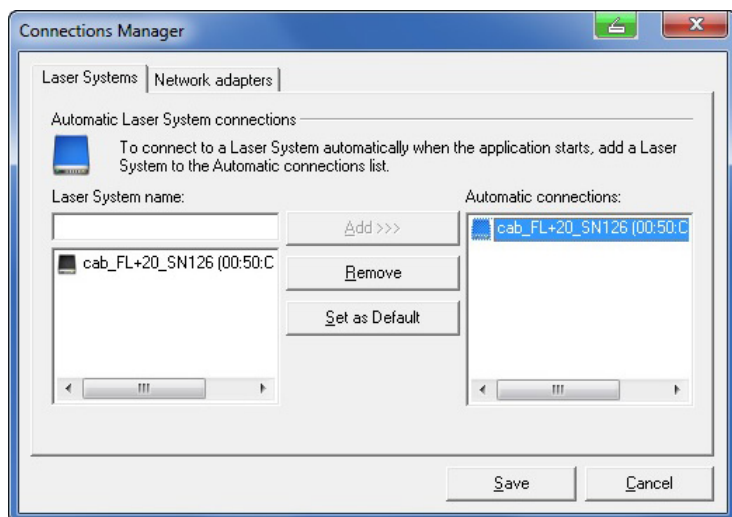
- Select menu "Connections" > Manage". This is followed by the input/output window:



Set up automatic connection

The left windows shows all marking lasers FL+ available in the network, each with their respective MAC addresses.

- Click on the marking laser FL+ to be connected:  
The marking laser selected is highlighted in blue.
- Click on "Add".  
The marking laser is added to the list "Automatic connections".



Set marking laser as default

- Select the marking laser required in the list "Automatic connections" and confirm by clicking on "Set as default".  
Selection is confirmed by a check mark in the PC icon.
- Click on "Save" to confirm the settings selected.



#### Note!

When starting the software, cabLase Editor 5 now always set up automatically connection with the marking laser FL+ selected.

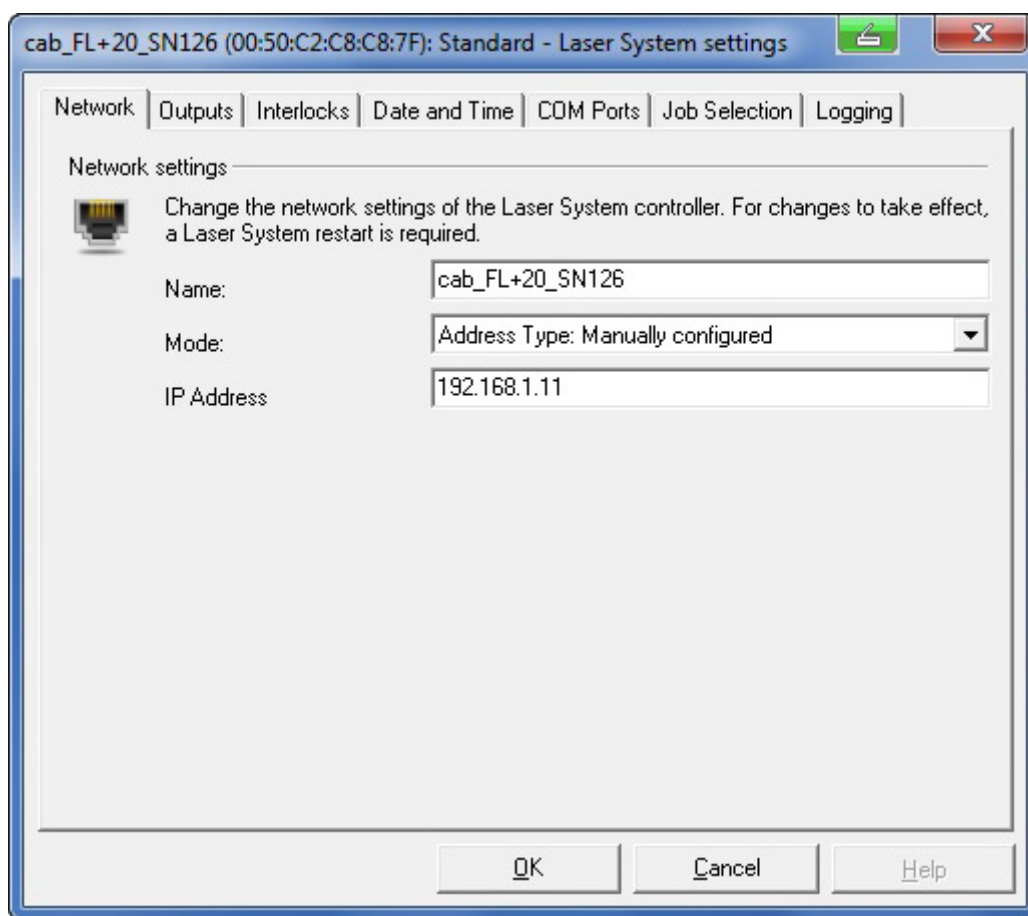
## 5.4 Changing the IP Address



### Attention!

Incorrect settings of the IP address may result in losing connection and only might be restored by changing the settings at the local PC

- ▶ Select and right-click within the window on the marking laser FL+ detected and click on "Default settings".
- ▶ Select "Network".



Setting the IP address

- ▶ Select under mode "Address type: Manually configured" and enter the IP address to be used.

## 6.1 General

The marking laser FL+ provides the ability to mark jobs without data transmission from the PC (stand-alone operation). In this case, a higher-level control (e.g. PLC) completely handles the operation of the marking laser FL+. For this purpose, the "Remote Command API" (Application Programming Interface) and "COM Automation Server" (► Scope of delivery software) are available and additionally allow access to stored data via the "Job Select" signal. To operate the marking laser FL+ in the stand-alone operation job files and laser character fonts need to be saved in binary format in the memory of the FL+ controller and called via subroutine or system explorer.



### Note!

Editing and modifying is only possible with \*.wlj files.

When transmitting data into the memory of the marking laser FL+, these are automatically converted via cabLase Editor 5 into binary \*.dat job files.

The laser system explorer can be used to copy available \*.dat files from the marking laser FL+ to a local PC.

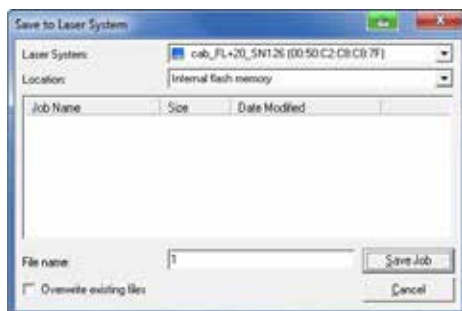
## 6.2 Storing \*.wlj files to the Memory of the Laser

The example requires two job files on the PC, named 1.wlj and 2.wlj.



Selection destination data storage

- Right-click within the window "Laser System Viewer" on an opened job.
- Select via "Assign to" the active marking laser FL+ as destination for data storage.
- Copy the job via "Save to marking laser" Job into the memory of the marking laser.



Setting file name

- Fix a name for the file and click on "Save job".  
The example files are converted into binary \*.dat job files and stored as 1.dat, respectively 2.dat in the memory of the marking laser. However, these data are a lot larger than the original \*.wlj files.

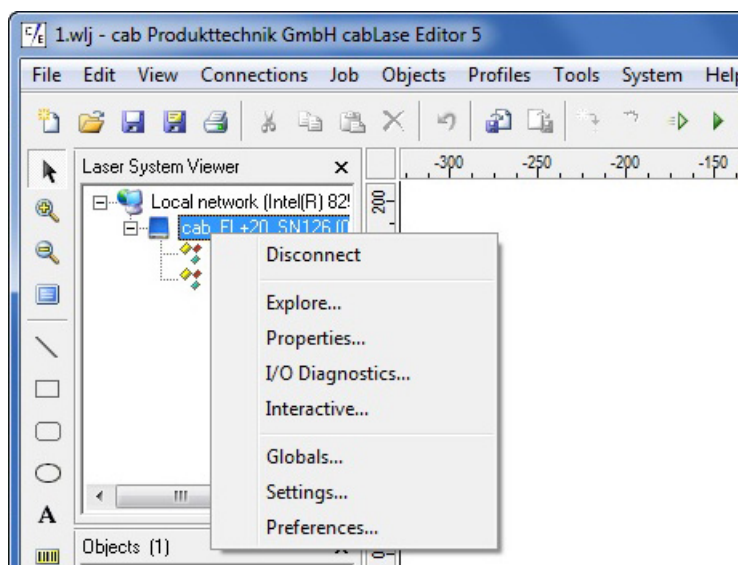


### Note!

To call marking layouts stored in the internal memory via digital coding, the input "Job Select" on CON2 of the marking laser FL+ needs to be activated!

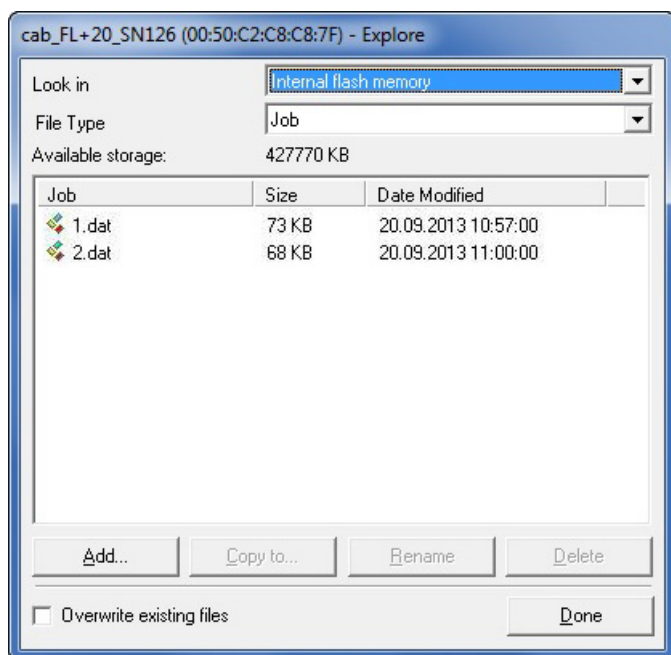
Access via "Remote Command API" and "COM Automation Server" is always possible.

### 6.3 Managing \*.dat Files



Calling the system explorer

- Click within the window "Laser System Viewer" on active device and select "Explore..."

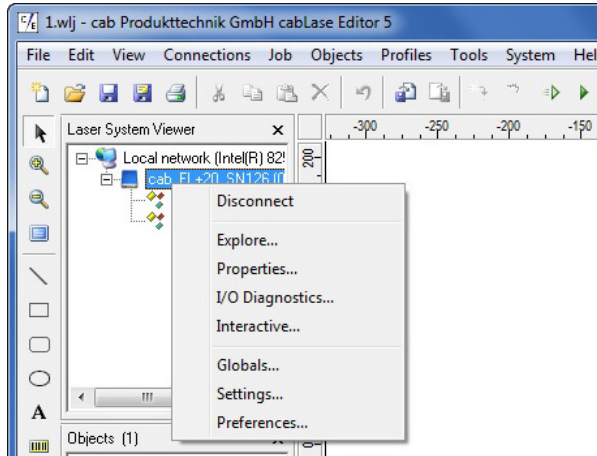


Display list job files

- Select "Internal flash memory" and type of file "Job" to show stored .dat files.
- Select "Copy to..." to copy files from the marking laser FL+ to the local PC.
- Select "Add.." to copy .dat files already available on the local PC to the marking laser FL+.

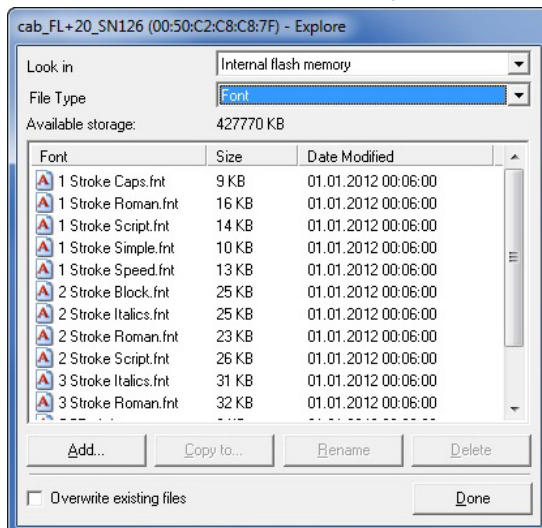
## 6.4 Storing Laser Font Files

The following example uses text objects in the job files called 1.wlj and 2.wlj, respectively 1.dat and 2.dat created with the with the Arial font type. This font type needs to be loaded into the memory of the laser.



Calling the system explorer

- Click within the window "Laser System Viewer" on active device and select "Explore..."



Display list font files

- Select "Internal flash memory" and type of file "Font" to show laser fonts available.
- Select "Add..." to copy laser font from the local PC to the marking laser FL+.
- Select "Copy to..." to copy laser font from the marking laser FL+ to the local PC.

### Note!

Default directory for laser font files installed on the local PC is: c:\marker\marker\fonts.  
Further information about converting TrueType fonts into laser fonts ► Documentation software.

### Note!

To be able to use a layout via Remote API interface, it is necessary to check before, whether the fonts used in the layout are stored on the laser control.

### Note!

To be able to use the fonts loaded, the laser system must be newly initialized. There are different possibilities to do this:

- Switch the laser off and on again
- Carry out a reset at the operation panel
- Release a reset from the external control vial CON3 - PIN10

## 7.1 API Command Set

The interface provided by the Remote Command API is a message based protocol. All command strings must be terminated by a Carriage Return (0X0D) and Line Feed (0X0A) to allow a successful transfer.

The following list describes a selection of the most important Remote Interface commands and their intended use. Please refer to our English Remote API Library for a complete list of all commands.

Commands with multiple parameters are sent to the Host in comma-delimited format. And, in case that responses are sent with multiple parameters these are also returned in comma-delimited format.



### Note!

All commands must be terminated by „Carriage Return“ and „Line Feed“ (0X0D & 0X0A / <CR>&<LF>).



### Note!

Commands with multiple parameters are sent to the Host in comma-delimited format.



### Note!

Responses with multiple parameters are returned in comma-delimited format.



### Note!

All responses must be terminated by „Carriage Return“ and „Line Feed“ (0X0D & 0X0A / <CR>&<LF>).

## 7.1 Abort

Command	Abort
Purpose	Stops the execution of a job
Implementation	1 or 1,blocking
Parameter	Parameter blocking: (integer value) 0 - to immediately stop, f. ex.: 1,0 1 - to stop after having finished the current job, f. ex.: 1,1
API Return messages	0 = Success, 1 = Idle
Comments	There are two possibilities to perform the command Abort. The first task is to immediately stop the execution of an in-process job. The second task is to stop after having completed and processed all queued data. Using the parameter 0, for immediate stop, the laser stops without regard to loss of data. Using parameter 1 the command waits until the whole process is completed. When calling 1, the shorter syntax version of the command, the parameter is automatically set to 1.



### Note!

Always use parameter blocking=1, to allow a synchronized processing and clear data assignment (serial number / tracking and tracing).

## 7.2 ClearJobList

Command	ClearJobList
Purpose	Removes all loaded jobs from memory.
Implementation	200
Parameter	-
API Return messages	0 = Success
Comments	Used to delete all marking jobs from the order memory. The command also applies to the currently in-process job.



### Note!

By using ClearJobList the marking job activated via MakeJobActive is deleted from the memory.

## 7.3 ConnectNetworkShare

Command	ConnectNetworkShare
Purpose	Makes a connection to a network resource
Implementation	523,Remotesharename,Username,Password
Parameter	Remotesharename: Specifies the network resource to connect to (256 characters maximum). The Remotesharename has to be entered in an exemplary manner in the format \\ServerDaten\Laserstation. Username: Specifies the user name for making the connection (49 characters maximum) Password: Specifies the password to be used when making the connection (63 characters maximum)
API Return messages	If the command fails because of a network connectivity error (NetworkConnectFail), the response is in the form 33,extendederrorinfo. The error messages are created by the Windows API. For example, if the call fails because of an access denied, the response will be 33,5. Please refer to the Windows documentation for more details.
Comments	This command only supports the "Microsoft Windows Network" provider. The share cannot be located directly on a Domain.



### Note!

Complete functionality of the command in the area "extendederrorinfo" is only provided with Windows-compatible networks.

## 7.4 Echo

Command	Echo
Purpose	Delivers a return message of the control on any string
Implementation	37,String
Parameter	String: A string to send that will echoed back by the laser control (3000 characters maximum).
API Return messages	Equal to the string value sent
Comments	This command can be used to verify communications, and, in activated power saving mode, for keep-alive purposes of the control.



## 7.5 EnableObject

Command	EnableObject
Purpose	Enable / disable execution of the specified object
Implementation	106, Objectindex, State
Parameter	Objectindex: Equals to the integer object index State: 0 means, the object will not be marked 1 means, the object will be marked
API Return messages	0 = Success
Comments	To execute this command the marking job must not be in process.



### Note!

In case of disabling a large number of objects it might be better to reload the layout instead of reactivating the objects.



### Note!

Object index equals to the object number visualized in cabLase, reduced by 1.

## 7.6 ExecuteJobContinuous

Command	ExecuteJobContinuous
Purpose	Starts the execution of the active job and will execute the job in an infinite loop.
Implementation	208, cacheobjects
Parameter	Cacheobjects: (Integer value 0 or 1) 0 does not cache marking objects, but wait for the StartMark from the internal FIFO memory signal 1 cache objects in the FIFO memory immediately after the command for marking is called
API Return messages	0 = Success, 2 = Busy, 109 = NoActiveJob
Comments	The client must call TakeHostControl before making this call. The marking job has to be loaded in the job memory of the laser control and activated by MakeJobActive. The control waits to start marking until a digital start marking signal is received and will continue repeating this process until the command Abort is called.  Cacheobjects = 0 and cacheobjects = 1 differ from each other as to the time in which objects and commands for the FIFO memory are processed and sent. When calling cacheobjects = 1 the control does not wait for the external start signal.  Calling cacheobjects = 0 is useful, if for example objects in the layout shall be controlled by digital input signals. Usual case is cacheobjects = 1.



### Note!

In the event of using ExecuteJobContinuous the external start signal must be activated when creating the marking layout.



### Note!

In case of the repeatedly marking of a job the command must be called again.

## 7.7 ExecuteJobOnce

Command	EnableObject
Purpose	Starts the execution of the active job and runs it once without repeat
Implementation	207,cacheobjects
Parameter	Cacheobjects: (Integer value 0 or 1) 0 do not cache marking objects, but wait for the StartMark from the internal FIFO memory signal 1 cache objects in the FIFO memory immediately after the command for marking is called
API Return messages	0 = Success, 2 = Busy, 109 = NoActiveJob
Comments	After having called the command the control will wait to start marking until an external start signal at Interface CON2 is received and will continue repeating this process until the command „Abort“ is called, even if the marking has not yet started. Calling cacheobjects = 0 is useful, if for example objects in the layout shall be controlled by digital input signals. Usual case is cacheobjects = 1.

**Note!**

In the event of using **ExecuteJobOnce** the external start signal must be activated when creating the marking layout.

**Note!**

In case of the repeatedly marking of a job the command must be called again.

## 7.8 GetAllIOWords

Command	GetAllIOWords																																																				
Purpose	Delivers the state of all the digital inputs and outputs as two WORDS.																																																				
Implementation	31																																																				
Parameter	-																																																				
API Return messages	<p>StandardWord,ExtendedWord</p> <p>The API delivers an 18-bit word followed by a 32-bit word  The StandardWord delivers the in- and outputs of the standard functions  The ExtendedWord delivers the in- and outputs of the extension</p> <p>StandardWord:</p> <table> <tr><td>Bit 0: User In 1</td><td>Bit 10: User Out 1</td></tr> <tr><td>Bit 1: User In 2</td><td>Bit 11: User Out 2</td></tr> <tr><td>Bit 2: User In 3</td><td>Bit 12: User Out 3</td></tr> <tr><td>Bit 3: User In 4</td><td>Bit 13: User Out 4</td></tr> <tr><td>Bit 4: Start Marking</td><td>Bit 14: Marking in Progress</td></tr> <tr><td>Bit 5: Job Load</td><td>Bit 15: Job Busy</td></tr> <tr><td>Bit 6: Interlock 1</td><td>Bit 16: System Error</td></tr> <tr><td>Bit 7: Interlock 2</td><td>Bit 17: Ready</td></tr> <tr><td>Bit 8: Interlock 3</td><td></td></tr> <tr><td>Bit 9: Interlock 4</td><td></td></tr> </table> <p>ExtendedWord:</p> <table> <tr><td>Bit 0: User In 5</td><td>Bit 16: User Out 5</td></tr> <tr><td>Bit 1: User In 6</td><td>Bit 17: User Out 6</td></tr> <tr><td>Bit 2: User In 7</td><td>Bit 18: User Out 7</td></tr> <tr><td>Bit 3: User In 8</td><td>Bit 19: User Out 8</td></tr> <tr><td>Bit 4: User In 9</td><td>Bit 20: User Out 9</td></tr> <tr><td>Bit 5: User In 10</td><td>Bit 21: User Out 10</td></tr> <tr><td>Bit 6: User In 11</td><td>Bit 22: User Out 11</td></tr> <tr><td>Bit 7: User In 12</td><td>Bit 23: User Out 12</td></tr> <tr><td>Bit 8: User In 13</td><td>Bit 24: User Out 13</td></tr> <tr><td>Bit 9: User In 14</td><td>Bit 25: User Out 14</td></tr> <tr><td>Bit 10: User In 15</td><td>Bit 26: User Out 15</td></tr> <tr><td>Bit 11: User In 16</td><td>Bit 27: User Out 16</td></tr> <tr><td>Bit 12: User In 17</td><td>Bit 28: User Out 17</td></tr> <tr><td>Bit 13: User In 18</td><td>Bit 29: User Out 18</td></tr> <tr><td>Bit 14: User In 19</td><td>Bit 30: User Out 19</td></tr> <tr><td>Bit 15: User In 20</td><td>Bit 31: User Out 20</td></tr> </table>	Bit 0: User In 1	Bit 10: User Out 1	Bit 1: User In 2	Bit 11: User Out 2	Bit 2: User In 3	Bit 12: User Out 3	Bit 3: User In 4	Bit 13: User Out 4	Bit 4: Start Marking	Bit 14: Marking in Progress	Bit 5: Job Load	Bit 15: Job Busy	Bit 6: Interlock 1	Bit 16: System Error	Bit 7: Interlock 2	Bit 17: Ready	Bit 8: Interlock 3		Bit 9: Interlock 4		Bit 0: User In 5	Bit 16: User Out 5	Bit 1: User In 6	Bit 17: User Out 6	Bit 2: User In 7	Bit 18: User Out 7	Bit 3: User In 8	Bit 19: User Out 8	Bit 4: User In 9	Bit 20: User Out 9	Bit 5: User In 10	Bit 21: User Out 10	Bit 6: User In 11	Bit 22: User Out 11	Bit 7: User In 12	Bit 23: User Out 12	Bit 8: User In 13	Bit 24: User Out 13	Bit 9: User In 14	Bit 25: User Out 14	Bit 10: User In 15	Bit 26: User Out 15	Bit 11: User In 16	Bit 27: User Out 16	Bit 12: User In 17	Bit 28: User Out 17	Bit 13: User In 18	Bit 29: User Out 18	Bit 14: User In 19	Bit 30: User Out 19	Bit 15: User In 20	Bit 31: User Out 20
Bit 0: User In 1	Bit 10: User Out 1																																																				
Bit 1: User In 2	Bit 11: User Out 2																																																				
Bit 2: User In 3	Bit 12: User Out 3																																																				
Bit 3: User In 4	Bit 13: User Out 4																																																				
Bit 4: Start Marking	Bit 14: Marking in Progress																																																				
Bit 5: Job Load	Bit 15: Job Busy																																																				
Bit 6: Interlock 1	Bit 16: System Error																																																				
Bit 7: Interlock 2	Bit 17: Ready																																																				
Bit 8: Interlock 3																																																					
Bit 9: Interlock 4																																																					
Bit 0: User In 5	Bit 16: User Out 5																																																				
Bit 1: User In 6	Bit 17: User Out 6																																																				
Bit 2: User In 7	Bit 18: User Out 7																																																				
Bit 3: User In 8	Bit 19: User Out 8																																																				
Bit 4: User In 9	Bit 20: User Out 9																																																				
Bit 5: User In 10	Bit 21: User Out 10																																																				
Bit 6: User In 11	Bit 22: User Out 11																																																				
Bit 7: User In 12	Bit 23: User Out 12																																																				
Bit 8: User In 13	Bit 24: User Out 13																																																				
Bit 9: User In 14	Bit 25: User Out 14																																																				
Bit 10: User In 15	Bit 26: User Out 15																																																				
Bit 11: User In 16	Bit 27: User Out 16																																																				
Bit 12: User In 17	Bit 28: User Out 17																																																				
Bit 13: User In 18	Bit 29: User Out 18																																																				
Bit 14: User In 19	Bit 30: User Out 19																																																				
Bit 15: User In 20	Bit 31: User Out 20																																																				
Comments	The ExtendedWord for the cab FL+ is only valid until input 8 and output 8.																																																				



### Note!

In the event of calling ExtendedWord, bits 4-15 and bits 20-31 are not active for the FL+.

## 7.9 GetFlashJobFileList

Command	GetFlashJobFileList
Purpose	Delivers a comma-delimited list of all jobs stored in the flash memory of the laser control
Implementation	203
Parameter	-
API Return messages	0,job1,jobX job1 is equal to the 1st name of the marking job in the memory jobX is equal to the umpteenth name of the marking job in the memory
Comments	The internal flash memory is the system-internal memory at the laser control. Reading from and writing to the memory may be effected via cabLase or in the FTP mode.



### Note!

Please refer to Chapter 6.2 for more information about reading and from and writing marking jobs to the internal flash memory.



### Note!

Should the internal flash memory is not readable, the control delivers an API response ErrorCode.

## 7.10 GetFontFileList

Command	GetFontFileList
Purpose	Gets a comma-delimited list of all font files stored in the flash memory of the laser control
Implementation	48
Parameter	-
API Return messages	0,Fontfile1,...,fontfileN Fontfile1 is equal to the first font in the list FontfileN is equal to the laset font in the list
Comments	The flash memory is the internal storage memory of the laser control that is accessed via cabLase or FTP server.



### Note!

Please refer to Chapter 6.2 or to the assembly instructions of the marking laser FL<sup>+</sup> for more information about access to the internal flash memory.

### 7.11 GetKFactor

Command	GetKFactor
Purpose	Delivers the calibration factor of the current F-Theta lens configuration
Implementation	10
Parameter	-
API Return messages	KFactor: delivers the calibration factor of the lens configuration in field units (bits)/mm.
Comments	Use this command to discover the conversion between the measurement unit mm and the laser-specific field units.



**Note!**

Please keep in mind, that Platform 6 and Platform 7 boards maintain native coordinates in  $\mu\text{m}$ .

### 7.12 GetNetworkJobFileList

Command	GetNetworkJobFileList
Purpose	Gets a comma-delimited list of all marking jobs stored in a network storage location.
Implementation	221,Subfolder
Parameter	Subfolder: is the path specification for the network drive 221, transmits the data in the root directory 221,\Arbeitsdateien transmits the data in the directory \Arbeitsdateien
API Return messages	0,job1,jobX job1 is equal to the 1st name of the marking job in the network jobX is equal to the umpteenth name of the marking job in the network
Comments	A connection must already exist to the network drive by using the command ConnectNetworkShare.



**Note!**

It is obligatory to use the command ConnectNetworkShare to initialize a network connection before calling GetNetworkFileList.

### 7.13 GetObjectCenter

Command	GetObjectCenter
Purpose	Delivers the geometric center of the specified object in field units
Implementation	104,Objectindex
Parameter	Objectindex: Is the integer number of the indexed object
API Return messages	X,Y X is equal to the x coordinate of the object center, in bits Y is equal to the y coordinate of the object center, in bits
Comments	To be able to use this command, the marking job must not be in process. The marking field is set up as cartesian coordinate system originating in the center.

### 7.14 GetUSBJobFileList

Command	GetUSBJobFileList
Purpose	Gets a comma-delimited list of all marking jobs stored in a USB drive
Implementation	204
Parameter	-
API Return messages	0,job1,jobX job1 Is equal to the 1st name of the marking job in the memory jobX Is equal to the umpteenth name of the marking job in the memory
Comments	If no USB drive is found, NoDrive is returned from the control. USB drives differ in terms of format, memory size and addressing. Should you have problems in accessing please change the storage medium or contact the manufacturer.



**Note!**

If the USB storage medium can not be read an **ErrorCode** will be returned.

### 7.15 HardwareReset

Command	HardwareReset
Purpose	Performs a reset of the laser control
Implementation	8
Parameter	-
API Return messages	No response, as the laser control will be newly initialized
Comments	After receiving this command, the laser control will perform a soft reset. Before booting the socket connection to the client will be automatically closed and must be reconnected after initialization. You also need to start again with the command TakeHostControl. Any changes made to the IP address parameters will be applied at this time.



**Note!**

All jobs in the RAM will be deleted and have to be loaded and activated again. Transmitted variables are lost.



**Note!**

During the boot procedure a connection to the laser control can be established, however there is no data exchange available. Whether booting is terminated or not may be asked via status query or digital interface.



**Note!**

Initialization via HardwareReset may take up to 90 seconds.

### 7.16 LoadFlashJob

Command	LoadFlashJob
Purpose	Loads a job from flash memory into RAM, and sets the job as the ActiveJob.
Implementation	205.jobname
Parameter	jobname: (Stringwert) Is the file name of the layout with the extension, for example Circle.dat.
API Return messages	0 = Success, 3 = NoJob
Comments	There can be multiple jobs loaded in RAM simultaneously. Before interacting with a job (editing or modifying a layout), it must be made active with the MakeJobActive command.


**Note!**

Multiple marking jobs can be loaded in the RAM of the control simultaneously.


**Note!**

Before interacting with a job, for example editing or modifying variables, objects or other features), the layout must be made active with the MakeJobActive command.

### 7.17 LoadNetworkJob

Command	LoadNetworkJob
Purpose	Loads a job from a network location into RAM, and sets the job as the ActiveJob.
Implementation	222.jobname
Parameter	jobname: (Stringwert) Is the path and file name of the layout with the extension, for example \Arbeitsdateien\ Circle.dat.
API Return messages	0 = Success, 3 = NoJob
Comments	<p>The job name can contain subfolder locations relative to the connected network drive. The network drive is defined with the ConnectNetworkShare command and must not be specifically stated!</p> <p>To load the file circle.dat from the network drive server01 in the path \Arbeitsdateien\, use: 222,\Arbeitsdateien\circle.dat</p> <p>Access is made via: \\Server01\Arbeitsdateien\circle.dat</p> <p>Multiple marking jobs can be loaded in the RAM of the control simultaneously. Before interacting with a job (editing or modifying a layout), it must be made active with the MakeJobActive command.</p>


**Note!**

Network drives are defined independent of the command LoadNetworkJob with the ConnectNetworkShare command.


**Note!**

Drive names and server are not specified in the jobname, but only the relative subfolders.

**Note!**

There can be multiple jobs loaded in RAM simultaneously.

**Note!**

Before interacting with a job (editing or modifying a layout), it must be made active with the **MakeJobActive** command.

## 7.18 LoadUSBJob

Command	LoadUSBJob
Purpose	Loads a marking job from an external USB drive into RAM
Implementation	206,jobname
Parameter	jobname: (Stringwert) Is the file name of the layout with the extension, for example Circle.dat
API Return messages	0 = Success, 3 = NoJob
Comments	Multiple marking jobs can be loaded in the RAM of the control simultaneously. Before interacting with a job (editing or modifying a layout), it must be made active with the <b>MakeJobActive</b> command.

**Note!**

There can be multiple jobs loaded in RAM simultaneously.

**Note!**

Before interacting with a job (editing or modifying a layout), it must be made active with the **MakeJobActive** command.



### 7.19 MakeJobActive

Command	MakeJobActive
Purpose	Sets a job currently loaded into RAM as the ActiveJob
Implementation	201,jobname
Parameter	Jobname: (Stringwert) Is the file name of the layout with the extension, for example Circle.dat.
API Return messages	0 = Success, 2 = Busy, 3 = NoJob
Comments	Before changes, i.e. variables can be accepted, the layout must first be activated. If there are multiple layouts in RAM, the marking job that is activated with the command will be processed.



#### Tip!

Multiple marking jobs can be loaded in RAM simultaneously. Switching with MakeJobActive minimizes the processing time.

### 7.20 RemoveJob

Command	RemoveJob
Purpose	Deletes the current marking job from the memory.
Implementation	202
Parameter	-
API Return messages	0 = Success, 3 = NoJob
Comments	The routine allows deleting the current marking job in RAM activated with the MakeJobActive command. To delete several marking jobs, these must be activated separately.



#### Note!

RemoveJob deletes the activated marking jobs from RAM, but not the marking layouts which are stored at the laser control.

### 7.21 RemoveObject

Command	RemoveObject
Purpose	Deletes an object in the currently activated marking job
Implementation	141,objectindex
Parameter	objectindex: (Integerwert) Is the consecutive number of the object in the marking layout.
API Return messages	0 = Success, 10 = ArgOutOfRange, 104 = NoObject
Comments	Executing the command is only possible, if there is no marking job in process.



#### Note!

To recall the objects it is necessary to reload and activate the layout again.

## 7.22 ReleaseHostControl

Command	ReleaseHostControl
Purpose	Closes the connection of a client established to the host and resets the control back to the stream mode
Implementation	3
Parameter	-
API Return messages	0 = Success
Comments	Does a remote API client break the connection to the control, the control will be automatically switched into stream mode. This kind of data transferring interface is used with the help of the marking software cabLase.



### Note!

If multiple clients shall execute data exchange with a laser control, it is important to have each of the instances terminated with ReleaseHostControl.

## 7.23 ResetObject

Command	ResetObject
Purpose	Deletes the vector list of an object after a transformation and restores the original, initial vectors.
Implementation	111,objectindex
Parameter	Objectindex: Equals to the integer object index
API Return messages	0 = Success, 104 = NoObject
Comments	To be able to execute the command the marking job must not be in process. This command deletes the vectors of the transformed objects and creates the vector list of an object in its original status without transformation.



### Note!

This command deletes the vectors generated after the transformation based on the current value of the object, for example serial number. Should the original status be restored, it is necessary to reload the marking job.

## 7.24 ResetUserTransform

Command	ResetUserTransform
Purpose	Resets a object transformation and restores the original status
Implementation	112,objectindex
Parameter	Objectindex: Equals to the integer object index
API Return messages	0 = Success, 104 = NoObject
Comments	To be able to execute the command the marking job must not be in process. This command cancels the transformation. To additionally delete the generated vector list corresponding to the object, it is necessary to additionally apply the command ResetObject.



### Note!

For deleting the active vector list of an object already transformed it is necessary to use the command Reset-Object additionally to the commandResetUserTransform.

## 7.25 SetExternalStartMode

Command	SetExternalStartMode
Purpose	Sets the current ExternalStart mode of the active job.
Implementation	215,Mode
Parameter	Mode: 0 = Starts, if input is set to High 1 = Starts, if input is set to Low 2 = Starts after transition from Low to High 3 = Starts after transition from High to Low
API Return messages	0 = Success, 3 = NoJob
Comments	To be able to execute the command the marking job must not be in process. The command controls what type of signal transition on the Start Process at the digital interface CON2 will trigger the start of job execution.



### Note!

Generally, the digital start signal at interface CON2 is used to start a synchronized process. If mode = 1 is used, the execution of the marking job already starts with the ExecuteJobOnce command.

## 7.26 SetObjectString

Command	SetObjectString
Purpose	Sets the string value of a string based marking object contained in the active job
Implementation	100,objectindex,newstring
Parameter	objectindex: (integer value) Specifies the respective index number of the object from the layout  newstring: (string value) Is the variable information to be assigned to the object. Valid size: 1-2999 characters.
API Return messages	0 = Success, 10 = ArgOutOfRange
Comments	The object to be changed must be a string based marking object. The object index must equal to the appropriate and be part of the object list. The marking job must not be in process. The ExecuteJobOnce command must not be called before the variables' assignment.



### Note!

Not the object name, but it's index number is relevant to the assignment of variables. The index number can be determined in the software cabLase.



### Note!

Please take into account that, in the event of creating layouts, inserting objects will change the respective index number of subsequent objects.



### Note!

Object indexing starts with 0. The index number is each reduced by 1 and starts with "0".



### Note!

The marking job must not be in process when modifying variables. The ExecuteJobOnce command must not be called before the variables' modification.

**Note!**

To imbed control characters for the object type data matrix code, use the tilde ( ~ ) character before the control code. To imbed an actual ~ character, use two tilde characters in a row ( ~~ ). To imbed an ASCII 0 character, use ~@ instead of the ASCII 0. Refer to the software manual cabLase Editor 5, "Formatted String", for more detailed information.

## 7.27 TakeHostControl

Command	TakeHostControl
Purpose	Allows exclusive control of the laser control from external
Implementation	2
Parameter	-
API Return messages	0 = Success, 4 = InControl
Comments	A client or the software cabLase cannot gain exclusive control of the laser control, if it is busy processing a marking job. In this case, the other client must terminate the connection, the software must close or wait until the current marking job is processed.

**Note!**

Use the ReleaseHostControl command to terminate an exisiting connection.

**Note!**

Use the GetJobStatus command to determine, if there is a job currently being processed.

**Note!**

Terminate the software cabLase before starting the Remote API interface to enable release of the common port.

## 7.28 TransformObject

Command	TransformObject
Purpose	Applies rotation, scaling and offset to the specified object
Implementation	102, Objectindex, Rotation, Rotationscenterx, Rotationscentery, Xscale, Yscale, Xoffset, Yoffset
Parameter	<p>Objectindex: Equals to the integer object index of the object</p> <p>Rotation: Specifies the rotation angle, in degrees, valid range -360 to +360</p> <p>RotationcenterX: Specifies the coordinate position representing the center of rotation in the x-axis, in bits, valid range -2147483648 to +2147483647.</p> <p>RotationcenterY: Specifies the coordinate position representing the center of rotation in the y-axis, in bits, valid range -2147483648 to +2147483647.</p> <p>Xscale: Specifies the amount to scale the object in the x-axis. Value must be greater than 0.</p> <p>Yscale: Specifies the amount to scale the object in the y-axis. Value must be greater than 0.</p> <p>Xoffset: Specifies the amount to move the object in the x-axis, in bits, valid range -2147483648 to +2147483647.</p> <p>Yoffset: Specifies the amount to move the object in the y-axis, in bits, valid range -2147483648 to +2147483647.</p>
API Return messages	0 = Success, 10 = ArgOutOfRange
Comments	Both the Object outline and the Object fill are transformed with this call. Transformation remains the same, even if for example a SetObjectString command is called or if the object is a serial number field. Subsequent calls of the command will be relative to the last transformation performed. Transformations may effect that an object is outside the specified marking field. To check, whether an object is within the legal marking field or not, the GetObjectRectangle command in combination with the size of the marking field can be called. The marking field is structured as cartesian coordinate system originating in the center.



### Note!

Coordinates for movement and rotation center must be converted into bits.



### Note!

Using this command it is the responsibility of the programmer to insure that after an object has been transformed, it is within the legal marking field.



### Attention!

Repeated calls of TransformObject lead to a relative transformation. To clear all transforms and to reset the layout you have to call the ResetUserTransform command.

## 8.1 Control

Command	Description
Control	
1	Abort
2	TakeHostControl
3	ReleaseHostControl
4	GetHostControlStatus
5	GetHostInControl
6	EnableBroadcasting
7	LoadHardwareDefaults
8	HardwareReset
9	GetRemoteIP
10	GetKFactor
14	SetPerformanceGlobals
15	ResetPerformanceGlobals
16	OpenCOMPort
17	CloseCOMPort
18	COMWriteLine
19	GoToZ
20	GoToXYZ
21	SetMOTFEncoderRate
22	SetMemBuffer
23	GetMemBuffer
24	GetAvailableRAM
27	COMWriteChar
29	SetUserOutBit
30	GetUserInWord
31	GetAllIOWords
32	SetUserOutInitWord
33	GetUserOutInitWord
34	SampleMOTFEncoderCount
35	ClearMOTFEncoderCount
36	GetMOTFEncoderCount
37	Echo
38	GetLensFileList
39	LoadLensFile
40	SetUserOutPreferences
41	GetUserOutPreferences
42	SetUserOutWord
43	GetVersions
44	GetLaserFileList
45	LoadLaserFile
46	GetMotionFileList
47	LoadMotionFile
48	GetFontFileList
49	GetActiveLaser

Command	Description
Control	
50	GetActiveLens
52	GetAvailableDiskSpace
56	ClearCommandCache
57	TurnLaserOn
58	TurnLaserOff
59	GetMotionDeviceNames
60	GetMotionCalFactors
61	SendMotionCommand
63	GetMotionErrorCodes
64	GetMotionHomedOnceFlags
65	GetMotionStatus
66	GetProfileFileList
67	GetLastInterlockWord
70	PulseUserOutBit
71	COMWriteBinarychar
72	COMWriteCarEx
73	COMWriteLineEx
74	COMReadLineEx
75	CloseCOMPortEx
76	SetZOffsetRWU
77	SendMotionCommandEx

**Note!**

Please refer to the separate manual to get the complete documentation of all Remote API commands.

## 8.2 Objects

Command	Description
Objects	
100	SetObjectString
102	TransformObject
103	GetObjectRect
104	GetObjectCenter
105	GetObjectType
106	EnableObject
107	GetObjectString
108	GetObjectName
109	SetObjectUserData
110	GetObjectUserData
111	ResetObject
112	ResetUserTransform
113	TransformObjectByName
114	TransformObjectByNameEx
115	SetObjectProfile
116	GetObjectProfile
117	SetObjectProfileFromFile
118	GetObjectNumPasses
119	SetObjectNumPasses
120	GetObjectMarkMode
121	SetObjectMarkMode
122	GetObjectNumMarkingPasses
123	AddObjectMarkingPass
124	DeleteObjectMarkingPass
125	SetObjectPassSettings
126	GetObjectPassSettings
127	TransformObjectNewFill
128	TrnsformObjectByNameNewFill
136	NewObject
137	SetObjectUnicodeString
138	GetObjectUnicodeString
139	GetObjectVectors
140	SetObjectVectors
141	RemoveObject
142	GetObjectExecuteTime
143	SetObjectName
144	SetObjectProperties
145	GetObjectProperties
146	SetObjectOutlineSettings
147	GetObjectOutlineSettings
148	SetObjectFillSettings
149	GetObjectFillSettings
151	GetObjectFontMetrics



### 8.3 Marking Job

Command	Description
Marking job	
200	ClearJobList
201	MakeJobActive
202	RemoveJob
203	GetFlashJobFileList
204	GetUSBJobFileList
205	LoadFlashJob
206	LoadUSBJob
207	ExecuteJobOnce
208	ExecuteJobContinuous
209	GetJobStatus
210	GetLastError
211	GetObjectCount
214	GetJobExecutionStatus
215	SetExternalStartMode
216	GetExternalStartMode
218	GetActiveJob
219	SaveFlashJob
220	SaveUSBJob
221	GetNetworkJobFileList
222	LoadNetworkJob
223	SaveNetworkJob
224	GetLastMotionError
225	NewJob


**Note!**

Please refer to the separate manual to get the complete documentation of all Remote API commands.

## 8.4 Administration

Command	Description
Administration	
500	SetAdminPIN
501	GetAdminPIN
502	SetDHCPMode
503	GetDHCPMode
504	SetLocalGateway
505	GetLocalGateway
506	SetLocalIP
507	GetLocalIP
508	SetNodeFriendlyName
509	GetNodeFriendlyName
510	SetSubnetMask
511	GetSubnetMask
512	SetUserPIN
513	GetUserPIN
514	SetCOMPortSpeed
515	GetCOMPortSpeed
516	SetCOMPortAssignments
517	GetCOMPortAssignments
518	SetLocalTime
519	GetLocalTime
523	ConnectNetworkShare
524	SetCOMPortSpeedEx
525	GetCOMPortSpeedEx
526	GetLocalDeviceList
527	SetActiveLocalDevice
528	SetCOMPortMode
529	GetCOMPortMode



### Note!

Please refer to the separate manual to get the complete documentation of all Remote API commands.

The following table shows possible API Response Codes returned from the Host to the client. Response Codes are used for fault detection in case of difficulties in flow control.

Value	Short description	Description
0	Success	The operation completed successfully
1	Idle	idle mode, no marking job in process
2	Busy	A marking job is currently in process
3	NoJob	The specified job was not found
4	InControl	The requesting client has exclusive control of the Host
5	NotInContrl	The requesting client does not have exclusive control of the Host
6	LicenseUnavailable	No valid license was found
7	LicenseAccessDenied	Access denied. The current license does not allow the requested feature
8	BadCommand	The API command was not recognized
9	BadArg	A specified argument was invalid
10	ArgOutOfRange	A specified argument was out of range
11	UnkownTimeZone	The specified time zone cannot be found when setting up the system
12	Reserved	Reserved
13	BadConversion	Transmission error. Error while converting between multi-byte and Unicode characters
14	RegistryError	A Windows CE Registry read or write operation failed
15	TimeZoneFileError	A Time Zone File operation failed
16	ResetInterlock	An interlock was signaled and must be reset by calling Abort
17	ListNotOpen	An operation was attempted on a list that has not been opened
18	ListAlreadyOpen	The list is currently open
19	BadData	The data in the specified file was not in the correct format
20	APIException	The Remote API caused an unexpected exception
21	JobAborting	The job is currently aborting from a previous abort command
22	FPGALoadFail	An attempt to load the FPGA with instructions failed
23	JobManagerInitFail	The Job Manager failed to initialize properly
24	LaserLoadFail	The specified laser configuration failed to load properly
25	LensLoadFail	The specified lens configuration failed to load properly
26	PMLoadFail	The specified Performance Matrix configuration failed to load properly
27	MotionLoadFail	The specified motion configuration failed to load properly
28	HostManagerInitFail	The Host Manager failed to initialize properly
29	InvalidIPAddress	The specified IP address is not a valid IPv4 IP address
30	DataUnknown	The format of the data is not recognized
31	BadChecksum	The data failed a checksum test
32	NetworkShareNotConnected	There was an attempt to use a network resource, but no connection exists

Value	Short description	Description
33	NetworkConnectFail	An attempt to connect to a network share failed.
34	UnknownNetworkError	An unspecified network error has occurred
35	APICommandTimeout	A command that was sent to the Remote API timed out
36	ExternalProcessFail	Internal use
37	DLLLoadFail	Internal use
38	NoAdapter	No Network adapter was found
39	AddIPAddressFailure	An attempt to add a temporary IP address failed
40	BadAPIResponse	The Remote API returned an unexpected response
41	CannotCreateSocket	Internal use
42	CannotConnectSocket	Internal use
43	CannotGetFPGABufInfo	Internal use
44	CannotGetFPGABuf	Internal use
45	CannotWriteFPGABuf	Internal use
46	FPGAException	Internal use
47	FTPConnectionError	An attempt to connect to an FTP resource failed
48	FileAlreadyExists	The specified file already exists in the specified location
49	UnknownOS	Internal use
50	SocketException	Internal use
51	ProcessTimeout	The process returned a time out error
52	DeviceNotFound	The specified device cannot be found
53	LoginInProgress	An attempt to connect to a device is currently executing
54	APIClientInControl	There is an active connection to a Remote API client
55	StreamClientInControl	There is an active connection to a cabLase client
56	CannotConnectToAPI	An attempt to connect to a device using the Remote API port failed
57	ReadFail	Internal use
58	StreamBufferFull	Internal use
59	NoConfigRecord	Cannot find the specified configuration record
60	OperationCanceled	Operation was canceled by the user
61	NoData	Internal use
62	InitializationError	Internal use
63	FailToCreateServiceThread	Internal use
64	CannotOpenDevice	Internal use
65	SegmentFull	Internal use
66	MarkerLibraryNotInitialized	An operation was attempted before the Marker Library was initialized
67	RingBufferNotInitialized	An operation was attempted before the Ring Buffer was initialized
68	AccessDenied	Access to a resource was denied
69	RequiresUACElevation	The attempted operation can only be performed by elevation the UAC

Value	Short description	Description
70	NotAllowed	The requested operation is not allowed
71	NoLaserConfig	The laser config file was not found
72	NoLensConfig	The lens config file was not found
73	OutOfMemory	There is not enough memory to complete the task
74	LensTableNotFound	The specified lens correction table cannot be found
75	HostControlInitError	No Host Controllers loaded during device boot time
76	NoBytesRead	A read operation failed with no bytes read
77	WritePending	Data was added to the Pending queue and will execute when possible
84	NoPen	A pen found in the vectorlist did not have an associated Profile
100	NoFilesFound	No files were found at the specified path
101	NoDrive	No drive was found
102	JobOutOfMemory	Out of memory exception
103	TooManyObjects	Internal error, consult manufacturer
104	NoObject	The specified object does not exist
105	JobException	An internal job exception
106	NotInHostControl	Operation cannot be performed if the client is not in control
107	WrongHostType	Operation cannot be performed with this host type
108	ErrorJobBusy	Operation cannot be performed while a job is executing
109	NoActiveJob	There is no Active Job
110	ErrorSoftware	Internal error, consult manufacturer
111	LoadFail	A job load failed
112	NoObjects	Job file version not compatible with current firmware
113	WriteFail	Internal error writing job file
114	JobFileFormat	Job file format error
115	FileException	Internal error while processing file
116	UnknownObject	Unknown object type
117	UnknownType	Unknown type
118	NotSupported	Operation not supported
119	NotAvailable	Resource not available
120	FPGADDataFail	Internal FPGA data format failure
121	FileNotFound	The file specified was not found
122	FileCreationError	Error while attempting to create a file
123	WriteFileFail	Not all data was written to the file
124	PathNotFound	The specified path was not found
125	NotInCacheMode	The command requires that the job was started in Cache mode
126	NotWaitingForStartMark	The FIFO is not currently waiting for a Start Mark signal
127	MotionNotHomed	The command is not allowed if the device is not homed
128	No3DModel	The operation cannot complete because there is no 3D model loaded

Value	Short description	Description
129	ProjectionError	An error was encountered while projecting onto the 3D model
200	NoProperties	Object does not contain any properties
201	ObjectException	Internal object exception
202	Abort	Operation was aborted
203	NoFontResource	The font specified in the object was not found
204	NoOverride	Internal object error
205	ExternalEnableDenied	Operation denied by External control
206	CannotCreatePort	A port setting (baud rate, stop bits, etc.) is invalid
207	CannotOpenPort	Error while attempting to open COM port
208	PortNotOpen	Port must be open to execute command
209	PortTimeout	A port operation timed out
210	WrongPortNumber	Invalid port number
211	WrongObjectType	Operation is not supported by this object type
212	AxisNotConfigured	A motion control axis referenced in an object has not been configured
213	TextBufferOverflow	Too many characters in buffer (max. 3000 including command opcode)
214	InvBarcodeStringValue	The barcode string to encode contains invalid characters
215	InvBarcodeStringLength	The length of the barcode string to encode is either too short or too long
216	InvBarcodeNarrowWidth	The narrow to wide ratio is invalid
217	InvBarcodeWidthReduce	The width reduction value is invalid
218	InvBarcodeECC	The ECC value is invalid, or cannot be used to encode the string
219	BarcodeOutOfMemory	There was insufficient memory to complete an internal barcode operation
220	BarcodeUnknownError	Undocumented error. Please notify the manufacturer
221	BarcodeException	Incorrect data, or an internal operation resulted in an unexpected result
222	NoVectors	An object was saved to the job with both mark outline and mark fill disabled
223	BadMotionResponse	The motion controller responded with an unexpected value
224	MotionDriverNotFound	An object is referencing a motion driver that does not exist
225	AxisNotFound	An operation was attempted on an axis index that was not found
226	EncoderNotFound	An operation was attempted that depends on an encoder, and no encoder was found
227	InvStringValue	The string to process is not a valid string
228	MotionControllerNotFound	The motion system cannot detect a valid motion controller device
229	MotorNotProvisioned	The motor has not been provisioned for use with the LEC controller
230	RuntimeMotionError	An error was generated during a motor move operation

Value	Short description	Description
231	ObjectOutOfBounds	The object is outside the legal marking area
232	InvVersion	Invalid version
233	NoOutline	Not valid to have MarkOutline enabled with no outline to mark

**Note!**

It may happen that an error message in textform is received instead of the numerical value expected.

The following are System Error Codes that may be returned by the Remote API command GetLastError.

Value	Short description	Description
8001	QueueFul	Overflow spooler queue
9001	ProcessAbort	Process cancelled
9002	FIFOEmptyTimeout	Timeout FIFO memory – empty
9003	EventTimeout	Event Timeout
9004	BadOpcode	Invalid command
9005	FirmwareBug	Error firmware
9006	WriteDigitalBad	Error when writing digital information
9007	SetLaserPowerBad	Error laser power
9008	SetCorrectionTableBad	Error lense correction file
9009	SetLaserPulseBad	Error laser pulse width
9010	WaitForIOBad	Error digital IO interface
9011	WaitForIOTimeout	Timeout error digital IO interface
9012	SetLaserStandbyBad	Error standby laser
9013	CPLDTimeout	Timeout error CPLD
9014	LaserActiveTimeout	Error when switching on laser
9015	SetMotfOrientationBad	Error when orientating object
9016	EnableMotfBad	Error when using axis module
9020	ServoFault	Error servo driveehler Servoantrieb
9021	InterlockAssert	E-Stop detected
9022	interlockDeassert	E-Stop Reset



The laser marking software cabLase 5 uses different object types to create the layouts. In the event of addressing via Remote API interface the various object types are not transmitted in form of a name, but as integer value.

The following are the values for different object types.

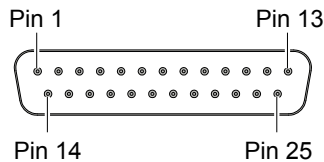
Value	Short description
0	Polyline
1	Barcode
2	Text
3	Bitmap
4	Vector Graphic
5	Point
6	Line
7	Polygon
8	Rectangle
9	Rounded Rectangle
10	Spiral
107	Laser Control
108	Set Port
109	Time Delay
110	Wati Port
111	Alignment
115	Rotary Motion
116	Linear Motion
117	XY Motion

**Note!**

Please note that for addressing objects via Remote API interface the continuous index number is relevant and not the type or name of the object.

### 12.1 External Interface I/O CON2

For the integration into higher-level control procedures the device is equipped with an I/O Interface allowing to individually program eight in- and outputs for example via cabLase Editor 5. The interface has a SUB-D connector, 25 pole.



External interface I/O

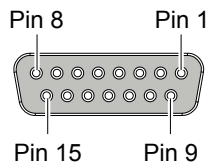
Pin	Signal	Name	Description	Activation / active
1		FP24V	Internal operating voltage +24 V, Si T 500 mA to supply external devices	
2	⊕→	Ready to Mark	Marking job is loaded, device is waiting to start marking	Contact Pin 2 / Pin 12 closed
3	⊕→	Busy	Marking job is running	Contact Pin 3 / Pin 12 closed
4	⊕→	User Out1	Individually programmable	Contact Pin 4 / Pin 12 closed
5	⊕→	User Out2	Individually programmable	Contact Pin 5 / Pin 12 closed
6	⊕→	User Out3	Individually programmable	Contact Pin 6 / Pin 12 closed
7	⊕→	User Out4	Individually programmable	Contact Pin 7 / Pin 12 closed
8	⊕→	User Out5	Individually programmable	Contact Pin 8 / Pin 12 closed
9	⊕→	User Out6	Individually programmable	Contact Pin 9 / Pin 12 closed
10	⊕→	User Out7	Individually programmable	Contact Pin 10 / Pin 12 closed
11	⊕→	User Out8	Individually programmable	Contact Pin 11 / Pin 12 closed
12	⊕→	Common for Outputs	Common potential for all outputs	
13		GND	0 V internal	
14	⊖←	Start Process	Start of marking according to settings made in the software	Connect to +24V between Pin 14 and Pin 25
15	⊖←	Job Select	<b>Active</b> : Job data are loaded from an internal file and can be opened via binary coding by using the digital inputs ▷ Storing job files for the stand-alone operation <b>Inactive</b> : Job data are loaded from the PC	Connect to +24V between Pin 15 and Pin 25
16	⊕→	Error	Error message of the control	Contact Pin 16 / Pin 12 closed
17	⊖←	User In1	Individually programmable	+24V between Pin 17 and Pin 25
18	⊖←	User In2	Individually programmable	+24V between Pin 18 and Pin 25
19	⊖←	User In3	Individually programmable	+24V between Pin 19 and Pin 25
20	⊖←	User In4	Individually programmable	+24V between Pin 20 and Pin 25
21	⊖←	User In5	Individually programmable	+24V between Pin 21 and Pin 25
22	⊖←	User In6	Individually programmable	+24V between Pin 22 and Pin 25
23	⊖←	User In7	Individually programmable	+24V between Pin 23 and Pin 25
24	⊖←	User In8	Individually programmable	+24V between Pin 24 and Pin 25
25	⊖←	Common for Inputs	Common potential for all inputs, 0V external	Note : Using the internal +24V (Pin 1) Supplying the inputs requires to connect Pin 25 with Pin 13.

Tabelle 2 Pinout external interface I/O

## 12.2 Remote Interface CON3

The Remote Interface CON3 is used to control the hardware of the FL+.

The interface has a SUB-D connector, 15 pole.



Remote Interface

Pin	Signal	Name	Description	Activation / active
1		FP24V	Internal operating voltage +24 V, max. 500 mA to supply external devices	
2	⊖→	Power	Mains voltage applied	Contact between Pin 2 and Pin 7 closed
3	⊖→	Shutter Open	Shutter is open	Contact between Pin 3 and Pin 7 closed
4	⊖→	Emission	Laser source is switched on and ready for marking ► Keep to the special safety instructions for the operation under laser class 4	Contact between Pin 4 and Pin 7 closed
5	⊖→	Error	Control is not yet ready after switching on or there is an error occurred in the control	Contact between Pin 5 and Pin 7 closed
6	⊖→	Laser Ready	Control and laser source are switched on and ready for marking	Contact between Pin 6 and Pin 7 closed
7	⊖→	Common for Outputs	Common potential for all outputs	
8		GND	0V internal	
9	⊖←	Job Select Register	<b>active</b> : User In/Out 1 to User In/Out 4 at CON2 are used as digital inputs/outputs 1 to 4 in cabLase <b>inactive</b> : User In/Out 1 to User In/Out 8 at CON2 are used as digital inputs/outputs 5 to 12 in cabLase <b>Note:</b> Supplying Pin 9 is only evaluated for the initialization of the marking laser (Switch on and Reset)! When the signal is inactive User in 1 to 8 can be used to select layouts but objects inside a layout cannot be selected.	Active, if supplied with +24V between Pin 9 and Pin 15
10	⊖←	Reset	Reset the marking laser <b>Note</b> : The restart after reset and initialization of the laser control may take up to 30 seconds!	Active, if supplied with +24V between Pin 10 and Pin 15
11	⊖←	Laser On/ Off	<b>Active</b> : Switch on laser source Conditions for switching on: Emergency Stop not active, shutter closed <b>Inactive</b> : Switch off laser source	Active, if supplied with +24V between Pin 11 and Pin 15
12	⊖→	Pilot Laser is On	Pilot laser is switched on	Contact between Pin 12 and Pin 7 closed
13	⊖←	Open Shutter	Requirements to open shutter Conditions for opening: Emergency Stop not active, interlock closed	Active, if supplied with +24V between Pin 13 and Pin 15



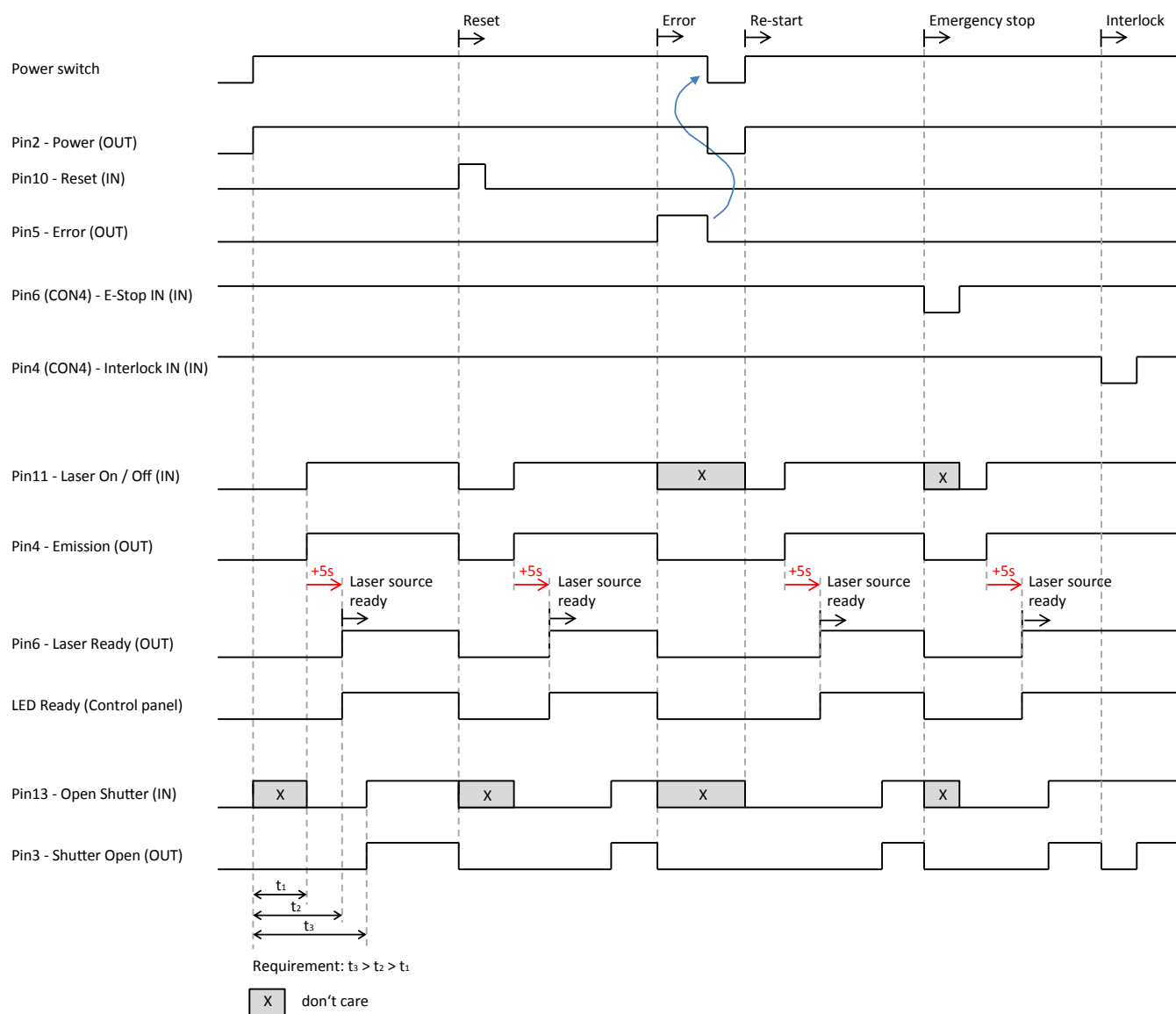
PIN	Signal	Name	Description	Activation / active
14		Pilot Laser On/Off	<b>Active</b> : Switch on pilot laser Conditions for opening: Emergency Stop not active, shutter closed <b>Inactive</b> : Switch off pilot laser	Active, if supplied with +24V between Pin 14 and Pin 15
15		Common for Inputs	Common potential for all inputs <b>Note</b> : Using the internal +24V (Pin 1) to supply the inputs requires the connection of Pin 15 with Pin 8 (GND).	

Tabelle 3 Pinout Remote Interface

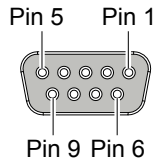
**Signal diagram remote interface**

Signal diagram remote interface

### 12.3 Interlock / E-Stop Interface CON4

The Interlock / E-Stop interface is used to integrate the marking laser into external safety circuits and to connect an external Emergency Stop.

The interface has a SUB-D connector, 9 pole.



Interlock / E-Stop interface



#### Note!

There is no marking possible without having connected the inputs Interlock In (Pin 4) and E-Stop In (Pin 6).

Pin	Signal	Name	Description	Activation / active
1		FP24V	Internal operating voltage +24 V, max. 500 mA	
2	⊕→	E-Stop signaling contact connection A	Status of Emergency Stop relay' Device ready for operation when signal active	Contact between Pin 2 and Pin 7 is open, if the Emergency Stop is not activated, i.e. the Emergency Stop relay is not current fed
3	⊕→	Interlock Signaling contact connection A	Status of Interlock-Relay' Device ready for operation when signal active	Contact between Pin 3 and Pin 8 is open, if the safety circuit is closed, i.e. the Interlock relay is current fed
4	⊖←	Interlock IN	Interlock relay Connection for safety switch Device ready for operation when signal active	Active, if connected to +24V between Pin 4 and Pin 9
5		GND INT	0 V intern	
6	⊖←	E-Stop IN	Emergency Stop relay Connection for Emergency Stop Device ready for operation when signal active	Active, if connected to +24V between Pin 6 and Pin 9
7	⊕→	E-Stop signaling contact connection B	Status of Emergency Stop relay' (return circuit)	▷ Pin 2
8	⊕→	Interlock signaling contact connection B	Status of Interlock relay' (return circuit)	▷ Pin 3
9		GND EXT	Common GND potential for Interlock IN and E-Stop IN	

Tabelle 4 Pinout Interlock / E-Stop interface

### 13.1 Reference Documents

	Name of document	Origin
1	20.015 LEC Remote API Manual.pdf	LC
2	cab_ma_flplus_400_de.pdf	cab

### 13.2 Revisio History of Sample Programs

Revision	Changes	Date
FL+ Remote API Library_160315.zip	Delivery condition	15.03.15

### 13.3 Contact

Name	Company	Function / Department	email
Mehmet Seker	cab GmbH & Co. KG	Service / Support	m.seker@cab.de
Martin Föll	cab GmbH & Co. KG	Service / Support	m.foell@cab.de
Hans Löhner	cab GmbH & Co. KG	Software / Support	h.loehner@cab.de