Open Advanced Process Control

User manual for ControlRoom, BeamConstruct and additional tools

Version 5.5

Table Of Contents

1 Copyright

2 Legal Issues

LICENSE AGREEMENT

The Open Advanced Process Control software package that is made available (the "Software") is owned by OpenAPC Project Group (the "Vendor"). This Software is protected by copyright laws, and is being made available solely for use by you in accordance with the following terms and conditions. Any use, reproduction or redistribution of the Software that is not in accordance with this Software License Agreement is expressly prohibited by law, and may result in civil and criminal penalties.

SOFTWARE LICENSE AGREEMENT

THE VENDOR IS WILLING TO LICENSE THIS SOFTWARE TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS LICENSE AGREEMENT. This is a legal agreement between you (either an individual end-user or an entity) and the Vendor ("Agreement"). By using this software, you are agreeing to be bound by the terms and conditions of this Agreement. If you do not agree to the terms and conditions of this Agreement, promptly return the software and other items that are part of this product in their original package with your sales receipt to your point of purchase for a full refund, or if you have downloaded this software from the Vendors web site, then you must stop using the software and destroy any copies of the software in your possession or control.

1. Grant of Agreement.

Subject to the terms and conditions of this Agreement, the Vendor and its suppliers grant to you a non-exclusive license to use one licensed copy of the Software and any documentation accompanying this Agreement on one computer. No other rights are granted. The Software is in use if it is loaded on the computer's permanent or temporary memory. For backup purposes only, you may make one copy of the Software. You must include on the backup copy all copyright and other notices included on the Software as supplied by the Vendor. Installation on a network server for the sole purpose of your internal distribution of the Software is permitted only if you have purchased an individual Software license for each networked computer to which the Software is distributed.

2. Restrictions.

The Software may contain copyrighted material, trade secrets, and other proprietary materials of the Vendor and its licensor's. You agree that in order to protect those proprietary materials, except as expressly permitted by applicable law, neither you nor a third party acting on your behalf will:
(I) decompile, disassemble or reverse engineer the Software;
(II) modify or create derivative works of the Software;
(III) transmit the Software, in whole or in part, over the Internet or other network; or
(IV) sell, distribute, rent, lease, sub license or otherwise transfer the Software to a third party; that does not influence your right to sell the Software together with additional equipment that is controlled by this Software

3. Ownership.

The Software is licensed, not sold, to you for use only under the terms and conditions of this Agreement, and the Vendor reserves all rights not expressly granted to you in this Agreement. The Vendor and/or its licensor's retain title to the Software, and all intellectual property rights therein.

4. Termination.

This Agreement is effective until terminated. Upon any violation of any of the provisions of this Agreement, rights to use the Software shall automatically terminate and the Software must be returned to the Vendor or all copies of the Software must be destroyed. You may also terminate this Agreement at any time by destroying all copies of the Software in your possession or control. If the Vendor makes a request via public announcement or press release to stop using the copies of the Software, you will comply immediately with this request. The provisions of paragraphs 3, 7, 8 and 12 will survive any termination of this Agreement.

5. Limited Product Warranty.

The Vendor warrants to you that the Software will substantially conform to its published documentation and the media containing the Software shall be free from defects in material, each for a period of thirty (30) days from the date of purchase. The Vendors limited warranty is non-transferable and is limited to the original purchaser. This warranty gives you specific legal rights, and you may also have other rights which vary under local laws.

6. Remedies.

The Vendors entire liability and your exclusive remedy for any breach of warranty shall be, at the Vendors option, to:
(a) repair or replace the Software or media, provided that the Software or media is returned to the point of purchase or such other place as the Vendor may direct, with a copy of the sales receipt, or
(b) refund the price paid.
Any replacement Software or media will be warranted for the remainder of the original warranty period or thirty (30) days,whichever is longer. These remedies are void if failure of the Software or media has resulted from accident, abuse, or misapplication.

7. DISCLAIMER OF WARRANTY.

THE WARRANTIES EXPRESSLY SET FORTH IN THIS AGREEMENT REPLACE ALL OTHER WARRANTIES. THE VENDOR AND ITS SUPPLIERS EXPRESSLY DISCLAIM ALL OTHER WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS WITH RESPECT TO THE SOFTWARE OR MEDIA, AND ANY WARRANTIES OF NON-INTERFERENCE OR ACCURACY OF INFORMATIONAL CONTENT. NO DEALER, AGENT, OR EMPLOYEE OF THE VENDOR IS AUTHORIZED TO MAKE ANY MODIFICATION, EXTENSION, OR ADDITION TO THIS WARRANTY.

Some jurisdictions do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you.

8. LIMITATION OF LIABILITY.

IN NO EVENT WILL THE VENDOR OR ITS SUPPLIERS BE LIABLE FOR ANY COSTS OF PROCUREMENT OF SUBSTITUTE PRODUCTS OR SERVICES, LOST PROFITS, LOSS OF INFORMATION OR DATA, OR ANY OTHER SPECIAL, INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES ARISING IN ANY WAY OUT OF THE SALE OF, USE OF, OR INABILITY TO USE ANY VENDORS PRODUCT OR SERVICE, EVEN IF THE VENDOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO CASE SHALL THE VENDOR ITS SUPPLIERS' TOTAL LIABILITY EXCEED THE ACTUAL MONEY PAID FOR THE VENDORS PRODUCT OR SERVICE GIVING RISE TO THE LIABILITY.

Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you. The above limitations will not apply in case of personal injury where and to the extent that applicable law requires such liability.

9. Export Law Assurances.

You agree and certify that neither the Software nor any other technical data received from the Vendor will be exported outside the Commonwealth of Australia except as authorized and as permitted by the laws and regulations of the Commonwealth of Australia. If you have rightfully obtained the software outside of the Commonwealth of Australia, you agree that you will not re-export the Software nor any other technical data received from the Vendor, except as permitted by the laws and regulations of the Commonwealth of Australia and the laws and regulations of the jurisdiction in which you obtained the Software.

10. Agents and Third Party Purchasers.

If you are acquiring the Software on behalf of another person or entity, you represent and warrant that you have the authority to bind the party or entity for which you are acquiring the Software to the terms and conditions of this Agreement.

11. General Terms and Conditions.

This Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Australia, without regard to or application of its choice of law rules or principles. If for any reason a court of competent jurisdiction finds any provision of this Agreement, or portion thereof, to be unenforceable, that provision of the Agreement shall be enforced to the maximum extent permissible so as to affect the intent of the parties, and the remainder of this Agreement shall continue in full force and effect. This Agreement constitutes the entire agreement between the parties with respect to the use of the Software and supersedes all prior or contemporaneous understandings, communications or agreements, written or oral, regarding such subject matter. The Vendor may, in its sole discretion, modify portions of this Agreement at any time. The Vendor may notify you of any changes by posting notice of such modifications on the Vendors web site(s) or sending notice via e-mail, postal mail or other means. Your continued use of the Software following notice of such modifications shall be deemed to be your acceptance of any such modifications to the Agreement. If you do not agree to any such modifications, you must immediately stop using the Software and destroy all copies of the Software in your possession or control.

12. The Software is protected by Commonwealth of Australia copyright law and international treaty. Unauthorized reproduction or distribution of the Software is subject to civil and criminal penalties.

NOTICE SPECIFIC TO INFORMATION AVAILABLE ON THE WEBSITE OF THE VENDOR:

The information provided is for informational purposes only and is subject to change without notice. It is provided "AS IS" and without any warranty. Any risk arising out of the use of the information on this website shall remain with the reader. IN NO EVENT SHALL THE VENDOR BE LIABLE FOR ANY INDIRECT, CONSEQUENTIAL, INCIDENTAL, SPECIAL, PUNITIVE OR OTHER DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION OR LOSS OF BUSINESS INFORMATION), ARISING OUT OF OR IN CONNECTION WITH THE AVAILABILITY, PERFORMANCE OR USE OF THE INFORMATION CONTAINED ON THIS WEBSITE, EVEN IF THE VENDOR HAS BEEN ADVISED OF THE POSSIBILITIES OF SUCH DAMAGES.

LINKS TO OTHER WEB SITES

Links to other web sites are available on the Vendors web site and these links allow you to leave the Vendors web site. The linked sites are not under the control of the Vendor, the Vendor is not responsible for the contents of any linked site or links within a linked site. These links are provided as a convenience only and do not imply any endorsement or recommendation by the Vendor.

3 Safety

The OpenAPC software package components are designed to control different kind of machinery. This machinery may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. The client shall solely be responsible to strictly comply with all applicable and relevant safety regulations regarding installation and operation of the system at any time.

4 Overview

This document describes the full Open Advanced Process Control software package including all existing functionalities, options and additional software components. Depending on the software variant you really use some of the described features and functionalities may be not available or may be available only in a limited state.

The Open Advanced Process Control software package is available for different hardware and software platforms. Dependent on the peculiarities of the used operating systems the exact storage position of the executable, the plug-ins and other required data may differ, here always a storage location is chosen that is a common position for such a software on that operating system. Within this document no operating system specific information are given, here please refer to the user manual of the operating system.

4.1 The Software Structure

The OpenAPC package is a collection of software components and additional tools that can be used for different purposes. The functionality of all of them can be combined in different ways depending on the type of application. The OpenAPC package contains the following main components that itself may consist of several sub-applications:

- **ControlRoom** is a visualisation and process control software that can be used to create and to run HMIs, to control machinery and production processes as well as to solve different professional and home automation tasks; the ControlRoom software itself consists of separate software applications that can be used for developing (OpenEditor and OpenDebugger) and for running a HMI (OpenPlayer/OpenHPlayer, OpenPlugger, InterlockServer, ...)
- **BeamConstruct** is a CAD-type application specialised for laser-based processing and for all kinds of laser marking operations (welding, cutting, engraving, surface cleaning and processing, ...); such laser marking projects can be executed out of BeamConstruct directly or they can be added to a ControlRoom process control project to integrated them into an existing HMI solution

5 ControlRoom

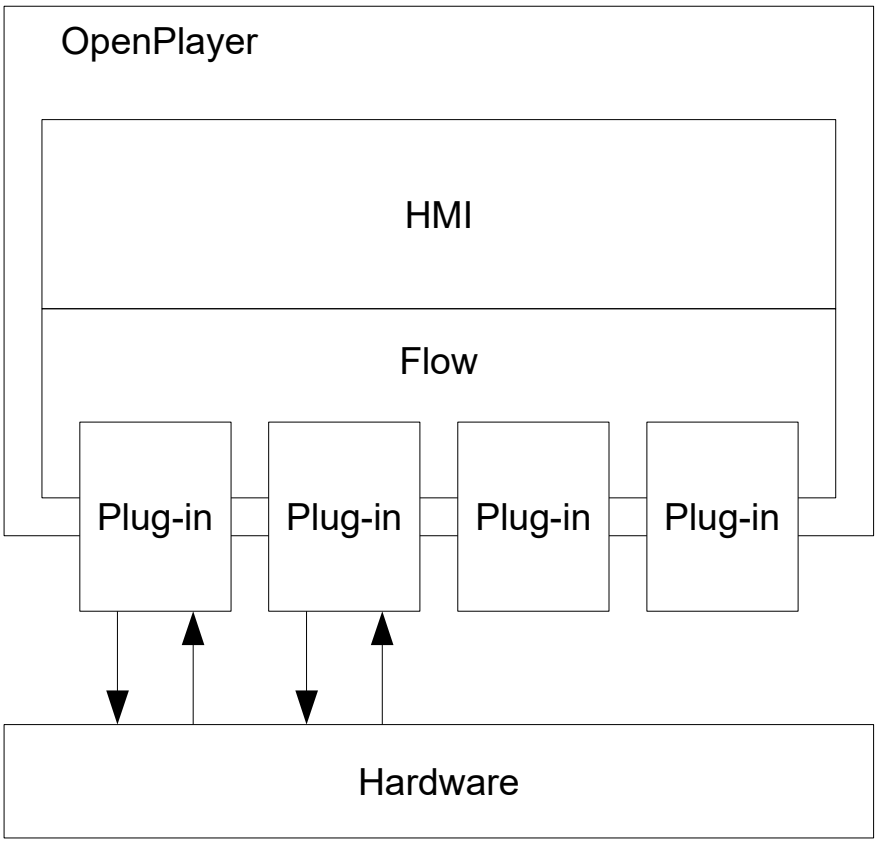
The ControlRoom component of the OpenAPC software package can be used to create HMI (human machine interfaces) and to define process control applications. It consists of several parts:

- the open editor (named „OpenEditor“, not available for all target systems) that can be used to set up a new project and to edit existing ones
- the open debugger (named „OpenDebugger“, not available for all target systems) that can be used to execute a project created with the OpenEditor, this debugger offers possibilities to analyse the flow of data and to test proper function of an OpenAPC project
- the open player (named „OpenPlayer“) that is used on the target system in order to execute a project created with the OpenEditor
- the special player variant “OpenHPlayer” that is able to execute projects without any HMI part, here the defined flows are executed only, no visual output is given
- the optional Interlock Server (named “OpenIServer”) that can be executed in background to reflect and store state information of a running project and that can be used to influence data flow and processes
- the optional runtime environment for plug-ins (named “OpenPlugger”) that can be used for plug-ins with hardware access that have to run outside of the scope of the OpenEditor and communicate with other components via the Interlock Server

Beside of these applications there are several plug-ins available. Such a plug-in is an external program (technically it is a dynamically linked / shared library that is loaded actively during runtime) that provides additional functionalities to the main applications, especially access to different hardware devices. On the target system all unused plug-ins can be removed from the installation in order to keep the size of the whole installation as small as possible. Please note: if there are plug-ins removed that are required by a project, it will not work properly. So please be careful when deleting files out of the runtime package manually.

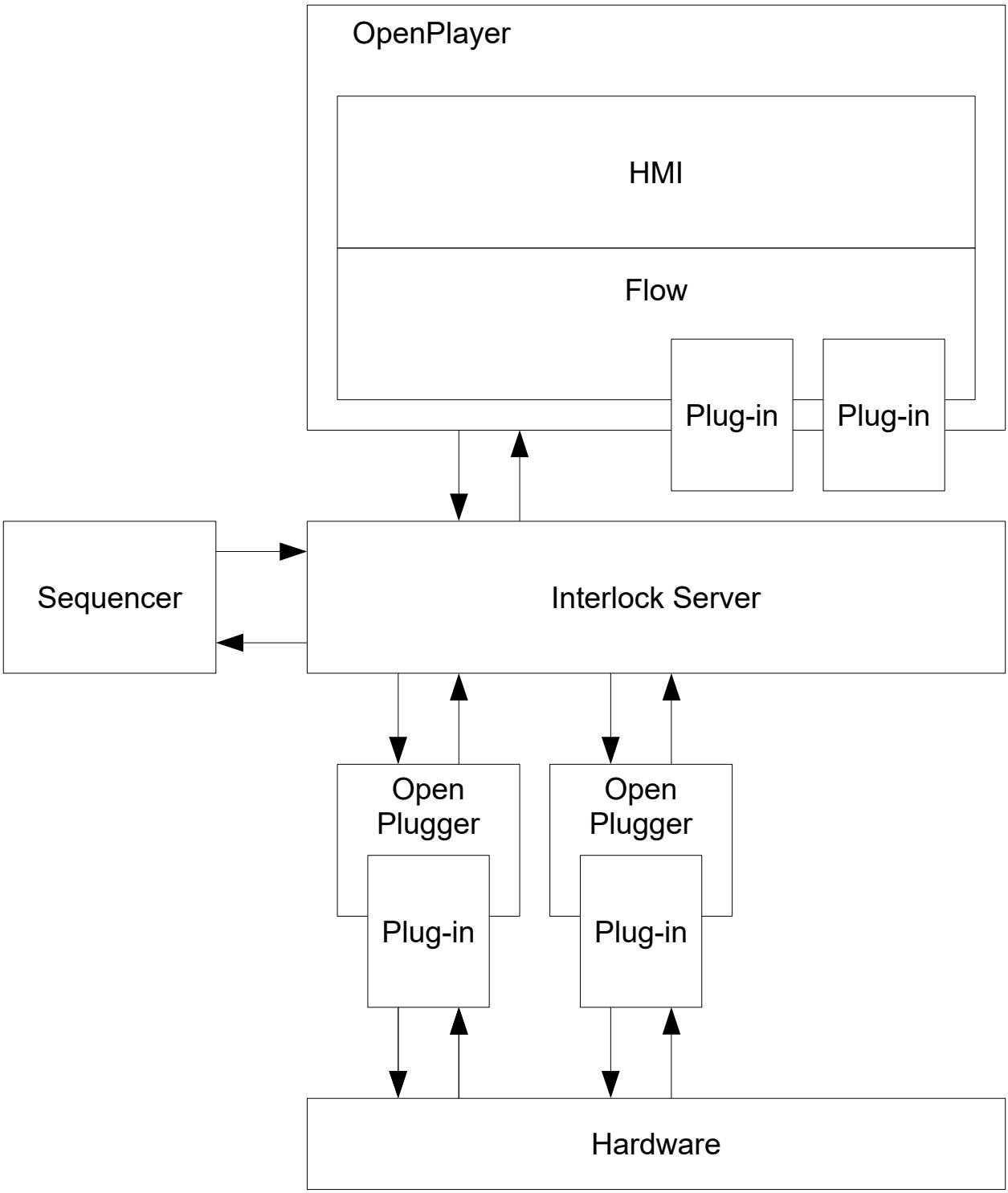
5.1 System Architecture

The ControlRoom software component offers different possibilities how a process control environment can be set up and used. It highly depends on the problem formulation and requirements of the system to be controlled how the environment should be set up.



The most simple architecture consists of the OpenPlayer only: here the Visualisation and the logic flow is handled by the OpenPlayer completely, plug-ins are connected to it directly so that access to hardware or other external data sources is done by the OpenPlayer too. While such a system can be set up fast and easy it has some great disadvantages: it can become confusing very fast when it grows because there is no clear structure where every part of the system is located on an own logic layer. So it is recommended to use this set-up only for very small and simple projects where no chance exists that they may grow over time.

For more complex environments it is recommended to uncompress the whole thing and to put every component into an own layer that corresponds to its task. So all elements (plug-ins) that somehow access the hardware should be uncoupled from the visualisation (HMI of the OpenPlayer). This requires an additional communication instance that gives the possibility to connect the now separated components with each other. Here the Interlock Server comes into account, it is located below the OpenPlayer and above the plug-ins that access the hardware. These plug-ins itself now are managed by a small program named OpenPlugger. This application is able to handle exactly one plug-in and to communicate with the Interlock Server. So for every plug-in that accesses hardware or external devices one instance of the OpenPlugger is used.



This structure now gives the additional possibility to dock other components to the Interlock Server that are able to access and influence all components of this environment. They can watch hardware states (as reported by OpenPlugger instances and their plug-ins), can react on user input (as reported by the OpenPlayer) and can control application flows and sequences. This element – named as Sequencer here – can be an own application, a LUA or Instruction List script or an other instance of the OpenPlayer that resides on an other system. After the communication with the Interlock Server is done via TCP/IP completely all the components referred here can be spread over a network.

The big advantages of such a structure are:

- it can be extended easily simply by adding new elements that all communicate via the Interlock Server
- the structure is clear and the communication paths simple (everything is routed via the Interlock Server, no confusing cross-connections between involved elements are necessary)
- the flow within the OpenPlayer only controls the HMI, no application logic needs to be mixed with the GUI logic
- the structure is much closer to the generally accredited Open Systems Interconnection Reference Model (http://en.wikipedia.org/wiki/OSI_model)
- due to the fact that the communication between elements is done via TCP/IP all logic parts (“Sequencer”, “OpenPlayer”, “Interlock Server”, “OpenPlugger”s) may be executed on own hosts so that a fairly complex system can be set up with components spread over a large network

Of course there is also a disadvantage: a system that is split in many separate parts is more difficult to set up and to maintain. Nevertheless the advantages overbalance them: with such a structure you are on the safe side when your project grows over the time, it will stay maintainable independent from growing requirements.

The structures shown in the pictures here are only examples, depending on the specific requirements the set-up of a system can differ from them. Following all components and features are described in detail, but depending on the chosen architecture it may be possible that some of the functionalities documented here don't have to be used.

5.2 Creating And Running A Project

The procedure of setting up and using a project successfully consists of some simple steps. The starting point for everything is the OpenEditor. Here a new project can be created or an existing one can be loaded to be extended in its functionality. Within the OpenEditor the user interface is set up within the HMI (Human-Machine-Interface) Editor and the logical relations between the user interface elements and plug-ins are set up within the Flow Editor. In case the extended software structure has to be used where hardware access is done via external OpenPlugger's, the third component, the Plugged Devices list can be used to add and configure the plug-ins that will run externally later.

Next this project can be executed within the OpenDebugger. Here the same environment is given to the project like it is used for the target system. Additionally there are some possibilities within the OpenDebugger to analyse the flow of the data, to perform single-stepping and to check the functionality of the project.

During the development process of a project it has to be switched between OpenEditor and OpenDebugger several times until it works as desired.

After this project works as expected it can be executed with the OpenPlayer on the target system. In some seldom cases it also might be necessary to run the application within the OpenDebugger on the target system to find out the reason for some timing problems – more informations about such runtime-dependent problems are given below within the description of the OpenDebugger.

5.3 Using the OpenEditor

The OpenEditor consists of three main parts that can be accessed via tabbed panes:

- the HMI Editor where the user interface and the layout that is visible to the end user can be set up
- the Flow Editor where the logic connections between user interface elements and plug-ins can be configured
- the Plugged Devices list where all these plug-ins can be defined and configured that will not run inside the player directly but are handled by the OpenPlugger; this list can be used only when parameters of a project are set to use the Interlock Server (please refer to “General Project Settings” below)

It can be switched between both editors and the list via the tabs on the upper part of the window. Above these tabs there are some tool bar buttons. Some of them correspond to menus of the application. They can be used for different things like

- creating a new project
- loading an existing project
- saving the current project using the current or a new file name
- editing and deleting the selected HMI element
- inspecting the application for errors and possible problems
- debugging and testing the current project within the OpenDebugger
- executing the current project within the OpenPlayer

These tool bar buttons act on all, the HMI and the Flow Editor as well as the Plugged Devices list. The same is true for the menus, here following additional functionalities can be found:

- changing the project configuration
- searching for an element by its name

5.3.1 The Project Configuration

5.3.1.1 General Project Settings

The menu “File → Project settings” gives access to a configuration dialogue where the general parameters of a project can be specified. These settings are stored within the project file and implicitly include information like the OpenEditor's window size and position and the active size of the Flow Editors editing area. Beside of that the following parameters can be influenced by the user directly:

- Visual Size – this is the width and height (unit pixels) of the HMI area, it specifies the total size of the editing area within the HMI Editor and the size that is used for the applications window when the project is executed in player or debugger. So here the total screen size of the target system should be set in order to hide all other user interface elements of that targets operating system.
- Visual Grid Size – the visual grid is a helper grid that is used and visible only within the HMI Editor. There a grid of points is displayed with a distance in x and y direction as specified here. When a project is executed within the debugger or player this grid is not shown.
- Snap to Grid – When this check box is selected all objects that are placed or re-positioned within the HMI Editor automatically snap with their upper left corner to a position that is specified by the visual grid. This option gives the possibility to position elements very exact, it applies only to the HMI editor and does not have any influence on a running project.
- Background Colour – specifies the background colour of the HMI area, the colour that is set here will be used in both cases, within the HMI Editor and when the project is executed on the target.
- Control Flow Time-Out – this parameter applies to the flow of data within a running project. As described in following sections the elements of a project can be connected to transfer data between each other. Every connection results in a flow of data. Depending on the total length of such a flow connection the transfer of data may need a nameable time. The value that can be set here limits the maximum allowed time for such a data flow. So this parameter can be used as time-out to avoid that a flow is running in an endless and useless loop. But please be careful: when you set a value that is too small for a specific project, a data flow could be terminated before it reaches its final destination. After the timing of such a data flow never will be 100% accurate it will be terminated on different positions for each data transmissions. That means when the time-out is too small it can result in a undefined and non-reproducible application behaviour.
- Timer Resolution – during a project is executed the application uses one master timer to trigger additional events from. This master timer is used to poll the outputs of external plug-ins (as long as they are polled cyclically). That means, the input poll cycle time that can be specified for these plug-ins must be bigger than the time specified here, in best case it should be a whole-numbered multiple of the timer resolution. Smaller timer resolution values result in a better timing accuracy but in a higher system load, bigger values result in project-global timers that are less accurate but save computing power on the target system
- External Application – here one of the options “Use local Interlock Server”, “Use remote Interlock Server”, “Use local, mirrored Interlock Server” or “Use remote, redundant Interlock Server” can be chosen for projects that have to use the Interlock Server for enhanced control and management tasks; in case one of these options is enabled it is possible to use the Plugged Devices list to configure externally running plug-ins.
  When one of the local Interlock Server modes is used, the player (or debugger) will perform everything automatically: load the project, start the Interlock Server and start all OpenPlugger instances that are necessary to handle external plug-ins. All these components then will communicate with each other over the Interlock Server on localhost.
  In case one of the options for remote Interlock Server modes is chosen, one or two network IP's have to be specified where the OpenPlayer or OpenDebugger can find the Interlock Server instances. In this case the player/debugger will not be able to start the Interlock Server or the OpenPlugger(s), that has to be done manually. Here the order is important: first the OpenIServer(s) has/have to be started, then – in case plug-ins are configured for external use – all required OpenPlugger's have to be executed and finally the OpenPlayer or OpenDebugger has to be started. Here all OpenPlugger's and the OpenPlayer or OpenDebugger have to use exactly the same project file for operation, elsewhere really weird effects resulting in heavy confusion and troubles are preprogrammed.
  To get additional information about the Interlock Server and its different operation modes please

refer the related section below.

- Touch-screen support / Activate – this check-box enables the support for touch-screens; in this case an on-screen-keyboard is opened automatically whenever the user selects a text input field and an on-screen-numpad is opened whenever an number input field is selected. This applies only for the OpenPlayer, the on-screen inputs are shown only there, not in editing or debugging mode. The following project options apply to the touch mode only
- Keyboard size factor – when the on-screen-keyboard or the on-screen-numpad is generated its size is shrunk to the minimum that is required for the currently used font and keyboard layout. In some cases this size may be too small. This size factor can be used to create a bigger layout with bigger buttons. So the keyboard size factor can be used to enlarge them by the given factor.
- Keyboard font – here a font can be chosen that is used for the on-screen-keyboard and the on-screen-numpad. Please note: a bigger font leads to a bigger size of the whole keyboard/numpad. Together with the keyboard size factor the resulting keyboard/numpad may become too big for the used display.

5.3.1.2 User Privilege Settings

Using menu “File → User Privileges” the internal user management functionality can be enabled, several types of user privileges can be defined and their order (which is equal to their priority) can be changed. As soon as the user management is enabled within this dialogue all related functionalities are available too: the User Settings within the project menu, the flow element to log in a user, the user management HMI element and the user privilege panels within the configuration of all HMI elements.

For more details about the usage of the internal user management functionality please refer to the related section above.

5.3.1.3 User Settings

The menu item “File” → “Users” can be used as soon as the internal user management was activated. Here within this panel new users can be added, existing ones can be deleted, the data of existing ones can be edited, their privileges can be changed and passwords can be set for them.

For more details about the usage of the internal user management functionality please refer to the related section above.

5.3.2 The HMI Editor of the OpenEditor

The HMI editor tab panel provides the functionality to set up a graphical interface that will be visible to the end-user when it is executed within the OpenPlayer. The total available size of that interface can be configured within the projects global settings. Normally it should be equal to the total screen size of the target platform in order to hide all other elements of the related operating system.

A new graphical element can be added quickly by right-clicking with the mouse into the HMI panel. There a pop-up menu opens that offers access to several elements like containers, buttons, input fields, display elements, images and others. To make creation of several elements of the same type easier there is also a repeat-functionality available. When function key “F2” is pressed the last element type that was created newly is created again. This new element is placed at the position of the last left mouse click within the HMI panel. Please note: this does not copy the currently selected element, it generates a new element of the same type like the last element created within the HMI editor.

On the right hand side of the editor exists a fold bar that offers an alternative way to select HMI elements. It provides the same categories of elements like the context menu. They can be opened by double-clicking the bar titles. Then the list of HMI elements is shown where one of it can be chosen. The selected element is highlighted and a slow left click within the editors drawing area adds this element at the clicked position. Selecting the same fold bar item again disables it so that an other slow left lick no longer places HMI elements.

Existing elements can be selected for further editing. Elements can be selected by surrounding them with a

selection rectangle (hold down the left mouse button within the panel and drag it over all elements that have to be selected). That selection method works for all kinds of elements. For some of the HMI elements there exists a faster way too: left-clicking it will select and highlight them. But please note: the left-click-method does not work for all kinds of HMI elements, depending on their type and internal behaviour some of them have to be selected by surrounding them with the selection box.

When one or more user interface elements are selected, they can be edited and modified. Now some additional functionalities of the pop-up menu are activated. Using that menu these elements can be duplicated, deleted, its properties can be edited and others more. A special operation that can be performed at this point is the cut/insert function. When HMI elements are cut out of the current view they are hold within an internal buffer, existing flow connections to and from these elements stay active although they are no longer shown within the Flow Editor. Now these elements can be re-inserted at a different position or within a different panel using the context menu item “Insert”. As a result of this operation they now are placed at this new position and with a new parent, additionally they re-appear within the Flow Editor together with all their flow connections.
After this operation the internal buffer is empty, the currently placed elements can't be placed again at an other position. When the “Cut” function is called twice without inserting the cut elements, the first HMI objects that have been cut are deleted by the second cut-operation.

When a single user interface element is selected, that selection is symbolized by a blue border around that element. This is a multi-functional border: Within the edges grey rectangles are shown that can be used to scale the size of the element. Here the lower right corner can be used to scale the size and to keep the aspect ratio, the other corners can be used for free scaling.

The areas between these rectangles can be used to drag an element and to change its position.

For both editing methods the same principle has to be used: hold down the left mouse button either within a rectangle or between them and drag it – the size or position will be changed according to the mouse movement.

When more than one element is selected they are marked by a similar border. Comparing to the preceding one this one does not contain the rectangles for scaling, here all the elements can only be re-positioned.

5.3.2.1 HMI Control Properties

When a user interface element is double-clicked with the left mouse button a properties dialogue is opened where that element can be configured in detail. For some of the provided user interface elements this double-click will not work due to its internal structure and behaviour, here you first have to select that element, then right-click it and choose the menu item “Selected Control(s)” → “Edit”.

The HMI Control Property dialogue that now opens consists of at least one definition panel. This panel can be used to configure name, layout, colour and other things that influence the appearance of the element. Optionally there can be a logging panel where definitions can be made to log the data handled by the element. Beside of that there can be some other panels that offer the possibility to do further definitions for that element. These additional panels provide element-specific functions and therefore are optional.

Some of the general, mandatory properties of the first panel are available twice on some element types: depending on the input signal on the digital input 0 (IN0, please refer to the following section) the state and design of that element can be changed. So the visual representation of a user interface element can be modified depending on the data flow, selection state or others.

5.3.2.1.1 General HMI Control Properties

Within the general property dialogue several parameters can be configured. This dialogue is a standard one and is used for all user interface elements. After some of the standard parameters do not apply to specific types of HMI controls some of the following parameters may be disabled in this case:

- Name – the name that is specified here is never shown to the end user, it is used to uniquely identify an object. This name will be displayed within the Flow Editor when you set an existing HMI element there. That name is also used together with the logging functionality, whenever the elements data meet the definitions that cause a log event this name is used within the log information. This name should be unique so that an element can be identified and accessed via an Interlock Server

connection which uses this name too.

- ID – right beside the name input field a number is displayed. This number is a unique identifier for every object and is used e.g. by the logging functionality to identify an object (please refer to the description of the Log Recorder flow object below for a description of this identifier)
- Min / Max – the values that can be entered here are relevant only for user interface elements that handle numerical data. Here the allowed range for such a HMI control can be specified. Please note: when the allowed range for an element is limited by these fields every input value to such an element will be clipped in case it is outside the defined range. This may lead to loss of information when values are modified this way.
- Position – these values influence the x and y position (in unit pixels) of the HMI control within the HMI Editor
- Size – similar to the position these values can be used to set a specific width (x) and height (y) for an element

The lower part of the configuration dialogue is divided into two rows, it corresponds to the digital input 0 (IN0) of the related flow object. Within that part of the dialogue the appearance of the HMI object can be controlled – and two different appearances can be defined. So the first row is active when the digital input 0 of the related object is set to LOW (default state). The appearance defined within the second row becomes active when a HIGH value is set at the digital IN0 of that object. This functionality can be used to e.g. signalise special states, information flows, limitations and other things.

Here following values can be set:

- Text – the text that is displayed with(in) the user interface element
- Font – the font that is used for the text
- Background Colour – the background colour that is set for this user interface element
- Foreground Colour – the foreground colour that is set for this HMI control
- State – some specific states that can be set for the element (like “read only”, “disabled”)

All these parameters are available within a first tab panel “Basic”. Depending on the used control some additional configurations might be necessary. In this case additional tab panes are added to this dialogue window.

5.3.2.1.2 Logging HMI Control Properties

The logging tab panel is an optional one and exists only for these element types that handle data in a way that may be interesting for logging events. If, how and which events and data have to be logged can be defined within such a logging definitions panel.

First of all there exist four log types:

- an error log that should be used for events that appear related to an error only (user has to take any actions in case of an error)
- an warning log that should be used for events that appear to a possibly dangerous state where the user has to be informed about (user should take some action in case of an error)
- an information log that should be used for events that appear to other states where the user could be informed about (user does not necessarily have to react on an information)
- an additional, freely usable event log

Every element that offers the logging definition panel and the related logging functionality, contains definitions for all four log types. Here it can be decided if and which events have to cause a reaction for which log type. The logged data itself are sent to a special flow object “Log Output” that is the target for all

these data and that offers outputs where the logging information can be fetched. For a more detailed description of this flow object please refer below.

Depending on the type of data that are handled by an HMI object it can be defined which values / changes / actions have to cause a log event. Here following possibilities exist:

1. Objects that handle digital data (like buttons)
   - logging when the value becomes LOW
   - logging when the value becomes HIGH
   - logging when the value changes
2. Objects that handle numerical data (in logging configuration a difference is made for objects that handle whole-numbered and floating point values)
   - logging when the value is greater than a defined one
   - logging when the value is smaller than a defined one
   - logging when the value is within a defined range
   - logging whenever the value changes
3. Objects that handle texts
   - logging whenever the value changes

5.3.2.1.3 User Privilege HMI Control Properties

This panel belongs to the integrated user management functionality and can be used only when this function was activated.

Here it can be defined if the user element has to become visible and enabled, visible and disabled or invisible whenever a user logs in that has granted some special user privileges. For details about how to use this user management functionality, the user privileges and user data please refer above.

The user privilege panel itself offers a matrix of visibility states and user privileges where it can be chosen if the element has to be enabled, disabled or hidden when a logged in user user has the related privilege granted.

For easier editing of this matrix below of every row there exists a button “Set all” which checks all elements of this row and therefore defines the related state for all privileges. A check box “Select visibility priority dependent” at the bottom of this panel gives the possibility to set the modes faster too: whenever a visibility state is set for one of the given privileges all privileges with lower priority (means all privileges below of the selected one) are set to the same state. This check box only modifies the behaviour during editing but does not have any influence on the logging functionality itself.

5.3.2.2 HMI Control Types

The different HMI object types are assigned to different categories. “Control” contains active elements that accept user input and emit data resulting from this user interaction. “Display” objects only can display data that are submitted to them from a different source. The “Static” elements can be used to add non-interactive objects to a user interface. Although they can not be used to emit or display data directly they can be controlled to change its appearance during runtime. The “Container” elements can be used to place HMI controls in own, staggered containers.

5.3.2.2.1 Controls

Following HMI elements of type “Control” are supported by the application directly:

- Simple Button – a simple button that can be clicked by the user
- Image Button – a button that can be pressed by the user and that displays pictures instead of the standard button design; here a predefined, internal image can be chosen or own images can be configured for normal, disabled and selected state. In last case the full path to these images has to be specified. It is also possible to embed these images directly into the project file to avoid external dependencies and complications with paths that may not exist on target system. To do that the

check box "Embed into project" has to be selected. This option can be used for all three external images but does not have any influence when one of the internal images is used. A button that can be pressed by the user and that displays pictures instead of the standard button design; here own images can be configured for normal, disabled and selected state

- Toggle Button – this is a button that stays at its selected state when it is pressed for the first time and has to be switched back to non-selected state by clicking it for a second time; so it toggles between two valid states. These buttons can be put into mutual exclusion groups, within such a group only one of all buttons can be selected. So when one of them is selected all other buttons within the same mutual exclusion group are deselected automatically. Creation of a new group or assigning a button to an existing group can be done within the HMI configuration panel, there a combo box exists that lists all existing mutual exclusion group and offers an additional option to create a new one.
- Radio Button – this kind of button is a small round element that can be used only together with other elements of same type and the mutual exclusion function as described for the Toggle Button. Usage and configuration of a Radio Button is similar to the Toggle Button.
- Check-box – this element offers the possibility to be checked (using a tick-symbol) and to be unchecked. The logical function of this element including its both states and the possibility to put it into a mutual exclusion group is similar to the Toggle Button, so for a detailed description please refer above.
- Horizontal Slider / Vertical Slider – sliders emit numerical values within a defined range and depending on the value that is set by dragging their knob to a specific position in horizontal or vertical direction
- Angular Regulator – this element displays a graphical representation of an round knob that can be rotated between defined angles and emits a numerical value dependent on that position; beside the standard parameters here the graphical start and end position, the number of displayed ticks and the colours of the parts can be configured
- Number Field – this HMI element can be used to enter whole numbers (manually)
- Floating Number Field – comparing to the preceding element here also floating point numbers can be entered
- Text Field – this is a field where the user can type all kinds of characters into
- Password Field – an input field similar to the Text Field except the fact that the typed characters are replaced by dots so that it can be used for passwords

Some of these elements can be also used for displaying data without the possibility to enter new ones, here the state flag “read only” or “disabled” has to be set within their property dialogue.

Other elements that might be listed within the same sub-menu are provided by external plug-ins.

5.3.2.2.2 Displays

Following HMI elements of type “Display” are supported by the application directly:

- Horizontal Gauge / Vertical Gauge – these elements linearly display a numeric value
- Angular Meter – with this HMI control a numeric value can be displayed using a round meter similar to a speedometer; this element offers additional configuration possibilities for the number of ticks, the start and end angle where the value has to be displayed in between and the colours of the different segments of it
- LC Numeric Display – this HMI element offers a visual representation of a number using a 7-segment display for every digit; the total number of digits to be displayed and the number of decimal place digits can be configured separately

Other elements that might be listed within the same sub-menu are provided by external plug-ins.

5.3.2.2.3 Static Elements

Following HMI elements of type “Static” are supported by the application directly:

- Rectangle – a filled rectangle aligned parallel to the X-axis; here the width and height of the rectangle are independent parameters that can be chosen freely and define width and height of it
- Line – a free line that can have any angle; width and height and position define the starting and end point of this line, using its parameter “Orientation” its orientation and therefore its direction can be flipped;
  additional configuration parameters give the possibility to define the thickness of this line and to specify special geometry for each of its ending point (like squares or circles) and their size
-
- Text Label – this is a plain text which can be placed within the HMI for describing and naming purposes
- Ellipsis – an ellipsis where the width and height defines its X and Y radius; the line width can be set within its configuration
- Frame – a not filled rectangle parallel to the X-axis where the width and height defines its size; the line width can be set within its configuration
- Image – a picture that is displayed at the specified position within the interface, here as additional parameter the path to the picture file can be given. When a path to such an external image is specified, this image can be embedded directly into the project file to avoid external dependencies and complications with paths that may not exist on target system. To do that the check box "Embed into project" has to be selected. A picture that is displayed at the specified position within the interface, here as additional parameter the path to the picture file can be given

Other elements that might be listed within the same sub-menu are provided by external plug-ins.

5.3.2.2.4 Containers

Following HMI elements of type “Container” are supported by the application directly:

- Tabbed panel – this element is the base for a combination of several tab panes that can contain different user interface elements each. The area within such a tabbed panel behaves similar to the global HMI editing area, there the same pop-up menus and functionalities are available. To add a HMI control to such a tabbed panel the pop-up menu inside this panel has to be used (means right-click within that panel to add a new element via the opening menu)
- Stacked Pane: This object is similar to the Tabbed Pane, it offers the possibility to place several panels at the same position and to give the possibility to browse between them. Different to the Tabbed panel this one does not offer any graphical tabs that can be used for page-turning, here the panels are stacked on each other invisibly. Switching to a different panel than the current one can be done within the HMI editor only via the context menu. Here the base Stacked panel has to be selected and right clicked, then the menu items **Selected Element –> Previous panel** and **Selected Elements -> Next panel** have to be chosen. During runtime the panels can be selected and switched only via its inputs (means there need to be some functionality defined via flows that access the panes).
- Additional panel – this element can be added to an existing Tabbed panel or Stacked panel only, it appends an other panel to it. The area within such an additional panel behaves similar to the global HMI editing area, there the same pop-up menus and functionalities are available. To add a HMI control to this additional panel the pop-up menu inside this panel has to be used. Several of these additional panes can be used together with one Tabbed panel or one Stacked panel to create complex, nested user interface layouts
- Single Panel – this element can be added as single, stand-alone panel only, it can be used to group elements. The area within such an additional panel behaves similar to the global HMI editing area, there the same pop-up menus and functionalities are available. To add a HMI control to this additional panel the pop-up menu inside this panel has to be used. If several of these additional

panels are used within each other they do not organize in a structured way like Tabbed Pane/Additional Pane, they in every case stay as single elements.

5.3.3 The Flow Editor Of The OpenEditor

Within the Flow Editor the user interface elements that have been set within the HMI Editor can be put together logically so that they can

- control each other
- control and influence other flow elements
- control and send data to connected devices

For every HMI object a flow object exists, that functionally corresponds to its purpose. Beside of that there are additional flow objects available within the Flow Editor that do not correspond to a user interface element but can be used to control the information flow or to access external hardware.

Within the Flow Editor the flow elements can be connected. Every connection is similar to a data flow, here during execution data are transferred from one element to an other. Such a flow looks like an electrical wire. But it is important, that these flows do NOT act like electrical devices. While an electrical line typically holds a state for a defined time; a logical state for a flow connection here is only temporarily. So such a flow connection is a path definition for transferring data. Comparing to electrical connections it acts as follows:

Electrical Signal	**Behaviour of a flow connection**
Static electrical state – a line stays at a defined value (e.g. A HIGH or LOW signal), there is no difference between signals of the same level	Static logical state – when a flow element submits a logic state on an output (e.g. LOW or HIGH) the flow lines are used to evaluate which other flow elements have to receive it, after the inputs of these flow objects are set to the related value the flow line itself has no state, the data transmission has finished. When two signals of the same level are transmitted via such a line they are separate pieces of information and can be differentiated (e.g. two HIGH-signals following directly on each other that are not separated by a LOW signal in between them can still be counted as two signals)
Electric pulse – a line is toggled for a defined time (e.g. to HIGH or LOW), afterwards the line stays at the previous value (e.g. LOW or HIGH)	Transmission of pulse – a pulse is nothing more than the transmission of two logical states, so the flow line is used to evaluate which flow elements have to receive two data packages containing two logical states (e.g. LOW for the first one, HIGH for the second one); after this operation has finished the flow lines do not have a state until the next data package is transferred

To sum it up a flow is more like a packet that is transferred over the flow lines from one element to an other. Such a flow element stores the received data until an other package is received. It is important that you keep these differences in mind while you set up a connection between your flow elements.

The flow objects itself support different data types. An output of a flow object can be connected only with an input of the same type, it is not possible to exchange the data type that is transferred over a flow line. To connect elements with different data types special conversion flow elements are available that change the data type. Following data types are supported:

Data Type	Valid Range	Colour	Remarks
DIGITAL	0 and 1 (equal to LOW and HIGH)	white	
NUMERIC	-2.100.000.000,000 ... 2.100.000.000,000	cyan	Depending on the HMI object type and the flow object operation a numeric value can be floating point or whole-numbered; when a floating point value is processed by a whole-numbered operation the number is rounded before
CHARS	Plain text including all printable alphanumerical characters	purple	
BINARY	Other data of variable size, variable purpose and variable internal format	yellow	Binary data are of a defined type and length, the data package itself contains that information internally. Different to the other data types the user has to take care about connecting elements with binary data flow types they can handle. So e.g. image data can be sent only to flow elements that are able to handle image data, the operation will fail or the data package will be dropped by the receiver when such data are sent to e.g. elements that expect sound data.

The colour that represents a special data type can be found within the flow symbols in- and outputs.

5.3.3.1 The Flow Symbols

The symbols that are used within the Flow Editor use a specific layout to symbolise their behaviour. They consist of small rectangles. On the upper side the incoming flow connections connections can be connected with the inputs. These inputs are small arrows that point into the symbol. They use the colour that represents the data type this input is able to handle.

In general it is possible to connect several flow lines with the same input. But it is recommended for digital flows and digital inputs to concatenate them using a logical flow operation (like an logic OR) and connect the resulting single flow line to the desired input.

On the lower side of a flow element the outgoing flow connections can be started to be drawn. These outputs are small arrows that point out of the flow objects symbol. They use the colour that represents the data type this output is able to handle.

Beside of that some very specific flow elements can have an additional output at the right hand side of the flow symbol. This is an overflow output. Its behaviour depends on the kind of flow element but in most cases data are sent out there that could not be handled within this element or are outside of the valid range this element is able to handle.

5.3.3.2 Adding HMI Objects

To add the flow object of a HMI element that was set within the HMI Editor before you have to right-click within the working area of the Flow Editor. Then select the option “Put Control” within the pop-up menu. It opens a dialogue that lists all HMI controls that are available to be set within the Flow Editor. When such a control is marked by a “*” it means that this HMI object already was placed within the Flow Editor. If you choose such an HMI control by double-clicking it, the flow element is replaced to the new position where you opened the pop-up menu. If you select a HMI object that doesn't exists within the Flow Editor at the moment it is put into the editing area newly.

5.3.3.3 Adding Flow Objects

Adding of flow objects that do not have a HMI part can be done very similar. First choose the position where you want to place the new flow element. Then right-click the mouse to open a pop-up menu. There choose the option “Add Element”. Below of that option you can find a hierarchy of flow element types and the related flow elements. To make creation of several elements of the same type easier there is also a repeat-functionality available. When function key “F2” is pressed the last flow element type that was created using this pop-up menu is created again. This new element is placed at the position of the last left mouse click within the Flow Editor.

On the right hand side of the editor exists a fold bar that offers an alternative way to select and add flow objects. It provides the same categories of elements than the context menu described above. These categories can be opened by double-clicking the bar titles. Then the list of flow elements is shown where one of it can be chosen. The selected element is highlighted and a slow left click within the editors drawing area adds this element at the clicked position. Selecting the same fold bar item again disables it so that an other slow left lick no longer places flow elements of this type.

Currently following types of flow elements are supported:

- Connectors – this type of flow element applies only to grouped objects: within the Flow Editor is is possible to select several flow elements and to put them into one logical/functional group – that puts them together to one single group-symbol with a bigger number of inputs and outputs. To add a new input or output connector to such a group this kind of flow elements can be added. As soon as it is placed, the related group handles one additional incoming or outgoing connection that can be used for transferring data into our out of that group. For a more detailed description of groups please refer below.
- Data Conversion – these flow elements can be used to convert data from one type to an other using different (configurable) conversion rules
- Logic Operations – when flow elements out of this category are used, logic operations (mainly with digital and numerical data) can be performed
- Mathematical – the flow elements within this category offer the possibility to perform calculations
- Flow Control – elements out of this category are able to influence data flows in an generic way (start new ones, stop flows,...)
- Input/Output – these flow elements perform IO operations with applications or devices outside of the scope of the current ControlRoom instance
- Motion – this is a special subtype of flow elements that can be used to control external hardware that is somehow motion-related (like motors)
- Data – these flow elements do perform data operations such as loading or saving specific formats, accessing databases and other things more that handle any kind of data (but not related to external hardware devices)
- Laser – this category includes all plug-in that are related to laser marking including scanner controllers, lasers and laser marking software interfaces
- Miscellaneous – within that subtype all these flow elements are collected that do not belong to one of the other types

Beside of the menu item “Add Element” the pop-up menu contains an item “Add Macros”. It contains the same subtypes like described above. Here no plain, basic flow elements can be added but complete macros that contain a more complex functionality. Internally these macros are grouped flow elements, for a description about how these functional groups can be created and used please refer below.

5.3.3.4 Editing Flow Objects

When flow objects have been added to the Flow Editor they can be edited and modified there. First of all it is possible to drag them around with the left mouse button held down on such a flow object. This allows you to reposition them.

Additional editing possibilities (like deleting an existing flow object) is possible via the known pop-up menu again. When you press the right mouse button directly over a placed flow object this menu offers additional functionalities including functions to edit the parameters of the flow object. Depending on the type of object here a dialogue opens where the configuration and behaviour of the flow object can be changed. All these flow element dependent configuration dialogues have some common parameters that can be set up for all kinds of flow elements:

- Name – the name of the flow element; here a unique name without any blanks (spaces or tabs) has to be chosen. This name can be used to access the data of such a flow element within the Interlock Server (here the related option has to be enabled within the project settings). As an example: when a flow element is named “DigitalInOut_13” its related name within the data space of the Interlock Server will be “/DigitalInOut_13/in” for its inputs and “/DigitalInOut_13/out” for its outputs.
  In case the flow element corresponds to an HMI element in HMI Editor this name is the same like it is known from the HMI configuration dialogue, changing it here also changes the name specified within the HMI Editor.
- Interlock Server / Map outputs to server – this option requires the Interlock Server option to be enabled within the global project configuration; when it is set, the outputs of this flow element are submitted to the Interlock Server every time they change (so this option enables a data flow from player to Interlock Server)
- Interlock Server / Allow modification - this option requires the Interlock Server option to be enabled within the project configuration; it defines a connection between the server and the inputs of this flow element: whenever the related data element is changed within the data space of the server the input(s) of this flow element is/are set to the new value which is sent from the Interlock Server (so this option enables a data flow from a connected Interlock Server to the player)

5.3.3.5 Creating and Editing Flow Connections

When there are at least two flow objects placed within the Flow Editor they can be connected using flow lines. As described above these lines specify the data flow from one flow object to an other. Such a flow connection has to be started at the output of one flow object and has to end at the input of an other one. The flow lines can be horizontal and vertical. To start such a line move the mouse over the outputs source flow object. These outputs are symbolised by small arrows at the bottom of the flow object symbol that point out of it. When you have localised the output the appearance of the mouse pointer changes to crossed lines. Now the left mouse button can be held down. As long as the left mouse button is pressed a flow connection is drawn. So when the mouse is moved to the input of the target flow object now, the related flow line is drawn.

The inputs of a flow object are symbolised by small, coloured arrows at the top of the flow symbol that point into that symbol. When the position of such an input was found during drawing a flow connection the flow line is highlighted using green colour. When the mouse button is released now the connection between output and input is established automatically – as long as both have the same data type. So when an output arrow is connected with an input arrow of the same colour both have the same data type and therefore can be connected successfully.

An existing flow connection can be edited afterwards. So it is possible to change the position of the flow lines: vertical flow lines can be moved in horizontal direction, horizontal flow lines can be dragged to be moved to a different vertical position.

PLEASE NOTE: The Flow Editor would also allow it to create data flow loops. Such loops exist when a flow starts from the output of an object and is connected with the input of the same object. That may be a direct connection or a connection with some additional flow elements in between. After such connections could cause endless loops the software tries to detect them during runtime. So when a data flow reaches an object where it was started from it sets its value to the input and is terminated afterwards, no loop is performed.

Such problems can be detected within the OpenDebugger, there the whole operation can be stopped in case such a problem appears. The debugger then will print out an information that points to the element where this data flow was terminated.

The software in general tries to avoid such possibly endless loops for all flow elements. Nevertheless there are conditions where such a loop is required, e.g. when a counter is implemented that counts up or down a value. So for several flow elements there is a possibility within the properties dialogue to disable the check for an endless loop. That's true e.g. for mathematical flow elements. **This functionality has to be handled with care!** When such a potentially endless loop is defined and explicitly enabled there has to be a condition to interrupt such a loop (e.g. by using a gate element to interrupt the data flow of the loop) or at least a delay of a nameable time has to be added. When this is not done, the project may consume much processing time in OpenPlayer and may stop or cumber other processes on the target machine and it may consume much computing power. So in such cases a very exact and accurate debugging is necessary.

5.3.3.6 Grouping Flow Objects

Within the Flow Editor one or more flow elements that belong together can be put into a group. To do that they first have to be surrounded by a rectangular box that specifies the elements that have to be put into one logical group. That can be done by holding down the left mouse button and dragging the mouse around the desired elements. This draws a thin rectangle from the starting point of the mouse to its current position which can be used to include flow elements. When the mouse button is released next the right mouse button has to be pressed and within the opening pop-up menu the item “Group Elements” can be selected.

Now the current project is modified: all the selected flow-elements are replaced by one bigger box that has as much inputs and outputs as all the flow elements had to connect with the not selected elements (which now are the “outer world”). This bigger box is a functional group that can be entered by double-clicking it. Now the contents of the group are displayed – and these contents are all the flow elements that have been selected before. By double-clicking somewhere in the editor where no element or flow connection line is located such a group can be left.

Such groups can be nested, means a group itself can contain similar functional groups too. A group can contain only flow elements, the flow-part of an HMI element can not be put into a group, they have to stay on top level.

A group can be saved separately, here the .APCG-format is used. To do that the right mouse button has to be pressed while it is positioned over a group and the pop-up menu item “Save Group” has to be selected.

On the other hand saved groups can be loaded into an existing project to extend it by the functionalities encapsulated within that group. To do that the right mouse has to be pressed at a position where the new group symbol has to be located and the pop-up menu item “Load Group” has to be selected. Now a file dialogue opens that gives the possibility to choose the correct .APCG-file.

When a group is right-clicked and the menu item “Edit Element” is selected a configuration dialogue opens. Here the name of the group can be changed and the names of all inputs and outputs of that group – that are similar to the connections within that group – can be changed. Beside of that in panel “Basic” a category can be chosen this group belongs to. This category is important in case a group has to be used as functional macro.

Such functional macros can be added like normal flow elements out of the Flow Editors pop-up menu. The category that can be specified for a group is similar to the categories the macros are using. So if a group is categorised within its properties dialogue and when it is saved within the folder “macros” of the applications installation or library directory it is added to the pop-up menu within the Flow Editor automatically.

5.3.3.7 The Flow Objects of the Flow Editor

Following all available flow objects are described together with their functionality.

5.3.3.7.1 Flow Objects of HMI Controls

All flow objects that belong to a HMI element use digital input 0 and output 0 for the same functionality.

Depending on the value that is set at digital IN0 the appearance of the user interface element is changed. Afterwards the same value is sent out at digital OUT0. That functionality is fixed and common to all HMI elements, therefore it is not described separately within the following sections.

5.3.3.7.1.1 Flow Objects of Control-type HMI Elements

Simple Button:

Digital IN1 – set the buttons state, this kind of button can be toggled for a short time only also if the input keeps a 1 for a longer time. Therefore this functionality in most cases will not make much sense and exists mainly for the sake of completeness.

Char IN3 – sets a new label; changes on IN0 will overwrite this label with the predefined ones

Digital OUT1 – pass through of digital IN1 and output of the button state; when the button is pressed two data flows using a digital 1 and a digital 0 are emitted directly after each other

Image Button:

Digital IN1 – set the buttons state, this kind of button can be toggled for a short time only also if the input keeps a 1 for a longer time. So this functionality in most cases also will not make much sense and exists mainly for the sake of completeness.

Binary IN3 (E) – set binary image data for the enabled state of the button

Binary IN4 (S) - dynamic binary image data for the selected state of the button

Binary IN5 (D) – set binary image data for the disabled state dynamically here

Digital OUT1 – pass through of digital IN1 and output of the button state; when the button is pressed two data flows using a digital 1 and a digital 0 are emitted directly after each other

Toggle Button:

Digital IN1 – set the buttons state

Char IN3 – sets a new label; changes on IN0 will overwrite this label with the predefined ones

Digital OUT1 – pass through of digital IN1 and output of the user-triggered button state; when the button is pressed one data flow is emitted that submits a 0 or a 1 according to the buttons state

Horizontal Slider, Vertical Slider, Angular Regulator, Number Field and Floating Number Field:

Digital IN1 – when it is configured within the flow properties dialogue of this element this input can be used to trigger a data submission on Numerical OUT6 or OUT7

Numeric IN2 – input for dynamically setting the minimum allowed value, this input overwrites the default settings defined within the objects properties

Numeric IN3 – input for dynamically setting the maximum allowed value, this input overwrites the default maximum defined within the objects properties

Numeric IN7 – when a number is transmitted to this input it is used and displayed as new value

Numeric OUT6 and numeric OUT7 – here the value of the element can be transmitted out to other flow elements; within the flow configuration dialogue of this element it can be decided under which conditions such a transmission is started at which output

Text Field and Password Field:

Digital IN1 – when configured within the flow properties dialogue of this element this input can be used to trigger a data submission on Numerical OUT6 or OUT7

Char IN7 – when a text is transmitted to this input it is used and displayed as new value

Char OUT6 and char OUT7 – here the value of the element can be transmitted out to other flow elements; within the flow configuration dialogue of this element it can be decided under which conditions such a transmission of text data is started at which output

5.3.3.7.1.2 External Flow Objects of Control-type HMI Elements

BeamConstruct:

The BeamConstruct plug-in provides full functionality and wide parts of the user interface of separate BeamConstruct application (please refer to description of BeamConstruct below). Comparing to the BeamConstruct2Control plug-in it also encapsulates all hardware control, means no scanner/laser/motion controllers have to be connected to this element in Flow Editor. The plug-in shows the user interface of BeamConstruct containing the tabbed panes, the drawing view and the element tree. All other buttons and user interaction functionalities can be added via the control input. Here command values can be set to cause some actions similar to real user interaction. These command values can be generated e.g. by buttons that are converted to numbers.

Comparing to the stand-alone application the BeamConstruct plug-in behaves different in a few, important points. So it manages default settings too, comparing to BeamConstruct main application these default settings are loaded every time after a new project file is opened. Here all hardware-settings of the loaded project are overwritten to reflect the configuration of the used machine. This includes all connected devices but not the pens and their parameters.

The plug-in provides following in- and outputs:

Char IN1 (FILE) – when a path to a .BEAMP project file is given here this project is loaded into BeamConstruct plug-in

Char IN2 – corresponds to all OpenAPC inputs within the current project, replaces the contents of related dynamic elements at input 2 with the text given here

Char IN3 – corresponds to all OpenAPC inputs within the current project, replaces the contents of related dynamic elements at input 3 with the text given here

Char IN4 – corresponds to all OpenAPC inputs within the current project, replaces the contents of related dynamic elements at input 4 with the text given here

Numeric IN5 (CMD) – command input, here specific input numbers cause a reaction within the plug-in and can be used to trigger user interface operations:

0 – stop a currently running marking process

All four-digit numbers correspond to menu items and/or tool bar buttons and their functionality within the main application, when such a number is received at this input the related operation is invoked (for a more detailled description please refer the BeamConstruct documentation below):

1001 – new project
1002 – load existing project file
1003 – import custom file format (as supported by BeamConstruct: vector, 3D and raster image formats)
1004 – save the current project using a known name
1005 – save the current project using a new name
1101 – export vector data in CSV format
1102 – export vector data in HPGL/PLT format
1103 – export layer vector data in CLI format
1006 – open general project settings dialogue
1007 – open pen settings dialogue
1008 – save the current configuration as default settings
1009 – load default settings (this command has normally not to be used because the BeamConstruct plug-in overwrites all hardware-related settings out of a loaded project with the default settings immediately after a new project file was loaded)
1010 – save project with options
1011 – import custom 3D file format (as supported by BeamConstruct: 3D meshes)
1013 – insert a new .BEAMP project file without deleting already existing project data
1014 – save hardware settings into a .BEAMH file
1015 – save geometries, control elements and pen definitions into a .BEAMV file

1016 – load hardware settings from a .BEAMH file and leave geometries, control elements and pen definitions unchanged
1017 – load geometries, control elements and pen definitions from a .BEAMV file and leave current hardware settings unchanged
1018 – open general machine settings dialogue
1019 – import custom file format (as supported by BeamConstruct: vector, 3D and raster image formats) in case of a 3D mesh add the data to an already existing Active Slice Group instead of creating a new, separate one
1020 – import custom 3D file format (as supported by BeamConstruct: 3D meshes) annd add it to an already existing Active Slice Group instead of creating a new, separate one
2001 – undo last operation
2002 – redo last un-done operation
2101 – concatenate all selected elements into one group
2102 – concatenate all selected elements into a splitgroup
2103 – concatenate all selected elements into a movegroup
2104 – ungroup the elements out of the selected group
2105 – duplicate the selected element
2201 – merge the geometries of the selected element into a static one
2202 – split the geometries of the selected element logically
2203 – optimise the geometries of the selected element
2204 – reduce the geometries of the selected element using a lossy algorithm
2205 – extrude the selected element into third dimension
2301 – freeze the currently shown background video
2302 – calibrate the currently connected camera
2303 – teach fiducials
2304 – load fiducials
2305 – save teached fiducial(s)
2306 – drop teached fiducial(s)
2307 – drop camera calibration data
2308 – toggle image capturing on or of (when configured to be started manually in global project settings)
3001 – start process simulation
3002 – opens the marking dialogue to let the user control the marking process
3003 – marks the current project once, here the mark dialogue is opened to let the user stop the marking process if necessary; the dialogue is closed automatically as soon as marking was finished
3006 – increment the elements in current project
3007 – decrement the elements in current project
3008 – reset the elements in current project to their default/start values
3011 – open devices (scanner card, motion controller and image capture device as configured)
3012 – close opened devices
3013 – open devices when not already done and show scanner card state dialogue
3015 – marks the selected elements out of the current project once, here the mark dialogue is opened to let the user stop the marking process if necessary; the dialogue is closed automatically as soon as marking was finished
3020 – this command starts marking of the current project in a loop while opening a mark dialogue which consists of a stop-button only. This mark dialogue and mark-loop can be left only by pressing the stop-button or by clicking the close-symbol of the mark-dialogue itself.
3021 – start the pilot laser with the mark dialogue visible and in pilot mode “element outline”
3022 – start the pilot laser without a visible mark dialogue using pilot mode “element outline”
4001 – show the "about"-dialogue
4002 – show license information
4003 – reset the current license
4004 – show the "credits"-dialogue
4006 – generate a bugreport

All five-digit numbers correspond to remaining tool bar buttons and their functionality within the main application, please refer to section „BeamConstruct“ below to get detailled information about them:

10001 – zoom into view
10002 – zoom out of view
10003 – zoom to view full working area
10004 – zoom to fully view selected element
10005 – enable pointdrag mode for selected element

10006 – log in a user
11002 – create a new bezier primary element
11003 – create a new circle primary element
11004 – create a new delay primary element
11005 – create a new dot primary element
11006 – create a new external trigger primary element
11007 – create a new laser output primary element
11008 – create a new line primary element
11009 – create a new motion primary element
11010 – create a new polygon primary element
11011 – create a new rectangle primary element
11012 – create a new spiral primary element
11013 – create a new star primary element
11014 – create a new text primary element
11015 – create a new triangle primary element
12001 – set a hatch additional element to the selected one
13001 – add a OpenAPC input element to the currently selected one
13002 – add a serial/date/time input element to the currently selected one
14001 – add a curve postprocessing element to the currently selected one
14002 – add a sine postprocessing element to the currently selected one

Numeric OUT1 (ERR) – output for error codes in case some specific operations fail

Digi OUT6 (BUSY) – signals if marking is in process

This plug-in is located in hmiplugins/libio_hmi_beamconstruct

Linear Regulator:

This HMI plug-in is able to display a numeric value within a linear field and gives the user the possibility to change this value. Data, usage and handling is very similar to the internal UI element "Vertical Slider". But beside of the standard configuration parameters some additional data can be set that influence the appearance of this element:

- Limits - the colour for the limit values
- Value - the colour of the displayed value
- Border - the colour of the overall border of the meter
- Ticks - the colour of the ticks
- Number of Ticks - the total number of displayed ticks, this value should fit to the minimum and maximum allowed value because for every tick the appropriate value is printed

This plug-in is located in hmiplugins/libio_hmi_linearregulator

Pinpad:

The Pinpad plug-in offers a possibility for numerical data input with a set of number buttons 0..9, a Clear- and an OK-button like it is known from payment card terminals. It can be used to get whole-numbered values from the user:

Digital IN0 - set the selection mode for the list box and apply the parameters according to the configuration done for the two possible input states LOW and HIGH

Numeric IN7 – sets the given value into the display and overwrites a formerly entered value

Numeric OUT6 – emits the entered value as soon as the OK-button was pressed by the user

Numeric OUT7 – emits the entered value as soon as the OK-button was pressed by the user

This plug-in is located in hmiplugins/libio_hmi_pinpad

Position Correction:

This is a complex plug-in that lets the user perform position corrections via touch. It provides buttons for change of position as well as two rotation buttons. The translation and rotation step size can be changed too, all operations can be done completely without any keyboard. The position correction values are emitted using a binary data structure:

Binary OUT7 (POSCORR) – the position correction data are updated whenever the user presses one of the translation or rotation buttons

Symbol Button:

Using this plug-in some common symbols can be used within a visualisation and without the need to use external images. It offers a resource-saving possibility to use these symbols easily. Once such an element was added to the HMI Editor within the configuration the colours and type of the symbol can be chosen. Following symbols are supported at the moment:

- Simple Knob - a round knob
- Horizontal Valve - symbol for a valve to be connected with horizontal pipes
- Vertical Valve - symbol for a valve to be connected with to vertical pipes
- Pump (right) - a pump that pumps to the right direction
- Pump (left) - a pump that pumps to the left direction
- Pump (up) - a pump that pumps up
- Pump (down) - a pump that pumps something down
- Lamp - electrical symbol for a lamp
- Horn - electrical symbol for a buzzer or horn
- Switch - electrical symbol for an open switch, this symbol changes its appearance when it is enabled
- Motor - electrical symbol for a motor

Such a symbol button behaves like a toggle button, a first click enables it / turns it on, a second click to it turns it off. Beside of that it also can be switched via its input:

Digital IN0 - input to turn on/off the symbol

Digital OUT0 - output that sends a LOW signal when the symbol is turned off and a HIGH signal when it is turned on, here it doesn't matters if that happened by user interaction or by sending a signal to IN0

This plug-in is located in hmiplugins/libio_hmi_symbolubtton

Text Listbox:

Using this plug-in a list box can be added to a user interface. Within such a list box several lines of text can be displayed, new lines can be added and the user can choose one by single- or double-clicking it. Beside the standard configuration options this plug-in does not offer any additional parameters. The complete custom handling of the plug-in is done via its IO's:

Digital IN0 - set the selection mode for the list box and apply the parameters according to the configuration done for the two possible input states LOW and HIGH

Numeric IN1 (SEL) - select an entry within the list box that is located at the row specified by the number given at this input

Digital IN2 (CLR) - clear the complete list box and remove all elements out of it

Char IN7 - append this text to the end of the list within the box

Digital OUT - the through-connection of the selection mode input

Char OUT4 (1x) - whenever a list box element is single- or double-clicked this output emits the text that is stored at the selected position

Char OUT5 (2x) - whenever a list box element is double-clicked this output emits the text that is stored at the selected position

Numeric OUT6 (1x) - whenever a list box element is single- or double-clicked this output emits the index number of the selected row

Numeric OUT7 (2x) - whenever a list box element is double-clicked this output emits the index number of the selected row

This plug-in is located in hmiplugins/libio_hmi_list box

5.3.3.7.1.3 Flow Objects of Display-type HMI Elements

Horizontal Gauge, Vertical Gauge, Angular Meter and LC Numeric Display:

Numeric IN7 – the number that is received on this input is used as new value to be displayed

Numeric OUT7 – pass through of numeric IN7, here the received number is available for transmission to the next flow element

5.3.3.7.1.4 External Flow Objects of Display-type HMI Elements

Analogue Clock:

Using this element different representations of an analogue clock can be added to a project. It supports different styles of clock faces and freely editable colours ans sizes of the hands. All these parameters can be defined within the configuration panel of this element:

- Hour hand – colour and size of the hour hand for LOW and HIGH state of digital input 0
- Minute hand – colour and size of the minute hand for LOW and HIGH state of digital input 0
- Second hand – colour and size of the second hand for LOW and HIGH state of digital input 0
- Type – appearance of the clock, here a clock face can be chosen; dependent on the type of clock face it might be necessary to change the colours of the hour/minute/second hand in order to have a good contrast

The Analogue Clock itself does not contain or generate any time information and therefore will not display anything useful without an appropriate data source that delivers the time information:

Digital IN0 – set the selection mode for clock and apply the parameters according to the configuration done for the two possible input states LOW and HIGH

Numeric IN7 – set the current time information in UNIX format (the number of seconds elapsed since 1st of January 1970; the flow element Clock is able to deliver such a time format

Digital OUT - the through-connection of the selection mode input

Numeric OUT7 – the through connection of the time information as set as IN7

This plug-in is located in hmiplugins/libio_hmi_aclock

Chart

This plug-in can be used to display up to 8 values in a chart. Here different bar- and pie-chart styles are supported. Data for this chart have to be numeric values and can be delivered by other elements. The chart HMI element does not visualise the change over time (here the Plot2D element has to be used) but the actual state.
The behaviour and layout of the plug-in can be configured within its properties that are accessible in HMI-Editor. In tab-pane "Settings":

- Chart Type – here the kind of chart can be selected that has to be drawn by this plug-in; the bar-charts are able to display both, positive and negative values while the pie-chart only shows positive values, negative ones are dropped and don't become part of the pie

- Used Input Values – this input field specifies how much of the up to 8 data inputs are used for calculation of the chart, when a value smaller than 8 is given here, data at disabled inputs are ignored and not shown within the chart
- Show Axis Value – when this checkbox is set the minimum and maximum values are shown beside the axis; this field applies to bar charts only and does not have any influence on the pie chart
- Labels – here it can be specified if every value shown within the chart should have no label,should show the name of the value or the current value; when the name is chosen to be shown it can be specified in the following text fields
- Name – specifies the name of a value; this name is shown within the chart when "Labels" is set to "Names"

In second tab-pane "Colours" it is possible to choose or change a specific colour for every bar that is shown within the chart.
The chart is updated whenever a new value is set at one if the Numerical input IN0..IN8. Please note: When input-values change very fast and often this may cause a nameable load to the graphics which in case of slow hardware could slow down the whole system.

Flow Indicator:

Using this plug-in the flow direction and speed of moved materials can be visualised. It consists of an area where some kind of wave is shown that moves into a definable direction. Here following additional parameters can be used to modify the behaviour of the element:

- Start Colour / End Colour – the two colours that are used to create the flow indicators waves
- Border Colour – colour of the border around the waves
- Flow Speed – movement speed of the waves
- Flow Direction – can be used to select if waves have to move horizontally or vertically

The flow speed can be modified via the related numerical input, a negative value changes the flow direction and 0 stops the movement completely:

Numeric IN7 – sets the speed of the currently shown flow, a value that is set here overrides the default value of the default configuration

Linear Meter:

This HMI plug-in is able to display a numeric value within a linear meter. Data, usage and handling is very similar to the internal UI element "Gauge". But beside of the standard configuration parameters some other data can be set that influence the appearance of this element:

- Limits - the colour for the upper and lower limits
- Value - the colour of the displayed value
- Border - the colour of the overall border
- Ticks - the colour of the ticks
- Number of Ticks - the total number of displayed ticks, this value should fit to the minimum and maximum allowed value because for every tick the appropriate value is printed

This plug-in is located in hmiplugins/libio_hmi_linearmeter

Plot2D:

Using this plug-in up to six independent graphs can be displayed within a two-dimensional coordinate system. The data for these graphs can be delivered from other flow elements as numeric data. The graphs are updated every time a new value is set at an input. So when a graph has to be displayed that shows a change over time the input has to be updated cyclically not only when a new value is available. That can be done by using a triggered numeric gate where the clock-input is connected to a clock source that triggers it using the desired period.

This plug-in offers several possibilities to set-up and configure the behaviour and the layout of the plotted graphs. Beside the standard configuration panels of the HMI object the following additional panels can be used for setting up the parameters.

Configuration:

- Initial Graphs – the managed graphs can be enabled and disabled to show and to hide them. This parameter specifies how much graphs (counted from 1) are enabled initially
- Maximum Graphs – the maximum number of graphs this flow element will handle; here up to six graphs can be configured for single axis modes and up to three graphs when two inputs are combined to feed the X and Y axis with data (please see the drawing direction below)
- Direction – specifies the direction the graphs move to when a new value is set at an input; here one of the modes "X" (the input data influence the height of the graph while the width is updated automatically), "Y" (the input influences the width, the height is incremented automatically) and X/Y combined (the inputs 1 and A, 2 and B, 3 and C are combined so that one sets the X value and one the Y value) can be chosen
- Maximum Graph Length – specifies the maximum number of data a graph may collect until it is reset
- Data Skip Mode – this mode specifies the method of skipping/dropping data in case the maximum length of a graph was reached; here "Delete at beginning" removes the first data while new data are still appended at the end of the graph (so it continues endless in its chosen direction), "Reset everything" drops all data and starts drawing from the beginning and "Overwrite from beginning" starts drawing from the beginning while the old data aren't rejected before, they are still available until they are overwritten
- Scientific Notation – when numbers are displayed at axes and this option is set, these numbers are shown in scientific notation (as floating point numbers with exponent)
- Logarithmic Y Axis – here displaying of data along Y-axis can be switched to logarithmic mode

Graphs:

Here every of the parameter sets described below exists once for every of the up to six graphs and for every of the two input selection states HIGH and LOW:

- Colour - the colour the graph has to be drawn with
- Line Size - the thickness of the drawn line in unit "pixels"
- Line Style - the style of the graph, here "Simple" is a plain line with no interruptions, "Dotted" enables a mode where no lines are drawn but a single dot is shown at the position of every dataset, "Short Dashes" enables a graph consisting of short dashes (the line is interrupted in regular distances) and "Dots and dashes" enables a line consisting of a dot and a short line alternating

Digital IN0 - input for the selection state, depending on this input one of the two configurations "HIGH" and "LOW" is chosen (that was done in configuration panel "Graphs").

Numeric IN1 (CMD) - using this input the state of the graph panel can be changed during runtime. Every command exists of a number "xy" where the decimal digit "x" can be in range 1..6 and specifies the number of the graph (the numbers 4..6 are assigned to the graphs A..C). The digit "y" specifies the operation that has to be done with this graph: 0 disables it so that it is no longer visible (but still collects data), 1 enables it to become visible, 2 resets all data completely so that it starts from the beginning (this command acts independent from the chosen Data Skip Mode)

Numeric IN2 (1) - input data from graph 1

Numeric IN3 (2) - input data from graph 2

Numeric IN4 (3) - input data from graph 3

Numeric IN5 (A) - input data from graph A

Numeric IN6 (B) - input data from graph B

Numeric IN7 (C) - input data from graph C

This plug-in is located in hmiplugins/libio_hmi_plot2d

Vector2D:

This plug-in can be used to display two-dimensional vector data out of sequences of binary control structures. These structures can be generated e.g. with the “CSV to Control”, “HPGL to Control”, “BeamConstruct to Control” plug-ins (please refer below). Thus the Vector2D-plug-in offers a possibility to implement a preview of how the data look like or what was processed last. It offers several configuration possibilities that influence the behaviour and appearance of the 2D-view that can be set-up via the properties dialogue within the HMI-Editor. There the tab-pane “Configuration” contains the following parameters:

Maximum Number of Vectors – defines how much vectors are displayed within the 2D view at maximum, when more data than the amount specified here are transmitted to the plug-in the oldest ones are dismissed

Field Left Position – defines the left position (X-value) of the visible field in unit mm

Field Upper Position – defines the upper position (Y-value) of the field displayed by this plug-in in unit mm

Field Size – specifies the total size of the shown 2D-view in unit mm, this parameter together with the two preceding ones specifies which section of the whole coordinate system is shown by the Vector2D-plug-in

Swap X and Y – exchanges the incoming X and Y values and therefore flips the display

Mirror X – mirrors the incoming X-values along the Y-axis of the coordinate system

Mirror Y – mirrors the incoming Y coordinate values along the X-axis of the coordinate system

Within the “tab-pane Colours” following definitions can be made for both activation states LOW and HIGH:

Border – the colour of the border drawn around the 2D-view

Line 0% Power – the colour that is used for lines where the power value within the related, incoming binary control structure is set to 0%

Line 100% Power – the colour that is used for lines where the power value within the related, incoming binary control structure is set to the maximum of 100%; all other values between 0% and 100% are interpolated so that the change of power is visualised by a colour fade

Line Size – the size of the lines to draw the vector data with (in unit pixels)

Vector2D supports the following plug-in-specific IO's that can be used within the Flow Editor:

Digital IN1 (RES) – reset all internally stored data and clear the 2D-view completely

Binary IN7 (CTRL) – input for the binary control structure data that have to be shown; they become visible only when there are at least two consecutive coordinate values where the tool is turned on and when they are located within the visible range that was defined by the field parameters, movement data that are performed with the tool turned off are not shown

The plug-in itself is located in hmiplugins/libio_hmi_vec2d

5.3.3.7.1.5 Flow Objects of Static HMI Elements

Static HMI elements that are not described below do not offer additional inputs or outputs beside the digital IN0 and digital OUT0 that is described above

Text Label:

Char IN3 – sets a new label; changes on IN0 will overwrite this label with the predefined ones

Image:

Binary IN3 – set binary image data dynamically during runtime through this input; changes on IN0 will NOT overwrite this image and will NOT switch back to the default one

5.3.3.7.1.6 External Flow Objects of Static HMI Elements

Pipes:

This plug-in can be used to design pipes easily. These pipes can't be selected and don't change their layout depending on any input. They are mainly a possibility to quickly insert fancy pipes into a HMI layout without the need to load external graphics.

Within the plug-ins configuration panel the type of pipe can be chosen. Every type is available in two colours for a steel pipe in metallic look and for a black plastic pipe:

- plain horizontal pipe for connections in horizontal direction
- plain vertical pipe for point-to-point connections in vertical direction
- T-piece to connect three pipes with each other, here it can be chosen out of four T-pieces that differ in their direction where the third pipe can be connected at
- elbow to switch from horizontal to vertical pipe or from vertical to horizontal pipe, here also four different elbow elements are available that differ in the direction where the two open ends of the elbow are oriented into

This plug-in is completely static and does not react on any input

The plug-in itself is located in hmiplugins/libio_hmi_hardpipes.

5.3.3.7.1.7 Flow Objects of Container-type HMI Elements

Tabbed Pane:

Digital IN1 / SEL – select the first tab and put it in foreground

Digital IN2 / EN – disable (LOW) or enable (HIGH) the first tab

Numeric IN3 / SEL – select the tab with the number that is sent to this input, here other tabs than only the first one can be accessed, if no tab is available at the position specified by this number nothing is done

Numeric IN4 / EN – enable the tab with the number that is sent to this input, here other tabs than only the first one can be accessed, if no tab is available at the position specified by this number nothing is done

Numeric IN5 / DIS – disable the tab with the number that is sent to this input, here other tabs than only the first one can be accessed, if no tab is available at the position specified by this number nothing is done

Single Panel:

Digital IN1 / SEL – select this tab and put it in foreground, the same operation can be done with the numeric IN3 of the parent Tabbed panel or Stacked Pane

Digital IN2 / EN – disable (LOW) or enable (HIGH) this tab, the same operation can be done with the numeric IN3 and IN4 of the parent Tabbed panel or Stacked Pane

Stacked Pane:

Digital IN1 / SEL – select the first of all stacked tabs and put it in foreground

Digital IN2 / EN – disable (LOW) or enable (HIGH) the first tab, this operation puts the tab in background when it is visible, otherwise nothing is changed

Numeric IN3 / SEL – select the tab with the number that is sent to this input, here other tabs than only the first one can be accessed, if no tab is available at the position specified by this number nothing is done

Numeric IN4 / EN – enable the tab with the number that is sent to this input, here other tabs than only the first one can be accessed, if no tab is available at the position specified by this number nothing is done

Numeric IN5 / DIS – disable the tab with the number that is sent to this input, here other tabs than only the first one can be accessed, if no tab is available at the position specified by this number nothing is done

5.3.3.7.1.8 Flow Objects of Miscellaneous-type HMI Elements

User Management Panel:

This is a complex HMI element that can be used together with the internal user management functionality. Therefore this element can be added to an HMI only when the user management is used and enabled. The panel itself offers the same functionality like the user management dialogue within the OpenEditor but it gives the possibility to offer these functionalities to an end user that operates an HMI within the player. So within this HMI element users can be added, deleted and edited in the same way like it can be done within the OpenEditor.
This panel itself does not offer any in- or outputs except the two standard digital IO's 0 for the selection state of the element. All other data are handled internally.

Digital IN0 - set the selection mode for the whole user management panel and apply the parameters according to the configuration done for the two possible input states LOW and HIGH

Digital OUT - the through-connection of the selection mode input

This HMI element can exist exactly once within a project.

5.3.3.7.2 Stand-Alone Flow Objects

For flow objects that do not correspond to a visual representation within the HMI no general set up is specified, the available parameters completely depend on their specific functionality. So within the following sections a full description of the existing, application-internal flow objects and external flow-plug-ins is given together with the purpose of their in- and outputs.

5.3.3.7.2.1 Data Conversion Flow Objects

Digital to Number:

This element can be used to convert digital input values and combinations of digital inputs to numbers. Within the configuration of the object the user can decide which inputs are mapped to which outputs and which conversion method has to be used. So the user can decide if the binary value on a set of selected inputs has to be converted to its (decimal) number or if specific values have to be output dependent on the state of an input.

Digital IN0 / CLK – depending on the configuration of this element the digital input 0 can be used either as data input for binary conversion to a decimal number or as clock. In last case these outputs, that are configured to use the clock signal, converted data are sent only when a digital 1 is received on this input.

Digital IN1 .. digital IN7 – data inputs to be converted to numbers according to the configuration of this flow object

Numeric OUT1 .. numeric OUT7 – outputs that send the numbers according to the configuration of the flow element

Digital to Characters:

This element can be used to convert digital input values and combinations of digital inputs to text. Within the configuration of the object the user can decide which inputs are mapped to which outputs and which conversion method has to be used. So the user can decide if the binary value on a set of selected inputs has to be converted to a text consisting of a combination of “0” and “1” that is equal to the input pattern or if specific values have to be output dependent on the state of an input.

Digital IN0 / CLK – depending on the configuration of this element the digital input 0 can be used either as data input for binary conversion to a text or as clock. In last case these outputs, that are configured to use the clock signal, converted data are sent only when a digital 1 is received on this input.

Digital IN1 .. digital IN7 – data inputs to be converted to text according to the configuration of this flow object

Char OUT1 .. char OUT7 – outputs that send the created text according to the configuration of the flow element

Digital to CMD/Value-Pair:

A CMD/Value pair is a combination of a value and a command name that is assigned to this value. Such pairs can be used to connect to external applications or to identify a value for other purposes. Using this element digital data can be converted to such pairs. Within the configuration of the flow element a name can be assigned to every input. The values that are received at this input will be combined with that name (used as command) to the pair of command and value.

Digital IN0 - Digital IN7 - the inputs for the values that have to be combined with the command name

Char OUT0 (CMD) – the output for the command, whenever an value is set to an input an command is emitted here as specified within the configuration

Digi OUT1 - the output for the value that is assigned to the command, whenever an value is set to an input an command is emitted together with this value

CMD/Value-Pair to Digital:

This element performs the conversion back from Command/Value pairs to plain values. Within the configuration of the flow element a name can be assigned for every input. When a command is sent into this flow element that is identical to this name the value is emitted at the related output

Char IN0 (CMD) – the input for the command, here the command name that is assigned to the value is expected

Digi IN1 - the input for the value that is assigned to the command

Digital OUT0 - Digital OUT7 - the outputs for the values that could be identified by the configured command; when the string that is set to Char IN0 could not be found within the configuration of the flow element nothing is output here

Number to Digital:

With this flow object a conversion from numbers to digital signals is done that internally is performed as comparison. Dependent on the input value specific digital output values are created (LOW, HIGH or pulse).

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – inputs for the numbers that have to be converted

Digital OUT0 / OVL – the clock overflow signal is emitted whenever an overflow condition appeared during the conversion for outputs, that react on the external clock. This overflow clock signal e.g. can be transferred to the CLK input of the next “Number to Digital” converter

Digital OUT1 .. digital OUT7 – the output signals that are emitted dependent on the input values and the configuration of this object

Overflow – the numeric overflow output; when the conversion operation did not match the rules defined within the configuration of the flow element so that no digital output signal could be found that belongs to the number at an input, this number is sent to the overflow output. Together with the OUT0 / OVL signal both can be used for a next conversion step to check some more conversion conditions. Thus several conversion objects can be cascaded in order to implement complex conversion rules.

Number to Bits:

Comparing to the preceding converter this one is more specialized: it takes exactly one number as input, sends the bit pattern of its lower 8 bits to its own outputs and outputs the resulting number (right-shifted by 8 bits) on the overflow. So with cascaded sets of converter whole numbers of 8..32 bit width can be converted to their related bit patterns.

Numeric IN0 – receives the number where the lower 8 bits have to be converted

Digital IN0 .. digital IN7 – the converted bit pattern

Overflow – the 8 bits right-shifted rest of the input number that can be forwarded to the next converter

Number to Characters:

With this flow object a conversion from numbers to texts is done that internally is performed either as real conversion or as comparison. Dependent on the input value specific digital output values are converted to

- a text that displays the number
- a text that is created using a format string
- a static, predefined text that is assigned to an output dependent on a specific input value.

The format string that is used here can consist of a text that can be chosen freely and some place holders. The input numbers are put into that text on all positions, where these place holders are set. They consist of a "%" sign, a character "d" for whole numbers or "f" for floating point numbers and an additional number in range 1..7 that specifies which input value has to be used. So whenever there is a place holder "%d1" in the text, it will be replaced by the number at input 1, a place holder "%f4" would be replaced by the number at input 4 in floating point representation.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – inputs for the numbers that have to be converted

Digital OUT0 / OVL – the clock overflow signal is emitted whenever an overflow condition appeared during the conversion for outputs, that react on the external clock. This overflow clock signal e.g. can be transferred to the CLK input of the next "Number to Characters" converter.

Char OUT1 .. char OUT7 – the output signals that are emitted dependent on the input values and the configuration of this object

Overflow – the numeric overflow output; when the conversion operation did not match the rules defined within the configuration of the flow element so that no text output could be found that belongs to the number at an input, this number is sent to the overflow output. Together with the OUT0 / OVL signal both can be used for a next conversion step to check some more conversion conditions. Thus several conversion objects can be cascaded in order to implement complex conversion rules.

Number to CMD/Value-Pair:

A CMD/Value pair is a combination of a value and a command name that is assigned to this value. Such pairs can be used to connect to external applications or to identify a value for other purposes. Using this element numerical data can be converted to such pairs. Within the configuration of the flow element a command name can be assigned for every input. The values that are received at this input will be combined with that name to the pair of command and value.

Numeric IN0 - Numeric IN7 - the inputs for the values that have to be combined with the command name

Char OUT0 (CMD) – the output for the command, whenever an value is set to an input an command is emitted here as specified within the configuration

Numeric OUT1 - the output for the value that is assigned to the command, whenever an value is set to an input an command is emitted together with this value

CMD/Value-Pair to Numeric:

This element performs the conversion back from command/value pairs to plain values. Within the configuration of the flow element a name can be assigned for every input. When a command is sent into this flow element that is identical to this name the value is emitted at the related output

Char IN0 (CMD) – the input for the command, here the command name that is assigned to the value is expected

Numeric IN1 – the input for the value that is assigned to the command

Numeric OUT0 - Digital OUT7 – the outputs for the values that could be identified by the configured command; when the string that is set to Char IN0 could not be found within the configuration of the flow element nothing is output here

Characters to Digital:

With this flow object a conversion from text to digital signals is done. Internally this is performed as a comparison operation. Dependent on the input text specific digital output values are created (LOW, HIGH or pulse).

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Char IN1 .. char IN7 – inputs for the texts that have to be converted

Digital OUT0 / OVL – the clock overflow signal is emitted whenever an overflow condition appeared during the conversion for outputs, that react on the external clock. This overflow clock signal e.g. can be transferred to the CLK input of the next “Characters to Digital” converter

Digital OUT1 .. digital OUT7 – the output signals that are emitted dependent on the input values and the configuration of this object

Overflow – the text overflow output; when the conversion operation did not match the rules defined within the configuration of the flow element so that no digital output signal could be found that belongs to the text at an input, this number is sent to the overflow output. Together with the OUT0 / OVL signal both can be used for a next conversion step to check some more conversion conditions. Thus several conversion objects can be cascaded in order to implement complex conversion rules.

Characters to Number:

With this flow object a conversion from text to digital signals is done that internally is performed as comparison or – in case the related configuration option is set by the user – as conversion where the software tries to parse the text and to interpret it as number. Dependent on the input value specific digital output values are created (LOW, HIGH or pulse).

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Char IN1 .. char IN7 – inputs for the texts that have to be converted

Digital OUT0 / OVL – the clock overflow signal is emitted whenever an overflow condition appeared during the conversion for outputs, that react on the external clock. This overflow clock signal e.g. can be transferred to the CLK input of the next “Characters to Number” converter

Numeric OUT1 .. numeric OUT7 – the output signals that are emitted dependent on the input values and the configuration of this object

Overflow – the text overflow output; when the conversion operation did not match the rules defined within the configuration of the flow element so that no digital output signal could be found that belongs to the text at an input, this number is sent to the overflow output. Together with the OUT0 / OVL signal both can be used for a next conversion step to check some more conversion conditions. Thus several conversion objects can be cascaded in order to implement complex conversion rules.

Characters to CMD/Value-Pair:

A CMD/Value pair is a combination of a value and a command name that is assigned to this value. Such pairs can be used to connect to external applications or to identify a value for other purposes. Using this element digital data can be converted to such pairs. Within the configuration of the flow element a command name can be assigned for every input. The values that are received at this input will be combined with that name to the pair of command and value.

Char IN0 - Char IN7 – the inputs for the values that have to be combined with the command name

Char OUT0 (CMD) – the output for the command, whenever an value is set to an input an command is emitted here as specified within the configuration

Char OUT1 - the output for the value that is assigned to the command, whenever an value is set to an input an command is emitted together with this value

Binary to CMD/Value-Pair:

With this element digital data can be converted to CMD/Value pairs. Within the configuration of the flow element a name can be assigned for every input. The values that are received at this input will be combined with that commands name to the pair of command and value.

Binary IN0 - Binary IN7 - the inputs for the values that have to be combined with the command name

Char OUT0 (CMD) – the output for the command, whenever an value is set to an input an command is emitted here as specified within the configuration

Binary OUT1 - the output for the value that is assigned to the command, whenever an value is set to an input an command is emitted together with this value

CMD/Value-Pair to Binary:

This element performs the conversion back from Command/Value pairs to plain binary data blocks. Within the configuration of the flow element a name can be assigned for every input. When a command is sent into this flow element that is identical to this name the value is emitted at the related output

Char IN0 (CMD) – the input for the command, here the command name that is assigned to the value is expected

Binary IN1 - the input for the value that is assigned to the command

Binary OUT0 - Binary OUT7 - the outputs for the values that could be identified by the configured command; when the string that is set to Char IN0 could not be found within the configuration of the flow element nothing is output here

CMD/Value-Pair to Characters:

This element performs the conversion back from Command/Value pairs to plain values. Within the configuration of the flow element a name can be assigned for every input. When a command is sent into this flow element that is identical to this name the value is emitted at the related output

Char IN0 (CMD) – the input for the command, here the command name that is assigned to the value is expected

Char IN1 - the input for the value that is assigned to the command

Char OUT0 - Char OUT7 - the outputs for the values that could be identified by the configured command; when the string that is set to Char IN0 could not be found within the configuration of the flow element nothing is output here

Mixed to Characters:

This flow object performs a format string based conversion of numbers and characters. Here the format string consists of a text that can be chosen freely and that contains place holders for the input values. These place holders start with a “%” followed by a format character “d” for whole numbers, “f” for numbers in floating point representation and “s” for a text. As last part of the place holder the input number has to be added. So for this format string allowed place holders are “%d1” .. “%d4”, “%f1” .. “%f4” and “%s5” .. “%s7”.

Digital IN0 (CLK) – the clock signal that is used to start the conversion for the current input values (in case this option is enabled within the settings)

Numeric IN1 .. numeric IN4 – the inputs for the numeric part that can be used for conversion

Char IN5 .. char IN7 – the inputs for the texts that can be used for conversion

Char OUT1 – the text that was converted according the rules of this element

5.3.3.7.2.2 External Data Conversion Flow Objects

BeamConstruct to Control:

This plug-in converts BeamConstruct laser marking project files to a stream of binary control data structures. Comparing to the plug-ins that convert static CSV or HPGL vector data this one supports all the dynamic and lasermarking-specific features that BeamConstruct offers. This includes setting of laser and scanner parameters, counting of serial numbers, setting date/time information for every new process cycle, manipulating the contents of a project via the inputs of the plug-in, synchronising different motion information, waiting for external trigger signals, setting outputs of the scanner controller card and others more.

Before using this plug-in following parameter has to be set within the configuration panel:

Default Filename – path to a predefined BEAMP-file that has to be loaded and converted; this project file is loaded at startup of the application and can be replaced dynamically via character input IN1

This conversion plug-in uses the following in- and outputs:

Digital IN0 (CLK) – start conversion of the loaded BeamConstruct project, as soon as a HIGH value is given here the complete file is converted so that the related output emit a continuous stream of data according to the values found within the file

Char IN1 (FILE) – specifies and loads a new BEAMP file that will be converted when the next HIGH signal is given to IN0

Char IN2 – input for dynamic text data that can modify the currently loaded BEAMP-file in case the related input-element is used within that project

Char IN3 – input for dynamic text data that can modify the currently loaded BEAMP-file in case the related input-element is used within that project

Char IN4 – input for dynamic text data that can modify the currently loaded BEAMP-file in case the related input-element is used within that project

Digital IN7 (BSY) – this is a synchronisation input that has to be connected with the BSY-outputs of all scanner-controller and motion-elements that receive control data from this plug-in

Binary OUT0 (CTRL) – output for binary control data that contain laser marking information according to the values out of the loaded file; here a stream of data is emitted until all geometry of a given file is processed, an exact timing that would result out of the movement speeds and the position is ignored, this is handled by a connected scanner controller plug-in. Comparing to the following two outputs OUT2 and OUT3 this one in every case has to be connected to a scanner controller plug in since OUT0 does not only emit motion information that can be handled by a simple motor controller but the more complex and feature-loaded laser marking information a scanner controller card expects.

Numeric OUT1 (ERR) – this output emits error codes in case the specified BEAMP-file could not be loaded.

Binary OUT2 (CTRL) – here additional motion data are emitted that correspond to motion-elements used within the currently loaded BEAMP-file, the BSY-output of the plug-in that receives these control data has to be connected with the BSY-input of this plug-in

Binary OUT3 (CTRL) – here additional motion data are emitted that correspond to motion-elements used within the currently loaded BEAMP-file, the BSY-output of the plug-in that receives these control data has to be connected with the BSY-input of this plug-in

This plug-in is located in flowplugins/libio_beamp2ctrl

Characters to Mixed:

This flow-plug-in parses an input string and extracts contained numeric values or sub-strings out of it based on an format string string. This format string describes the structure of the input data. At positions where a number or a string is expected that has to be extracted and submitted via the outputs of the plug-in a place-holder has to be set. Allowed place-holders are “%f1”, “%f2”, “%f3” and “%f4” for numeric values that have to be emitted at outputs 1..4 and “%s5”, “%s6” and “%s7” for character values that have to be emitted at outputs 5..7. As an example: a format string “speed=%f3” would be able to parse an input data string “speed=135”, to extract the number “135” out of it and to emit it at output 3 of the plug-in. Please note: beside the place-holders described above only the escape-token “%%” for a single “%”-sign within the input string is allowed to be used. All other combinations of “%” followed by characters or numbers are strictly

forbidden! Any use of them could case an undefined behaviour of the application.

This plug-in offers following IO's:

Digital IN0 (CLK) – the clock signal that is used to start the conversion for the current input values (used only in case this option is enabled within the elements properties dialogue)

Char IN1 (DATA) – input data, here the string is expected that is formatted in a way like it is described by the format string given within the plug-ins configuration panel

Numeric OUT1 – output for values extracted out of the input string at positions with the place-holder “%f1”

Numeric OUT2 – output for values extracted out of the input string at positions with the place-holder “%f2”

Numeric OUT3 – output for values extracted out of the input string at positions with the place-holder “%f3”

Numeric OUT4 – output for values extracted out of the input string at positions with the place-holder “%f4”

Char OUT5 – output for values extracted out of the input string at positions with the place-holder “%s5”

Char OUT6 – output for values extracted out of the input string at positions with the place-holder “%s6”

Char OUT7 – output for values extracted out of the input string at positions with the place-holder “%s7”

This plug-in is located in flowplugins/char2mixed

Control To Number:

This plug-in accepts control data as input and converts them to numerical output values. It simulates the operation that would be done by e.g. a real motion plug-in that controls external drives. Thus the generated numerical values not only correspond to the position such a drive would be moved to but also to the speed that movement would be performed.
Within the configuration of this plug-in only one option is available that gives the possibility to exchange the coordinates of X and Y data. This would affect only coordinate-related control data, movement-related control data (which define the axis names A, B and C) are not influenced.

Following in- and outputs are supported:

Binary IN7 (CTRL) – input control data, here coordinate-related control data (including tool handling information) as well as plain motion-related control data are accepted

Numeric OUT0 (POS 1) – time-synchronous position information of axis 1 (X or A depending on the type of input control data) according to the control data set at IN7

Numeric OUT1 (MODE 1) – current state of axis 1, a value of 0 means the axis has stopped, values greater than 0 specify the current motion speed in unit mm/sec

Numeric OUT2 (POS 2) – time-synchronous position information of axis 2 (Y or B depending on the type of input control data) according to the control data set at IN7

Numeric OUT3 (MODE 2) – current state of axis 2, a value of 0 means the axis has stopped, values greater than 0 specify the current motion speed in unit mm/sec

Numeric OUT4 (POS 3) – time-synchronous position information of axis 3 (Z or C depending on the type of input control data) according to the control data set at IN7

Numeric OUT5 (MODE 3) – current state of axis 3, a value of 0 means the axis has stopped, values greater than 0 specify the current motion speed in unit mm/sec

Digital OUT6 (BSY) – signalises the busy-state of the plug-in as long as motion data are processed for at least one axis or as long as there are motion data available for processing; this output has to be connected with the BSY-input of the plug-in that generates the control data

Binary OUT7 (CTRL) – this output emits the tool-controlling part of coordinate-related control information that have been set at IN7, these data are emitted time-synchronous to the data at the numerical outputs

This plug-in is located in flowplugins/libio_ctrl2num

Control To Tool:

This plug-in uses a binary control structure as input and returns the single tool-related fields out of it as outputs. Thus it can be used to access a tool directly by sending the related parameters to it using separate communication channels that do not handle the ControlRoom-internal control structure. This converter expects binary control data that are not delivered by a converter (like the CSV or HPGL converter directly) but from a motion plug-in that emits the control data with correct timing (like the MDrive+ plug-in). The Control To Tool plug-in handles the on delays and off delays correctly in case they are positive, means the data emitted by this plug-in already have the correct timing according to the related movement.

Within the configuration panel of this plug-in it has to be configured which parameters have to be emitted at outputs 3..7. Default values for tool on delay and off delay have to be set, they are used as as long as the related delays are not available within the incoming binary control structure.

Binary IN0 (CTRL) – motion-synchronous control data that have to be converted

Digital OUT0 (ON) – state of the tool itself, HIGH signals it is turned on, Low it has to be turned off; when this signal changes its timing is corrected using the current on delay and off delay values (in case they are greater than 0 and therefore not handled within the preceding motion element).

Numeric OUT1 (PWR) – the power in range 0%..100% the tool has to be used with

Numeric OUT2 (FREQ) – the frequency in unit Hz the tool has to be used with

Numeric OUT3 (PRM) – here a new value is emitted whenever the custom parameter changes that was configured for this output within the settings panel of the plug-in

Numeric OUT4 (PRM) – here a new value is emitted whenever the custom parameter changes that was configured for this output within the settings panel of the plug-in

Numeric OUT5 (PRM) – here a new value is emitted whenever the custom parameter changes that was configured for this output within the settings panel of the plug-in

Numeric OUT6 (PRM) – here a new value is emitted whenever the custom parameter changes that was configured for this output within the settings panel of the plug-in

Numeric OUT7 (PRM) – here a new value is emitted whenever the custom parameter changes that was configured for this output within the settings panel of the plug-in

This plug-in is located in flowplugins/libio_ctrl2tooldata

CSV to Characters:

This plug-in is able to split single lines of CSV data, to extract the data fields out of them and to emit the contained values at its outputs. Here all data are threatened as characters, means also numbers are output as texts. Such numbers in textual representation have to be converted to real numeric data types explicitly, here e.g. the “Character To Number” conversion element can be used.
Whenever a field within a CSV line does not contain data, the related output does emit nothing, no empty string is sent in this case.

Within the configuration panel the CSV separation character has to be specified, here it can be chosen between tabulator, semicolon, comma, space and colon. Please note: the use of a comma to separate CSV fields excludes the use of floating point number data fields that use a comma too.
Beside of that for every output a number has to be specified that corresponds to the field number of the CSV line that has to be emitted at this output.

Char IN0 (CSV) – input for the CSV data, whenever a new line is set it is converted and split automatically, the fields are sent at the related outputs as configured

Char OUT0 .. Char OUT7 – output for a CSV data field as configured within the plug-in settings panel

This plug-in is located in flowplugins/libio_csv2char

CSV to Number:

This plug-in is able to split single lines of CSV data, to extract the data fields out of, convert it to floating point numbers and to emit them at its outputs. When a CSV field does not contain a valid number the assigned output is not set.

Within the configuration panel the CSV separation character has to be specified, here it can be chosen between tabulator, semicolon, comma, space and colon. Please note: the use of a comma to separate CSV fields excludes the use of floating point number data fields that use a comma too.
Beside of that for every output a number has to be specified that corresponds to the field number of the CSV line that has to be emitted at this output.

Char IN0 (CSV) – input for the CSV data, whenever a new line is set it is converted and split automatically, the fields are sent at the related outputs as configured

Number OUT0 .. Number OUT7 – output for a CSV data field as configured within the plug-in settings panel

This plug-in is located in flowplugins/libio_csv2number

(De)Compress data:

Using this plug-in binary data blocks can be compressed and decompressed. Here no additional parameters can be configured, the plug-in works fully automatic. The plug-in can compress only binary data that have not been compressed before, when a data block is handed over where already a compression has been applied, it is rejected. Within the debugger that rejection is indicated, within the OpenPlayer no error message appears and the data block is rejected silently.

Binary IN0 - input for binary data that have to be compressed using the ZIP compression algorithm

Binary IN1 - input for binary data that have to be compressed using the BZ2 compression algorithm

Binary IN2 - input for binary data that have to be compressed using the GZ compression algorithm

Binary IN7 (INFLATE) - input for data compressed using ZIP or BZ2

Binary OUT0 (ZIP) - here the ZIP-compressed data are given as soon as the compression has been finished, this method works asynchronously and requires some time dependent on the size and complexity of the input data

Binary OUT1 (BZ2) - here the BZ2-compressed data are given as soon as the compression has been finished, this method works asynchronously and requires some time dependent on the size and complexity of the input data

Binary OUT2 (GZ) - here the GZ-compressed data are given as soon as the compression has been finished, this method works asynchronously and requires some time dependent on the size and complexity of the input data

Binary OUT7 (INFLATE) - here the decompressed data are given as soon as the decompression has been finished, this method works asynchronously and requires some time dependent on the size and complexity of the input data

This plug-in is located in flowplugins/libio_compress

5.3.3.7.2.3 Logic Operation Flow Objects

Digital NOT:

This operation does not require any further definitions, it inverts the digital value that is set to the input.

Digital IN0 .. digital IN7 – the input for the values that have to be inverted

Digital OUT0 .. digital OUT7 – the inverted values from the related inputs

Digital NOP:

This operation does not require any further definitions. It does not modify the digital value that is set to the input but simply returns it on next processing cycle (NOP = **n**o **op**eration). It can be used e.g. for testing specific functionalities, to synchronize the order of data flows or to let data flows arrive after each other.

Digital IN0 .. digital IN7 – the input for the values that have to be inverted

Digital OUT0 .. digital OUT7 – the inverted values from the related inputs

Digital (N)OR:

Using this flow control several inputs can be OR-concatenated and the result can be inverted optionally. Within the configuration dialogue the inputs that belong to one output and where the values have to be concatenated can be configured.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears. If at least one output uses this input as clock, no other outputs can be configured to use IN0 as data input.

Digital IN1 .. digital IN7 – data inputs for the OR-concatenation

Digital OUT0 .. digital OUT7 – data outputs that send the values resulting from this operation

Digital (NOT) XOR:

Using this flow control several inputs can be Exclusive-OR-concatenated and the result can be inverted optionally. Within the configuration dialogue the inputs that belong to one output and from which the values have to be concatenated can be configured.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears. If at least one output uses this input as clock, no other outputs can be configured to use IN0 as data input.

Digital IN1 .. digital IN7 – data inputs for the XOR-concatenation

Digital OUT0 .. digital OUT7 – data outputs that send the values resulting from this operation

Digital (N)AND:

Using this flow control several inputs can be AND-concatenated and the result can be inverted optionally. Within the configuration dialogue the inputs that belong to one output and from which the values have to be concatenated can be configured.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears. If at least one output uses this input as clock, no other outputs can be configured to use IN0 as data input.

Digital IN1 .. digital IN7 – data inputs for the AND-concatenation

Digital OUT0 .. digital OUT7 – data outputs that send the values resulting from this operation

Digital Shift Register:

This flow elements implements a shifting register: it stores all data that are set at its data input. New values are stored so that they can be read at output 0 during the next read operation. Whenever a new value is set all the preceding ones are shifted up by one output number so that the old value from output 0 is now located at output 1, the old one from 1 is located at output 2 and so on. The old value from output 7 will be shifted to the overflow output where it is emitted immediately. This operation does not require any further parameters or definitions.

Digital IN0 (OUT) – when this input receives a HIGH signal all the outputs from 0..7 emit the data that are currently stored.

Digital IN1 (RES) – using this input all the stored data can be reset to their default value LOW

Digital IN2 (DATA) – every time new data arrive here the existing ones within the flow element are shifted up by one output position and the new one is stored to be fetched at output 0 at the next OUT-signal

Digital OUT0..digital OUT7 – on an HIGH-signal on digital IN0 here the currently stored output data are emitted

Position Correction to Number:

This plug-in separates the contents of a position correction binary structure and outputs it as plain numbers:

Binary IN7 (POSCORR) – position correction structure to be converted by this plug-in

Numeric OUT0 (X) – translation in X direction (mm)

Numeric OUT1 (Y) – translation in Y direction (mm)

Numeric OUT2 (Z) – translation in Z direction (mm)

Numeric OUT3 (XANG) – rotation around X axis (unit degrees)

Numeric OUT4 (YANG) – rotation around Y axis (unit degrees)

Numeric OUT5 (ZANG) – rotation around Z axis (unit degrees)

SR-FlipFlop:

A flip-flop is an element that can have two logical, stable states. This special kind of a flip-flop can set or reset that state, when the flip-flop is set its output Q is set to HIGH, elsewhere it is set to low. This flow element implements four flip-flops that provide the same functionality but act completely independent from each other.

Digital IN0 (S) – input to set the first flip-flop

Digital IN1 (R) – input to reset the first flip-flop

Digital IN2 (S) – input to set the second flip-flop

Digital IN3 (R) – input to reset the second flip-flop

Digital IN4 (S) – input to set the third flip-flop

Digital IN5 (R) – input to reset the third flip-flop

Digital IN6 (S) – input to set the fourth flip-flop

Digital IN7 (R) – input to reset the fourth flip-flop

Digital OUT0 (Q) – data output of the first flip-flop, it is set to HIGH when it is set (S)

Digital OUT1 (!Q) – inverted data output of the first flip-flop, it is set to HIGH when it is not set (R)

Digital OUT2 (Q) – data output of the second flip-flop, it is set to HIGH when it is set (S)

Digital OUT3 (!Q) – inverted data output of the second flip-flop, it is set to HIGH when it is not set (R)

Digital OUT4 (Q) – data output of the third flip-flop, it is set to HIGH when it is set (S)

Digital OUT5 (!Q) – inverted data output of the third flip-flop, it is set to HIGH when it is not set (R)

Digital OUT6 (Q) – data output of the fourth flip-flop, it is set to HIGH when it is set (S)

Digital OUT7 (!Q) – inverted data output of the fourth flip-flop, it is set to HIGH when it is not set (R)

Toggle-FlipFlop:

In its initial state the Q output is HIGH and the inverted output is LOW. Whenever a HIGH signal is set at the input of this FlipFlop both outputs change their state. So with the first HIGH signal Q is set to LOW and the inverted output to HIGH, with the next HIGH input signal they switch back to the original state.
This flow element contains four FlipFlops that act completely independent from each other.

Digital IN0 (T) – toggle input for the first FlipFlop

Digital IN2 (T) – toggle input for the second FlipFlop

Digital IN4 (T) – toggle input for the third FlipFlop

Digital IN6 (T) – toggle input for the fourth FlipFlop

Digital OUT0 (Q) – data output of the first FlipFlop

Digital OUT1 (!Q) – inverted data output of the first FlipFlop

Digital OUT2 (Q) – data output of the second FlipFlop

Digital OUT3 (!Q) – inverted data output of the second FlipFlop

Digital OUT4 (Q) – data output of the third FlipFlop

Digital OUT5 (!Q) – inverted data output of the third FlipFlop

Digital OUT6 (Q) – data output of the fourth FlipFlop

Digital OUT7 (!Q) – inverted data output of the fourth FlipFlop

Numerical NOT:

This operation does not require any further definitions, it treats the numerical value that is set to the input as 32 bit signed, whole numbered value and inverts it on its binary level.

Numeric IN0 .. numeric IN7 – the input for the values that have to be inverted

Numeric OUT0 .. numeric OUT7 – the inverted values from the related inputs

Numerical NOP:

This operation does not require any further definitions. It does not modify the numerical value that is set to the input but simply returns it on next processing cycle (NOP = **n**o **op**eration). It can be used e.g. for testing specific functionalities, to synchronize the order of data flows or to let data flows arrive after each other.

Numeric IN0 .. numeric IN7 – the input for the values that have to be inverted

Numeric OUT0 .. numeric OUT7 – the inverted values from the related inputs

Numerical (N)OR:

Using this flow control several inputs can be OR-concatenated and the result can be inverted optionally. Within the configuration dialogue the inputs that belong to one output and where the values have to be concatenated can be configured. The input numbers are treated as 32 bit signed, whole numbered values for concatenation, so possibly existing decimal places will get lost.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the OR-concatenation

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Numerical (NOT) XOR:

Using this flow control several inputs can be Exclusive-OR-concatenated and the result can be inverted optionally. Within the configuration dialogue the inputs that belong to one output and where the values have to be concatenated can be configured. The input numbers are treated as 32 bit signed, whole numbered values for concatenation, so all decimal places will get lost when this operation is performed.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the XOR-concatenation

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Numerical (N)AND:

Using this flow control several inputs can be AND-concatenated and the result can be inverted (optionally). Within the configuration dialogue the inputs that belong to one output and where the values have to be

concatenated can be configured. The input numbers are treated as 32 bit signed, whole numbered values for concatenation, so all decimal places will get lost.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the AND-concatenation

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Numeric Shift Register:

This flow elements implements a shifting register: it stores all data that are set at its data input. New values are stored so that they can be read at output 0. Whenever a new value is set all the preceding ones are shifted up by one output number so that the old value from output 0 is now located at output 1, the old one from 1 is located at output 2 and so on. The old value from output 7 will be shifted to the overflow output where it is emitted immediately. This operation does not require any further parameters or definitions.

Digital IN0 (OUT) – when this input receives a HIGH signal all the outputs from 0..7 emit the data that are currently stored.

Digital IN1 (RES) – using this input all the stored data can be reset to their default value 0

Numeric IN2 (DATA) – every time new data arrive here the existing ones within the flow element are shifted up by one output position and the new one is stored to be fetched at output 0 at the next OUT-signal

Numeric OUT0..numeric OUT7 – on an HIGH-signal on digital IN0 here the currently stored output data are emitted

Character NOP:

This operation does not require any further definitions. It does not modify the character value that is set to the input but simply returns it on next processing cycle (NOP = **n**o **op**eration). It can be used e.g. for testing specific functionalities, to synchronize the order of data flows or to let data flows arrive after each other.

Char IN0 .. char IN7 – the input for the values that have to be inverted

Char OUT0 .. char OUT7 – the inverted values from the related inputs

Character Shift Register:

This flow elements implements a shifting register: it stores all data that are set at its data input. New values are stored so that they can be read at output 0. Whenever a new value is set all the preceding ones are shifted up by one output number so that the old value from output 0 is now located at output 1, the old one from 1 is located at output 2 and so on. The old value from output 7 will be shifted to the overflow output where it is emitted immediately. This operation does not require any further definitions.

Digital IN0 (OUT) – when this input receives a HIGH signal all the outputs from 0..7 emit the data that are currently stored.

Digital IN1 (RES) – using this input all the stored data can be reset to their default value (an empty string)

Char IN2 (DATA) – every time new data arrive here the existing ones within the flow element are shifted up by one output position and the new one is stored to be fetched at output 0 at the next OUT-signal

Char OUT0..char OUT7 – on an HIGH-signal on digital IN0 here the currently stored output data are emitted

Binary NOP:

This operation does not require any further definitions. It does not modify the binary value that is set to the input but simply returns it on next processing cycle (NOP = **n**o **op**eration). It can be used e.g. for testing specific functionalities, to synchronize the order of data flows or to let data flows arrive after each other.

Binary IN0 .. binary IN7 – the input for the values that have to be inverted

Binary OUT0 .. binary OUT7 – the inverted values from the related inputs

Binary Shift Register:

This flow elements implements a shifting register: it stores all data that are set at its input. New values are stored so that they can be read at output 0. Whenever a new value is set all the preceding ones are shifted up by one output number so that the old value from output 0 is now located at output 1, the old one from 1 is located at output 2 and so on. The old value from output 7 will be shifted to the overflow output. This operation does not require any further definitions.

Digital IN0 (OUT) – when this input receives a HIGH signal all the outputs from 0..7 emit the data that are currently stored.

Digital IN1 (RES) – using this input all the stored data can be reset to their default value NULL

Binary IN2 (DATA) – every time new data arrive here the existing ones within the flow element are shifted up by one output position and the new one is stored to be fetched at output 0 at the next OUT-signal

Binary OUT0..binary OUT7 – on an HIGH-signal on digital IN0 here the currently stored output data are emitted

5.3.3.7.2.4 Logic Operation Flow Objects

Mutual Exclusion:

The in- and outputs of this plug-in all work together using a common logic which implements a mutual exclusion. At the input a HIGH signal is expected. Independent from the current state of all other inputs only the last input that is set to HIGH sets its opposite output to HIGH too, all other outputs are set to LOW. As an example: when there is set a HIGH signal at Digital IN3 the related output Digital OUT3 is set to HIGH too. Now when Digital IN5 is set to HIGH, digital OUT3 is set to LOW independent from the state of Digital IN3 and Digital OUT5 is set to HIGH instead. More than this: on every HIGH signal that is set on input side all eight outputs emit a value: the one that belongs to the currently set input emits a HIGH and the other seven outputs emit a LOW.

Digital IN0 – Digital IN7 – digital inputs for the mutual exclusion

Digital OUT0 – Digital OUT7 – the related outputs where exactly one of them can be HIGH, all other will be LOW

This plug-in is located in flowplugins/libio_digimutex.

Rising/Falling Edge:

This plug-in detects if there is a rising or falling edge in digital or numerical signals. The result is given at the output:

Digital IN0 – this input can be used to detect a digital rising edge (a change from LOW to HIGH), the result is given at Outputs OUT0 and OUT1

Digital IN2 – here a digital falling edge can be detected (a change from HIGH to LOW), as soon as such an edge is detected the result is emitted at OUT2 and OUT3

Numeric IN4 – this input can be used to detect a numeric rising edge (which is the case when the second of two given values is bigger than the first one), the result is given at Outputs OUT4 and OUT5

Numeric IN6 – this input can be used to detect a numeric falling edge (which is the case when the second of two given values is smaller than the first one), the result is given at Outputs OUT6 and OUT7

Digital OUT0 (0) – when there is detected a rising edge at IN0 this output emits a LOW signal

Digital OUT1 (1) – when there is detected a rising edge at IN0 this output emits a HIGH signal

Digital OUT2 (0) – this output emits a LOW signal when there is a falling edge at IN2 detected

Digital OUT3 (1) – this output emits a HIGH signal when there is a falling edge at IN2 detected

Digital OUT4 (0) – when there is detected a rising edge at IN4 this output emits a LOW signal

Digital OUT5 (1) – when there is detected a rising edge at IN4 this output emits a HIGH signal

Digital OUT6 (0) – this output emits a LOW signal when there is a falling edge at IN6 detected

Digital OUT7 (1) – this output emits a HIGH signal when there is a falling edge at IN6 detected

This plug-in is located in flowplugins/libio_edge.

5.3.3.7.2.5 Mathematical Flow Objects

Compare Digital:

This is a comparison function that checks if one of two digital input values is smaller, equal or greater than the second one, for digital data a HIGH value is always greater than a LOW value. Here one flow element contains two separate compare functions that act completely independent from each other. Beside the name no other parameters or values can be configured within the properties dialogue of this element. The comparison result is emitted every time a new input value is set.

Digital IN0 (A) – input of the first value of the first comparator that has to be compared

Digital IN1 (B) – input of the second value of the first comparator that has to be compared

Digital IN3 (A) – input of the first value of the second comparator that has to be compared

Digital IN4 (B) – input of the second value of the second comparator that has to be compared

Digital OUT0 (>) - this output is set to HIGH when value A of the first comparator is greater than value B

Digital OUT1 (=) - this output is set to HIGH when value A of the first comparator is equal to value B

Digital OUT2 (<) - this output is set to HIGH when value A of the first comparator is smaller than value B

Digital OUT3 (>) - this output is set to HIGH when value A of the second comparator is greater than value B

Digital OUT4 (=) - this output is set to HIGH when value A of the second comparator is equal to value B

Digital OUT5 (<) - this output is set to HIGH when value A of the second comparator is smaller than value B

Digital Counter:

This flow element can be used to count digital events.

Digital IN0 (RES) – reset the state of all counters back to 0

Digital IN1 (1) – count all digital data with HIGH state

Digital IN2 (0) – count all digital data with LOW state

Digital IN3 – count all digital data independent from their state

Numeric OUT1 – the number of counted HIGH states

Numeric OUT2 – the number of counted LOW states

Numeric OUT3 – the number of counted states

Numerical Addition:

Using this flow control the values of several inputs and/or a constant value can be added with each other. Within the configuration dialogue the inputs that belong to one output (and where the values have to be added) can be configured. The input values are treated as floating point numbers here.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the addition

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Numerical Subtraction:

Using this flow control the values of several inputs and/or a constant value can be subtracted from each other. Within the configuration dialogue the inputs that belong to one output (and where the values have to be subtracted) can be configured. The input values are treated as floating point numbers, subtraction is done from left to right, starting with the lowest input number. The configured constant is used for subtraction as last value.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the subtraction

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Numerical Multiplication:

Using this flow control the values of several inputs and/or a constant value can be multiplied with each other. Within the configuration dialogue the inputs that belong to one output (and where the values have to be multiplied) can be configured. The input values are treated as floating point numbers.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the multiplication

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Numerical Division:

Using this flow control the values of several inputs and or a constant value can be divided by each other. Within the configuration dialogue the inputs that belong to one output (and where the values have to be divided from) can be configured. The input values are treated as floating point numbers. Division is done from left to right, starting with the lowest input number. The configured constant is used for division as last value.

Digital IN0 / CLK – the clock signal that is used for conversion for these input/output mappings that are configured to act only when this external trigger appears

Numeric IN1 .. numeric IN7 – data inputs for the division

Numeric OUT1 .. numeric OUT7 – data outputs that send the values resulting from this operation

Compare Numbers:

This is a comparison function that checks if one of two numerical input values is smaller, equal or greater than the second one. Here one flow element contains two separate compare functions that act completely independent from each other. Beside the name of the flow element no other parameters or values can be configured within its properties dialogue. The comparison result is emitted every time a new input value is set.

Numeric IN0 (A) – input of the first value of the first comparator that has to be compared

Numeric IN1 (B) – input of the second value of the first comparator that has to be compared

Numeric IN3 (A) – input of the first value of the second comparator that has to be compared

Numeric IN4 (B) – input of the second value of the second comparator that has to be compared

Digital OUT0 (>) - this output is set to HIGH when value A of the first comparator is greater than value B

Digital OUT1 (=) - this output is set to HIGH when value A of the first comparator is equal to value B

Digital OUT2 (<) - this output is set to HIGH when value A of the first comparator is smaller than value B

Digital OUT3 (>) - this output is set to HIGH when value A of the second comparator is greater than value B

Digital OUT4 (=) - this output is set to HIGH when value A of the second comparator is equal to value B

Digital OUT5 (<) - this output is set to HIGH when value A of the second comparator is smaller than value B

Numerical Counter:

This flow element can be used to count numerical data events.

Digital IN0 (RES) – reset the state of all counters back to 0

Numeric IN1 (<>0) – count all numbers that are not 0

Numeric IN2 (0) – count all numbers that are equal to 0

Numeric IN3 – count all numerical data independent from their value

Numeric OUT1 – the number of counted values that are not equal to 0

Numeric OUT2 – the number of counted values that are equal to 0

Numeric OUT3 – the number of counted values

Compare Characters:

This is a comparison function that performs a case-sensitive check if one of two text input values is smaller, equal or greater than the second one. A character value is greater than the other one when they are not identical and when the first (ASCII-)character that does not match has a greater value. Here one flow element contains two separate compare functions that act completely independent from each other. Beside the name of the flow element no other parameters or values can be configured within its properties dialogue. The comparison result is emitted every time a new input value is set.

Char IN0 (A) – input of the first value of the first comparator that has to be compared

Char IN1 (B) – input of the second value of the first comparator that has to be compared

Char IN3 (A) – input of the first value of the second comparator that has to be compared

Char IN4 (B) – input of the second value of the second comparator that has to be compared

Digital OUT0 (>) - this output is set to HIGH when value A of the first comparator is greater than value B

Digital OUT1 (=) - this output is set to HIGH when value A of the first comparator is identical to value B

Digital OUT2 (<) - this output is set to HIGH when value A of the first comparator is smaller than value B

Digital OUT3 (>) - this output is set to HIGH when value A of the second comparator is greater than value B

Digital OUT4 (=) - this output is set to HIGH when value A of the second comparator is identical to value B

Digital OUT5 (<) - this output is set to HIGH when value A of the second comparator is smaller than value B

Character Counter:

This flow element can be used to count character data events.

Digital IN0 (RES) – reset the state of all counters back to 0

Char IN1 (<>' ') – count all texts that are not empty

Char IN2 (=' ') – count all texts with a length of 0 (empty texts)

Char IN3 – count all character data independent from their value

Numeric OUT1 – the number of counted values that have not been empty

Numeric OUT2 – the number of counted values that have been empty

Numeric OUT3 – the number of counted values

Binary Data Counter:

This flow element can be used to count binary data.

Digital IN0 (RES) – reset the state of all counters back to 0

Binary IN1 – count all binary data independent from their type, subtype, contents or compression state

Numeric OUT1 – the number of counted data blocks

5.3.3.7.2.6 External Mathematical Flow Objects

Averaging:

Using this plug-in a defined amount of samples can be collected. As soon as this number of samples was received by the plug-in it calculates the average value out of them and emits this value. So using this plug-in averaging can be done over a set of input samples. This is useful e.g. in cases where a noise with a higher frequency than the payload data has to be removed.
Within the configuration panel it is possible to define a default value for averaging. The number of samples to be collected specified there is used as long as no other value is given at the related #-input.

Numeric IN0 (DATA) – input for the samples that have to be collected

Numeric IN1 (#) - the number of samples that have to be collected at IN0 until the average is calculated and set at OUT0

Numeric IN2 (DATA) – input for the samples that have to be collected

Numeric IN3 (#) - the number of samples that have to be collected at IN2 until the average is calculated and set at OUT2

Numeric IN4 (DATA) – input for the samples that have to be collected

Numeric IN5 (#) - the number of samples that have to be collected at IN4 until the average is calculated and set at OUT4

Numeric IN6 (DATA) – input for the samples that have to be collected

Numeric IN7 (#) - the number of samples that have to be collected at IN6 until the average is calculated and set at OUT6

Numeric OUT0 (AVG) – the average of the last amount of samples given at IN0

Numeric OUT2 (AVG) – the average of the last amount of samples given at IN2

Numeric OUT4 (AVG) – the average of the last amount of samples given at IN4

Numeric OUT6 (AVG) – the average of the last amount of samples given at IN6

This plug-in is located in flowplugins/libio_averaging

Math Calculation 1:

This plug-in offers a set of mathematical operations that require exactly one operand. This operand is set at an numeric input, the result is given at the related output immediately. No additional configuration is necessary for this plug-in

Numeric IN0 (SIN) - input operand for calculation of sinus

Numeric IN1 (COS) - input operand for calculation of cosine

Numeric IN2 (ln) - input operand for calculation of e (logarithmus naturalis)

Numeric IN3 (lg) - input operand for calculation of logarithm

Numeric OUT0 (SIN) - result of sinus calculation

Numeric OUT1 (COS) - result of cosine calculation

Numeric OUT2 (ln) - result of e (logarithmus naturalis) calculation

Numeric OUT3 (lg) - result of logarithm calculation

This plug-in is located in flowplugins/libio_math1

Override Control Parameters:

This plug-in offers an easy way to override/modify some of the binary control data values globally. It is able to influence, power, frequency, speeds and delays. They can be changed by a factor that can be set as a default value within the plug-in settings and as dynamic values that can be specified dynamically via the related inputs.

Binary IN0 (CTRL) – the binary control data that have to be changed by this plug-in

Numeric IN1 (PWR) – factor (in unit percent) to change the power value with

Numeric IN2 (FREQ) – factor (in unit percent) to change the frequency value with

Numeric IN3 (SPD) – factor (in unit percent) to change all speed values with

Numeric IN4 (DEL) – factor (in unit percent) to change all delay values with

Binary OUT0 (CTRL) – the modified binary control data

PID Controller:

A PID-Controller can be used to set up a complete control loop, it is able to influence its parameters and to keep the controlled parameter.

Within the configuration panel all necessary values and parameters can be set up, independent from that they also can be changed dynamically using the related inputs. Following parameters can be defined:

- Default Reference Value – the reference value this controller has to reach as exact as possible
- Default P – the P-portion of the controllers calculation algorithm
- Default I – the I-portion of the controllers calculation algorithm
- Default D – the D-portion of the controllers calculation algorithm
- Minimum Output – the minimum value the resulting output may have
- Maximum Output – the maximum value the resulting output is allowed to have

Numeric IN0 (SET) – the actual input value that has to be compared with the reference value

Numeric IN1 (REF) – overwrites the default reference value

Numeric IN2 (P) – overwrites the controllers pre-configure P value

Numeric IN3 (I) – overwrites the controllers pre-configure I value

Numeric IN4 (D) – overwrites the controllers pre-configure D value

Numeric OUT0 – the resulting output value that has to be set in order to reach the desired reference value

This plug-in is located in flowplugins/libio_pid_controller.

Rotate Control Parameters:

This plug-in influences the geometry-part of binary control data, it rotates them by defined angles around X, Y or Z axis using a defined 3D coordinate point. The related values can be defined as static default within the plug-ins configuration panel or they can be set dynamically via the inputs:

Binary IN0 (CTRL) – the binary control data that have to be changed by this plug-in

Numeric IN1 (CX) – X-value of the coordinate to be used as rotation centre

Numeric IN2 (CY) – Y-value of the coordinate to be used as rotation centre

Numeric IN3 (CZ) – Z-value of the coordinate to be used as rotation centre

Numeric IN4 (AX) – angle (in unit degrees) to rotate the geometry data around the X axis

Numeric IN5 (AY) – angle (in unit degrees) to rotate the geometry data around the X axis

Numeric IN6 (AZ) – angle (in unit degrees) to rotate the geometry data around the X axis

Binary OUT0 (CTRL) – the modified binary control data

Scale/Translate Control Parameters:

Using this plug-in the geometry-part of binary control data can be translated and scaled. The related scaling and translation parameters can be defined as static default values within the plug-ins configuration panel or they can be set dynamically via the inputs:

Binary IN0 (CTRL) – the binary control data that have to be changed by this plug-in

Numeric IN1 (SCX) – scaling factor in X direction that resizes the geometry data according to the coordinate origin

Numeric IN2 (SCY) – scaling factor in Y direction that resizes the geometry data according to the coordinate origin

Numeric IN3 (SCZ) – scaling factor in Z direction that resizes the geometry data according to the coordinate origin

Numeric IN4 (TX) – translation value for X coordinate for a relative movement by the specified distance

Numeric IN5 (TY) – translation value for Y coordinate for a relative movement by the specified distance

Numeric IN6 (TZ) – translation value for Z coordinate for a relative movement by the specified distance

Binary OUT0 (CTRL) – the modified binary control data

5.3.3.7.2.7 Flow Control Flow Objects

Initial Flow Start:

This is a flow element that is executed exactly once at the beginning. Whenever a project is loaded and started, all elements of this type are executed and the values that are configured for them are sent out. So because it is a starting point that can't depend from any other condition it does not use any inputs.

Digital OUT1, digital OUT2 – digital constants that are sent

Numeric OUT3, numeric OUT4 – numeric constants that are sent

Char OUT3, char OUT4 – text values that are sent

Timer:

Comparing to the preceding one this is a flow element that is executed periodically. Within its configuration dialogue the values can be configured that are sent repeatedly and the time after which these values have to be sent can be configured.

PLEASE NOTE: comparing to the input poll cycle time of some external plug-ins this timer value does NOT depend on the projects global cycle time, it can have a smaller resolution.

PLEASE ALSO NOTE: Within the OpenDebugger this element may behave not exactly as expected. When the program flow is stopped there or when it is executed in single step mode the time configured here may cause different results than in the OpenPlayer due to the fact that a stopped or single stepped flow has a different timing in general.

Digital OUT1, digital OUT2 – digital values that are sent periodically

Numeric OUT3, numeric OUT4 – numeric values that are sent periodically

Char OUT3, char OUT4 – text values that are sent periodically

Exit Application:

This flow element exits an application: whenever a logic HIGH is received at its only input the whole player application is exited. This element can be used to implement a program-end-condition what is important especially within the OpenPlayer, here without such a condition no other possibility exists to exit the

application controlled. It does not offer any outputs because after this element is used there are no more flows that could be handled in any way.

Digital IN1 – digital input that exits the player application when a 1 is received here

Delay:

While other flow elements process their data as fast as possible and send out the results of their operation to their outputs nearly immediately, this element delays the data transfer. The inputs are mapped to their related outputs directly and within the configuration dialogue the delay for every data flow can be defined.

PLEASE NOTE: Within the OpenDebugger this element may behave not exactly as expected. When the program flow is stopped there or when it is executed in single step mode the time configured here may cause different results than in the OpenPlayer due to the fact that a stopped or single stepped flow has a different timing in general.

Digital IN0, digital IN1 – digital input for data flows that have to be delayed

Numeric IN2, numeric IN3 – numerical input for data flows that have to be delayed

Char IN4, char IN5 – text input for data flows that have to be delayed

Numeric IN7 (DLY) – changes the delay values of all inputs; a value set here overwrites the predefined default delays and can be used for a dynamic behaviour of the element

Digital OUT0, digital OUT1 – digital output for data flows that are delayed

Numeric OUT2, numeric OUT3 – numerical output for data flows that are delayed

Char OUT4, char OUT5 – text output for data flows that are delayed

Digital Triggered Gate:

A triggered gate expects a value at input 0 and emits it at an output, as soon as an digital “1” is received at the related input.

Digi IN0 – the input for the value that has to be emitted for every trigger signal

Digi IN1 – Digi IN7 – the inputs for the trigger signals that cause the emission of the value from input 0 at the related outputs

Digi OUT1 – Digi OUT7 – the outputs for the data that are assigned to the trigger signal inputs

Digital Gate:

This element can be used to enable or interrupt the flow of digital data. Comparing to the triggered gate the digital GATE input does not cause the emission of data that have been stored before, here this input allows incoming data to be output immediately (HIGH to enable the gate) or to be dropped (LOW for a disabled gate). When the GATE-input is set to HIGH that does not cause a data transmission of values that have been set before but only after the next data packets arrive.

Digital IN0 (GATE) – as long as this input is set to LOW all the outputs are disabled, no incoming data are forwarded to them

Digital IN1..digital IN7 – data inputs, the values that are set here are available at the related outputs only in case GATE is set to HIGH

Digital OUT1..digital OUT7 – data outputs that become active only in case the gate is opened

Numeric Triggered Gate:

This triggered gate expects a numerical value at input 0 and emits it at an output, as soon as an digital signal “1” is received at the related input.

Numeric IN0 – the input for the value that has to be emitted for every trigger signal

Digi IN1 – Digi IN7 – the inputs for the trigger signals that cause the emission of the value from input 0 at the related outputs

Numeric OUT1 – Numeric OUT7 – the outputs for the data that are assigned to the trigger signal inputs

Numeric Gate:

This element can be used to enable or interrupt the flow of numerical data. Comparing to the triggered gate the digital input does not cause the emission of data that have been stored before, here the input allows incoming data to be output immediately (HIGH to enable the gate) or to be dropped (LOW for a disabled gate). When the GATE-input is set to HIGH that does not cause a data transmission of values that have been set before but only after the next data packets arrive.

Digital IN0 (GATE) – as long as this input is set to LOW all the outputs are disabled, no incoming data are forwarded to them

Numeric IN1..numeric IN7 – data inputs, the values that are set here are available at the related outputs only in case GATE is set to HIGH

Numeric OUT1..numeric OUT7 – data outputs that become active only in case the gate is opened

Character Triggered Gate:

This triggered gate expects a character value at input 0 and emits it at the opposite output, as soon as an digital signal “1” is received at the related input.

Char IN0 – the input for the value that has to be emitted for every trigger signal

Digi IN1 – Digi IN7 – the inputs for the trigger signals that cause the emission of the value from input 0 at the related outputs

Char OUT1 – Char OUT7 – the outputs for the data that are assigned to the trigger signal inputs

Character Gate:

This element can be used to enable or interrupt the flow of character data. Comparing to the triggered gate the digital input does not cause the emission of data that have been stored before, here the input allows incoming data to be output immediately (HIGH to enable the gate) or to be stopped (LOW for a disabled gate). When the GATE-input is set to HIGH that does not cause a data transmission immediately but only after the next data packets arrive.

Digital IN0 (GATE) – as long as this input is set to LOW all the outputs are disabled, no incoming data are forwarded to them

Char IN1..char IN7 – data inputs, the values that are set here are available at the related outputs only in case GATE is set to HIGH

Char OUT1..char OUT7 – data outputs that become active only in case the gate is opened

Binary Triggered Gate:

This triggered gate expects a binary data block at input 0 and emits it at the opposite output, as soon as an digital signal “1” is received at the related input.

Binary IN0 – the input for the value that has to be emitted for every trigger signal

Digi IN1 – Digi IN7 – the inputs for the trigger signals that cause the emission of the data set at input 0

Binary OUT1 – Binary OUT7 – the outputs for the data that are assigned to the trigger signal inputs

Binary Gate:

This element can be used to enable or interrupt the flow of binary data. Comparing to the triggered gate the digital input does not cause the emission of data that have been stored before, here the input allows

incoming data to be output immediately (HIGH to enable the gate) or to be stopped (LOW for a disabled gate). When the GATE-input is set to HIGH that does not cause a data transmission immediately but only after the next data packets arrive.

Digital IN0 (GATE) – as long as this input is set to LOW all the outputs are disabled, no incoming data are output

Binary IN1..binary IN7 – data inputs, the values that are set here are available at the related outputs only in case GATE is set to HIGH

Binary OUT1..binary OUT7 – data outputs that become active only in case the gate is opened

5.3.3.7.2.8 External Flow Control Flow Objects

Dialogue:

Using this plug-in some user interaction is possible to inform the user about specific events or to ask a question in order to get some feedback. This feedback is given within a dialogue window that pops up above the main window and has to be confirmed. Within the configuration panel the text font and style can be defined which has to be used for the dialogue window. A single plug-in can handle only one message at the same time. So when a new message text is set at one of the inputs while the dialogue window is still open it is rejected and will not be displayed. Within the debugger that rejection is indicated, within the OpenPlayer this message is rejected silently.

Char IN0 - here the text for an error dialogue has to be given, this text is displayed as the error message within the dialogue

Char IN1 - here the text for an warning dialogue has to be given, this text is displayed as the warning message within the dialogue

Char IN2 - here the text for an informational dialogue has to be given, this text is displayed as the information message within the dialogue

Char IN3 - here the text for an question has to be given, this text should form a question that can be answered with "Yes", "No" or "Cancel"

Digital OUT0 - this output sends a HIGH signal as soon as the error message is confirmed by the user

Digital OUT1 - this output sends a HIGH signal as soon as the warning message is confirmed by the user

Digital OUT2 - this output sends a HIGH signal as soon as the information message is confirmed by the user

Numeric OUT3 - when the question was answered by the user the result is emitted here, the returned number informs which of the buttons was pressed: 1 stands for "Yes", 2 for "No" and 3 for "Cancel"

This plug-in is located in flowplugins/libio_dialogue.

Filedialogue:

This is an other plug-in that provides some user interaction, here dialogues for opening a file, saving a file or selecting a directory can be created. Within the configuration panel of the plug-in the font and text style for the dialogues can be chosen and the texts of the title bars can be predefined. Beside of that for the load and save file dialogue a wild card can be defined that provides a filter to the dialogue so that only files with defined extensions are displayed. As long as a file dialogue is open, no other dialogue of the same type can be created. Within the debugger there is a notification in this case, within the OpenPlayer the second request for a dialogue is rejected without any further notice.

Digital IN0 – opens the dialogue for loading a file

Digital IN1 – opens the dialogue for saving a file, when the user selects a file that already exists a confirmation question pops up

Digital IN2 – here a dialogue is opened that can be used to select a directory instead of a file

Char OUT0 – when a file was selected and the file dialogue was left with "OK" the complete path to the selected file is returned here

Char OUT1 – when a file name was specified by the user and the file dialogue was left with "OK" the

complete path to the selected file is returned here; when this file already exists it can be overwritten by the application because in this case the user already confirmed that

Char OUT2 – when a directory was selected and the file dialogue was left with "OK" the complete path to that directory is returned here

This plug-in is located in flowplugins/libio_filedialogue.

5.3.3.7.2.9 IO Operation Flow Objects

Currently there exist no internal flow objects of this type.

5.3.3.7.2.10 External IO Operation Flow Objects

AS-i Serial Interface:

This plug-in can be used to access AS-i devices and to exchange digital data with them. It supports 240 output and 240 input lines (AS-i device range 1A..31A and 1B..31B). The AS-i master can be controlled via serial interface.

The serial parameters have to be set within the configuration panel together with a Poll Delay value that decides how often the inputs have to be polled for changes. A mapping of the inputs and outputs has not to be configured, they are assigned to the 8 inputs and outputs of the plug-in statically.

Digital IN0 - AS-i inputs 1A..8A

Digital IN1 - AS-i inputs 9A..16A

Digital IN2 - AS-i inputs 17A..24A

Digital IN3 - AS-i inputs 25A..31A and 1B..2B

Digital IN4 - AS-i inputs 3B..10B

Digital IN5 - AS-i inputs 11B..18B

Digital IN6 - AS-i inputs 19B..26B

Digital IN7 - AS-i inputs 27B..31B

Digital OUT0 - AS-i outputs 1A..8A

Digital OUT1 - AS-i outputs 9A..16A

Digital OUT2 - AS-i outputs 17A..24A

Digital OUT3 - AS-i outputs 25A..31A and 1B..2B

Digital OUT4 - AS-i outputs 3B..10B

Digital OUT5 - AS-i- outputs 11B..18B

Digital OUT6 - AS-i outputs 19B..26B

Digital OUT7 - AS-i outputs 27B..31B

This plug-in is located in flowplugins/libio_asi

AVR Net-IO (Ethernet).

This plug-in can be used to access the AVR Net-IO controller board via network to set digital outputs and to retrieve the state of digital and analogue inputs.

Within the configuration panel of the plug-in the network parameters of the board have to be set together with a polling time that decides how often it has to check for new or changed data. Independent from this polling time the outputs of the plug-in emit data only in case something has changed on the inputs of the board.

Digital IN0 - set the digital output 1 at the board

Digital IN1 - set the digital output 2 at the board

Digital IN2 - set the digital output 3 at the board

Digital IN3 - set the digital output 4 at the board

Digital IN4 - set the digital output 5 at the board

Digital IN5 - set the digital output 6 at the board

Digital IN6 - set the digital output 7 at the board

Digital IN7 - set the digital output 8 at the board

Digital OUT0 - the value of the digital input 1 at the board

Digital OUT1 - the value of the digital input 2 at the board

Digital OUT2 - the value of the digital input 3 at the board

Digital OUT3 - the value of the digital input 4 at the board

Numeric OUT4 - the value of the ADC input 1

Numeric OUT5 - the value of the ADC input 2

Numeric OUT6 - the value of the ADC input 3

Numeric OUT7 - the value of the ADC input 4

This plug-in is located in flowplugins/libio_avr_netio

AVR Net-IO (serial).

This plug-in can be used to access the AVR Net-IO controller board via its serial interface to set digital outputs and to retrieve the state of digital and analogue inputs.

Within the configuration panel of the plug-in the parameters of the serial interface have to be set together with a polling time that decides how often the plug-in has to check for new or changed data. Independent from this polling time the outputs of the plug-in emit data only in case something has changed on the inputs of the board.

Digital IN0 - set the digital output 1 at the board

Digital IN1 - set the digital output 2 at the board

Digital IN2 - set the digital output 3 at the board

Digital IN3 - set the digital output 4 at the board

Digital IN4 - set the digital output 5 at the board

Digital IN5 - set the digital output 6 at the board

Digital IN6 - set the digital output 7 at the board

Digital IN7 - set the digital output 8 at the board

Digital OUT0 - the value of the digital input 1 at the board

Digital OUT1 - the value of the digital input 2 at the board

Digital OUT2 - the value of the digital input 3 at the board

Digital OUT3 - the value of the digital input 4 at the board

Numeric OUT4 - the value of the ADC input 1

Numeric OUT5 - the value of the ADC input 2

Numeric OUT6 - the value of the ADC input 3

Numeric OUT7 - the value of the ADC input 4

This plug-in is located in flowplugins/libio_avr_serio

Capture Image:

Captures an image from a video device attached to the operating system. Within the plug-ins configuration panel the device number to capture the data from can be configured. Beside of that it is possible to specify following values that influence the resulting image:

- Scale to image width – the image is scaled to the given width (in unit pixels)
- Scale to image height – the image is scaled to the given height (in unit pixels)
- Don't scale – when this checkbox is set, the two previous values are ignored and the resulting image uses the largest resolution the capture device supports
- Brightness, Contrast, Gamma – changes the brightness, gamma an contrast of the captured image
- R, G, B – changed the red, green and blue image components; these values are applied also in case the image is converted to greyscale, here the colour correction is applied before the conversion to grey takes place
- Convert to greyscale – the captured image is converted to an image with 256 shades of grey (a luminance-based conversion is done here)
- Mirror X/Y – the image is mirrored along the X or Y axis

The plug-in is polled by the main application periodically:

Binary OUT7 – emits the captured image

This plug-in is located in flowplugins/libio_grabimg

Data Dummy:

This plug-in offers no functionality except than forwarding data from an input unmodified to the related output. This it does not offer any specific configuration parameter. This plug-in can be used for testing purposes, even tough it handles data only it was put into this functional category so that it can be used for device-related tests.

Digi IN0 – data are forwarded to Digi OUT0

Digi IN1 – data are forwarded to Digi OUT1

Numeric IN2 – data are forwarded to Numeric OUT2

Numeric IN3 – data are forwarded to Numeric OUT3

Char IN4 – data are forwarded to Char OUT4

Char IN5 – data are forwarded to Char OUT5

Binary IN6 – data are forwarded to Binary OUT6

Binary IN7 – data are forwarded to Binary OUT7

This plug-in is located in flowplugins/libio_dummy.

E1701M-IO

This plug-in can be used to use E1701M controller in a mode where no axes are moved but where only it's digital inputs and outputs are set and read.

Within the configuration panel of the plug-in the IP or the serial interface name of the controller have to be set together with a polling time that decides how often the plug-in has to check for new or changed data at DIn0 .. DIn7 of E1701M controller. Independent from this polling time the outputs of the plug-in emit data only in case something has changed on the inputs of the board.

Digital IN0 – set DOut0 (Step0) at the board

Digital IN1 – set DOut1 (Step1) at the board

Digital IN2 – set DOut2 (Step2) at the board

Digital IN3 – set DOut3 at the board

Digital IN4 – set DOut4 (Dir0) at the board

Digital IN5 – set DOut5 (Dir1) at the board

Digital IN6 – set DOut6 (Dir2) at the board

Digital IN7 - set DOut7 at the board

Digital OUT0 – the value of DIn0 at the board

Digital OUT1 – the value of DIn1 at the board

Digital OUT2 – the value of DIn2 at the board

Digital OUT3 – the value of DIn3 at the board

Digital OUT4 – the value of DIn4 at the board

Digital OUT5 – the value of DIn5 at the board

Digital OUT6 – the value of DIn6 at the board

Digital OUT7 – the value of DIn7 at the board

This plug-in is located in flowplugins/libio_e170m_io

Generic analogue temperature:

This plug-in can be used together with a scanner controller card which offers analogue input data (in BeamConstruct environment) or with control data providing analogue input values at binary IN7 (ControlRoom environment) to get temperature data from and to react on the measured values accordingly. The behaviour of the plug-in can be configured. The first configuration-panel “Basic” provides following set-up possibilities:

- Analogue input port – here the port of the scanner controller card has to be selected where the temperature sensor is connected with (in BeamConstruct environment) or the type of analogue provided on binary IN7 the plug-in has to work with.
- Temperature at 0 – specify the temperature which belongs to an analogue input value of 0
- Temperature at max – specify the temperature which belongs to the maximum possible analogue input value

The second panel “Thresholds” gives the possibility to define the behaviour of the plug-in via following options:

- Lower Temperature – here a default temperature can be set which is used to turn on an external device (e.g. a heater), when measured temperature falls below of the limit specified here, the device is turned on; within a BeamConstruct environment this value can be overridden by pen parameters, within ControlRoom Numeric IN0 can be used to set a new lower temperature value during operation
- Upper Temperature – here a default temperature can be set which is used to turn off an external device (e.g. a heater), when measured temperature rises above the limit specified here, this device is turned off; within a BeamConstruct environment this value can be overridden by pen parameters, within ControlRoom Numeric IN1 can be used to set a new lower temperature value during runtime
- Additional Digital Output Port – this parameter can be used within BeamConstruct only to configure the digital output port of a connected scanner controller card to be used for switching an external device; within a ControlRoom environment always Digital OUT6 is used for this
- Additional Digital Output Bit – this parameter can be used within BeamConstruct only to configure the pin number of the digital output port configured with previous parameter
- Stop Process on Temperature Deviation >10% - this parameter is working within BeamConstruct only, when this option is set, a running marking process is stopped, when the temperature is more than 10% above the limit specified by "Upper Temperature" parameter, the process is continued only when temperature falls below the "Upper Temperature" value

When this plug-in is used within BeamConstruct, the pen parameters are extended by an additional tab-pane “Heating” that offers some parameters that can be set to individual values for each pen. During marking these related values are sent to the plug-in whenever then pen changes:

- Lower Temperature – a temperature which is used to turn on an external device (like a heater), when measured temperature falls below of the limit specified here, the device is turned on; this device is toggeled via the output port and output bit specified in configuration panel “Thresholds”
- Upper Temperature – a temperature which is used to turn off an external device when measured temperature rises above the limit specified here; this device is toggeled via the output port and output bit specified in configuration panel “Thresholds”

Within a ControlRoom environment following IO's can be used:

Numeric IN0 (L) - to specify a new lower temperature that overrides the default value specified in settings

Numeric IN1 (H) - to specify a new upper temperature that overrides the default value specified in settings

Binary IN7 – control input which reacts on analogue data as specified in panel “Basic”

Numeric OUT0 (CURR) - the current temperature measured with the pyrometer

Digital OUT6 (ON) - using this output a device like a heater can be controlled, it goes to HIGH signal as soon as measured temperature falls below of the given lower temperature and switches back to LOW as soon as it exceeds the higher temperature threshold

Binary OUT7 (CTRL) - this output emits several additional control data in binary format signalling the state of the plug-in and the measured values

This plug-in is located in flowplugins/libio_generic_ana.

GPS (NMEA/SIRF binary):

This plug-in can be used for GPS-based positioning. It is able to access a GPS-device via serial interface (or via USB-connection that is mapped to a serial interface like most GPS mouses are doing it). It requires a GPS device that supports the NMEA ASCII protocol or the SIRF binary protocol.

For the plug-in itself only the serial connection parameters have to be configured, the communication protocol detection is done automatically.

Digital OUT0 (OK) – when there are position data available (GPS fix) this output returns a logic HIGH, whenever the GPS fix is lost for some reason this is signalled by a logic LOW

Numeric OUT1 (LAT) – in case of GPS fix here the latitude in unit degrees is given

Numeric OUT2 (LON) – this output emits the longitude in unit degrees

Numeric OUT3 (HGT) – the height (output is valid only in case of GPS fix)

Numeric OUT4 (SPD) – the speed over ground, here a value is returned only in case of GPS fix

Numeric OUT5 (T) – the time of the current week in seconds

This plug-in is located in flowplugins/libio_gps.

GPSd:

This plug-in works similar to the preceding one with two major differences:

- it is available for Linux only
- it makes use of GPSd and therefore supports nearly all GPS devices and their data formats that are out there

For a full description of the outputs and their purpose please refer to “GPS” above

This plug-in is located in flowplugins/libio_gpsd.

GSV-2 Measurement Amplifier:

The GSV-2 measurement amplifier is a special device that can be used to watch states of special equipment and to send a signal when a specified limit is exceeded.

This plug-in communicates via serial interface with the amplifier so within the first panel of the configuration panel the serial parameters have to be set. Within the second panel the default threshold can be specified that is used as limit (and triggers the LIM output) and the amplifier-specific gain value has to be set.

Digi IN0 (RES) – whenever the limit was exceeded the digital output LIM is set; using this input the amplifier can be reset and it starts again watching the measured values

Numeric IN1 (THR) – using this input the default threshold specified within the configuration panel can be overwritten

Digi OUT0 (LIM) – whenever the measured values exceed the given threshold this output is set to HIGH, after that happened and to continue measurement a reset has to be done using input RES

Numeric OUT1 (VL) – this output permanently emits the current peak value measured by the device

This plug-in is located in flowplugins/libio_gsv2.

Hermes Interface

This plug-in implements the Industry 4.0/Smart Factory communication and automation standard “Hermes” (aka “The Hermes Standard”) which can be used to integrate machine into an automated production environment where working pieces are handed over from one machine to an other automated and with transmission of additional information. Details and the full specification can b found at https://the-hermes-standard.info.

This plug-in requires following parameters to be configured:

- Previous machine IP – IP of the machine which hands over working pieces to the own one and therefore precedes in a production line; when this machine is not accessible, a begin-of-line position of the own machine is assumed
- Previous machine port – the port number of the previous machine
- Next machine IP – the port the next machine has to connect with; the next machine is the one the current machine hands over its working pieces to; when no next machine connects to the own one, a end-of-line position of the own machine is assumed
- Allow remote configuration changes – this option has to be enabled when a remote instance should be able to change the local configuration (according to the remote configuration feature defined in Hermes standard)
- Own machine identifier – here a text/name has to be specified which uniquely identifies the own machine within the current production line (or within the whole factory)

The signals and states which can be sent to the previous and next machine via the I/Os of this plug in are defined by the Hermes standard specification, so for details about the functionality and usage principle of the following commands and messages please refer to the Hermes specification:

Digital IN0 (RDY) – send a MachineReady message (according to the Hermes standard specification) to previous machine

Digital IN1 (!RDY) – send a RevokeMachineReady message to previous machine

Digital IN2 (STA) – send a StartTransport message to previous machine

Digital IN3 (STO) – send a StopTransport message to previous machine

Digital IN4 (AV) – send a BoardAvailable message to next machine

Digital IN5 (!AV) – send a RevokeBoardAvailable message to next machine

Digital IN6 (TF) – send a TransportFinished message to next machine

Digital OUT0 (AV) – a BoardAvailable message was received from previous machine, how to react on such a message and which responses are possible when the previous machine is in this state, is described in detail in Hermes standard specification

Digital OUT1 (!AV) – a RevokeBoardAvailable message was received from previous machine

Digital OUT2 (TF) – a TransportFinished message was received from previous machine

Digital OUT4 (RDY) – a MachineReady message was received from next machine

Digital OUT5 (!RDY) – a RevokeMachineReady message was received from next machine

Digital OUT6 (STA) – a StartTransport message was received from next machine

Digital OUT7 (STO) – a StopTransport message was received from next machine

This plug-in is located in flowplugins/libio_hermes.

JoyWarrior Accelerometer:

This plug-in can be used to retrieve data from the JoyWarrior24F accelerometer family (and compatible). It retrieves the current acceleration in X-, Y- and Z-direction (in $m*s^{-2}$) and calculates the current speed out of these values (in $m*s^{-1}$). The speed value may differ from the real speed over time, thus there exists a possibility to synchronize it to the real speed (using a different data source or by setting it to 0 when all motion has stopped).

Within the settings dialogue the input has to be specified where the JoyWarrior is connected to and the maximum acceleration it has been calibrated to has to be set.

Numeric IN3 (SPDX) – sets the current speed for the X-direction, the speed value given here is used to correct the internal, calculated value

Numeric IN4 (SPDY) – sets the current speed for the X-direction, the speed value given here is used to correct the internal, calculated value

Numeric IN5 (SPDZ) – sets the current speed for the X-direction, the speed value given here is used to correct the internal, calculated value

Numeric OUT0 (ACCX) – the current acceleration in X direction, here a value is emitted whenever the acceleration changes

Numeric OUT1 (ACCY) – the current acceleration in Y direction of the device, here a value is emitted whenever the acceleration changes

Numeric OUT2 (ACCZ) – the current acceleration in Z direction of the sensor, here a value is emitted whenever the acceleration changes

Numeric OUT3 (SPDX) – the current speed calculated out of the acceleration values in X direction, here a value is emitted whenever the acceleration changes

Numeric OUT4 (SPDY) – the current speed calculated out of the acceleration values in Y direction, here a value is emitted whenever the acceleration changes

Numeric OUT5 (SPDZ) – the current speed calculated out of the acceleration values in Z direction, here a value is emitted whenever the acceleration changes

This plug-in is located in flowplugins/libio_jwarrior.

LCDproc Client:

Using this plug-in a connection to a LCDproc-controlled display can be established in order to show some information there (for more information please refer to http://www.lcdproc.org). It is possible to control up to four lines that show either text or a horizontal bar graph or a mixture of both. Within the configuration the IP and port number of the LCDproc server have to be specified. Global display parameters that can be set here (but partially depend on the real capabilities of the hardware) are:

- Backlight – the mode the backlight should be operated with
- Horizontal Split Position – when both text and a bar graph have to be displayed this position specifies where the lines of the display have to be divided, when only one element has to be displayed at a line here a number has to be given that is at least as big as the number of the displays characters per line
- Scrolling Speed – when a text is set that is larger than a line within the display, it is scrolled automatically; this parameter defines the speed this text is scrolled with, it does not apply to bar

graphs, they scale automatically to fit into the available space

Next for every of the up to four lines the layout has to be specified, here it can be chosen if the text and the bar graph have to be displayed within the left or right part of the line. When the split position is set to a value bigger than the number of display line characters, the part that is defined for the right side is disabled, data that are set at the related input are ignored.

Char IN0 (LN1) - the text that has to be displayed in line 1 of the display

Numeric IN1 (LN1) - the value for the bar graph that is displayed in line 1 of the display; possible values are in range 0..100

Char IN2 (LN2) - the text that has to be displayed in line 2 of the display

Numeric IN3 (LN2) - the value for the bar graph that is displayed in line 2 of the display; possible values are in range 0..100

Char IN4 (LN3) - the text that has to be displayed in line 3 of the display

Numeric IN5 (LN3) - the value for the bar graph that is displayed in line 3 of the display; possible values are in range 0..100

Char IN6 (LN4) - the text that has to be displayed in line 4 of the display

Numeric IN7 (LN4) - the value for the bar graph that is displayed in line 4 of the display; possible values are in range 0..100

This plug-in is located in flowplugins/libio_lcdproc.

LUA IO:

This plug-in does not access any external hardware but provides some in- and outputs that can be programmed freely via an embedded LUA-script. For details regarding LUA programming language please refer to http://www.lua.org. The plug-in comes with a predefined set of in- and outputs of different types which can be read and written out of a LUA-script. This script can run once (or within a loop) or can be triggered by the plug-in to run periodically, means the plug-in restarts the script as soon as it has finished.

A LUA example project and related script that accesses all Ios is available in example-projects.

Following additional functions are supported by the plug-in:

`luaio_recv_data_callback(input)`

This is a callback function which has to be defined within an own script in order to get informed whenever some data change at the inputs. The parameter `input` contains a nuber that specifies which input comes with new data. After this function was executed it is possible to read the input data by calling `luaio_get_value()`.

`luaio_get_value(input)`

This function can be called to fetch input data from a specific input- As parameter the number of the input the data have to be read from needs to be handed over. The function returns the read value using a data type that corresponds to the type of input it is read from. Valid range for `input` is 0..7.

`luaio_has_new_value(input)`

Alternatively to using the callback function `luaio_recv_data_callback()` this function can be used to poll an input for new data. As parameter the number of the input to be polled has to be handed over (in range 0..7). The function returns `false` when no new data are available at this input or `true` otherwise. In last case the new data can be fetched by calling `luaio_get_value()`.

`luaio_set_value(output,value)`

Using this function an output of the plug-in can be set. Here `output` specifies which output to

change (in range 0..7) and `value` are the data to be set. The data type of `value` has to correspond to the data type of the output, elsewhere an error occurs.

Errors within the LUA-script loaded into the plug-in can be detected by executing the project in OpenDebugger. Whenever the script fails the related error information are given in debug information window. Using these information the script can be modified and debugged. According to the working principle of this plug-in the provided in- and outputs do not have a special meaning by default:

Digital IN0 – freely programmable digital input, corresponds to LUA boolean data type

Digital IN1 – freely programmable digital input, corresponds to LUA boolean data type

Numeric IN2 – freely programmable numeric input, corresponds to LUA numerical data types

Numeric IN3 – freely programmable numeric input, corresponds to LUA numerical data types

Char IN4 – freely programmable text input, corresponds to LUA string data type

Char IN5 – freely programmable text input, corresponds to LUA string data type

Char IN6 – freely programmable binary input, corresponds to LUA pointer data type

Char IN7 – freely programmable text input, corresponds to LUA pointer data type

Digital OUT0 – freely programmable digital output, corresponds to LUA boolean data type

Digital OUT1 – freely programmable digital output, corresponds to LUA boolean data type

Numeric OUT2 – freely programmable numeric output, corresponds to LUA numerical data types

Numeric OUT3 – freely programmable numeric output, corresponds to LUA numerical data types

Char OUT4 – freely programmable text output, corresponds to LUA string data type

Char OUT5 – freely programmable text output, corresponds to LUA string data type

Char OUT6 – freely programmable binary output, corresponds to LUA pointer data type

Char OUT7 – freely programmable text output, corresponds to LUA pointer data type

Modbus Addressable RTU Master:

This is a specific variant of the plug-in “Modbus RTU Master” that offers alternative possibilities to access a Modbus slave. Instead of fixed, pre-configured slave addresses and base address offsets here these values can be set dynamically from outside. That gives the possibility to access a much bigger range of slaves and all addresses that are available within such a slave. The configuration of this plug-in is similar to the non-addressable variant, only the type and behaviour of the in- and outputs do slightly differ.
The plug-in establishes a connection to a Modbus slave via a serial connection by using the pre-defined slave addresses and address offsets at the first approach. For every of the data in- and outputs a different operation mode can be defined within the configuration panel. The valid range of the input depends on that operation mode and its capabilities conform to the Modbus specification.

As global configuration this plug-in expects the port name of the serial interface and the communication parameters of the slave device. Beside of that a "Slave Poll Delay" has to be set, it specifies how often the connected slave has to be polled for new data. Two other options give the possibility to swap the bytes of the used address offset – which might be necessary for some Modbus slaves – and to write protocol-related logging information into a file. The last option can be used to find the reason for communication problems but should be handled with care, such a log file can become very big in a very short time depending on the load that is handled by this plug-in.
An other option is relevant for older project files only: “OpenAPC 1.x compatibility mode” should not be set within new projects, but has to be left enabled for version 1.x projects that make use of one of the single-bit IO's Coils or Input Registers. As long as this option is set, the 32 bit numbers that form 32 coil or register values are concatenated using the old style with exchanged byte order. In case it is not set the bits are concatenated within their real order and without grouping them into bytes.
For every of the two possible slave access connections inputs following parameters have to be set:

- Operation Mode – the method of data exchange with the slave; this mode influences the range of in- and output values and decides if data can be read and written or read only

- Slave Address – the default address of the slave to communicate with as long as it is not changed dynamically
- Base Address Offset – the 0-based address offset within the slaves address space that has to be used as long as it is not changed by external input data
- Unit ID – unit/slave number of the target

Numeric IN0 (0) – input 0 for the values to be written to the slave

Numeric IN1 (SL) – slave address for input 0, when this address is changed all data written to input IN0 (0) now are directed to this slave

Numeric IN2 (AD) – address offset for input 0, when this offset is changed all data written to input IN0 (0) now are directed to this slave and the output OUT0 once emits the data read from this slave address and address offset

Numeric IN4 (1) – input 1 for the values to be written to the slave

Numeric IN5 (SL) – slave address for input 1, when this address is changed all data written to input IN4 (1) now are directed to this slave

Numeric IN6 (AD) – address offset for input 1, when this offset is changed all data written to input IN4 (1) now are directed to this slave and the output OUT4 once emits the data read from this slave address and address offset

Numeric OUT0 – (0) the values that have been read from the slave, here new data are emitted whenever the related value changes within the slave or when a new address offset was set for this slave at IN2

Numeric OUT4 – (1) the values that have been read from the slave, here new data are emitted whenever the related value changes within the slave or when a new address offset was set for this slave at IN6

This plug-in is located in flowplugins/libio_modbus_rtu_addr.

Modbus RTU Master:

This plug-in establishes a connection to a Modbus slave via a serial connection and exchanges data with that slave. For every of the data in- and outputs a different operation mode can be defined within the configuration panel. The valid range of the input depends on that operation mode and its capabilities conform to the Modbus specification.

As global configuration this plug-in expects the port name of the serial interface and the communication parameters of the slave device. Beside of that a "Slave Poll Delay" has to be set, it specifies how often the connected slave has to be polled for new data. Two other options give the possibility to swap the bytes of the used address offset – which might be necessary for some Modbus slaves – and to write protocol-related logging information into a file. The last option can be used to find the reason for communication problems but should be handled with care, such a log file can become very big in a very short time depending on the traffic that is handled by this plug-in.
An other option is relevant for older project files only: “OpenAPC 1.x compatibility mode” should not be set within new projects, but has to be left enabled for version 1.x projects that make use of one of the single-bit IO's Coils or Input Registers. As long as this option is set, the 32 bit numbers that form 32 coil or register values are concatenated using the old style with exchanged byte order. In case it is not set the bits are concatenated within their real order and without grouping them into bytes.
For every of the 8 inputs following parameters have to be set:

- Operation Mode - the method of data exchange with the slave; this mode influences the range of in- and output values and decides if data can be read and written or read only
- Slave Address - the logic address of the slave to communicate with
- Base Address Offset - the 0-based address offset within the slaves address space

Numeric IN0..IN7 - the values to be written to the slave

Numeric OUT0..OUT7 - the values that have been read from the slave, here new data are emitted whenever the related value changes within the slave

This plug-in is located in flowplugins/libio_modbus_rtu.

Modbus TCP Master:

This plug-in establishes a connection to a Modbus slave via a TCP/IP network connection and exchanges data with that slave. For every of the data in- and outputs a different operation mode can be defined within the configuration panel. The valid range of the input depends on that operation mode and its capabilities conform to the Modbus specification.

As global configuration this plug-in expects the IP and port number the slave is listening at. Beside of that a "Slave Poll Delay" has to be set, it specifies how often the connected slave has to be polled for new or changed data. Two other options give the possibility to swap the bytes of the used address offset – which might be necessary for some Modbus slaves – and to write protocol-related logging information into a file. The last option can be used to find the reason for communication problems but should be handled with care, such a log file can become very big in a very short time depending on the traffic that is handled by this plug-in.
An other option is relevant for older project files only: “OpenAPC 1.x compatibility mode” should not be set within new projects, but has to be left enabled for version 1.x projects that make use of one of the single-bit IO's Coils or Input Registers. As long as this option is set, the 32 bit numbers that form 32 coil or register values are concatenated using the old style with exchanged byte order. In case it is not set the bits are concatenated within their real order and without grouping them into bytes.
For every of the 8 inputs following parameters have to be set:

- Operation Mode – the method of data exchange with the slave; this mode influences the range of in- and output values and decides if data can be read and written or read onlyb
- Base Address Offset – the 0-based address offset within the slaves address space
- Unit ID – unit/slave number of the target

Numeric IN0..IN7 - the values to be written to the slave

Numeric OUT0..OUT7 - the values that have been read from the slave, here new data are emitted whenever the related value changes within the slave

This plug-in is located in flowplugins/libio_modbus_tcp.

Modbus TCP Slave:

This plug-in acts as Modbus slave and is able to accept incoming Modbus master connections via a TCP/IP network connection. For every of the data in- and outputs a different operation mode can be defined within the configuration panel. The valid range of the input depends on that operation mode and its capabilities conform to the Modbus specification. Independent from the configured ranges and addresses this plug-in internally manages the whole address range that is available for a Modbus device. Beside of that it is possible to add more than one Modbus TCP Slave plug-in to a project and to define different I/O addresses for them, in this case all of them will share the same address space and act as one single device.

As global configuration this plug-in expects the IP and port number the slave has to listen at. Two other options give the possibility to swap the bytes of the used address offset – which might be necessary for some Modbus slaves – and to write protocol-related logging information into a file. The last option can be used to find the reason for communication problems but should be handled with care, such a log file can become very big in a very short time depending on the traffic that is handled by this plug-in. An other slave-specific value can be set via the “Maximum number of incoming connections”, it specifies how much slaves are allowed to connect at the same time. As soon as this limit is reached all additional connection attempts will be rejected as long as none of the already connected slaves disconnects. In case more such slave plug-ins are available within a project only the first one is responsible for accepting connections, all other ones will exchange data with this first instance. Thus only the “Max incoming connection” value of the first plug-in will be used for limiting the maximum number of slaves.
For every of the 8 inputs following parameters have to be set:

- Operation Mode – the method of data exchange with the slave; this mode influences the range of in- and output values and decides if data can be read and written or read only
- Base Address Offset – the 0-based address offset within the slaves address space

Numeric IN0..IN7 - the values to be written to the slave

Numeric OUT0..OUT7 - the values that have been read from the slave, here new data are emitted whenever the related value changes within the slave

This plug-in is located in flowplugins/libio_modbus_slave_tcp.

Network Client:

This plug-in can be used to establish bi-directional communication via a TCP/IP network. It is able to connect to a server and to exchange data with it. These data can be formatted depending on the configured mode or they can be handed over to the plug-in in as plain and unformatted data.

Within the configuration panel the IP and port number of the server to connect with have to be specified. Additionally a user name and password can be given. These authentication data are valid only with the formatted transfer modes and require a server software that is able to understand this format (for an example the "Network Receiver" plug-in is able to handle this format, other examples can be found within the SDK's OpenAPC interface implementations).

Next one of the following operation modes has to be selected:

- Command/Value – a mode for formatted data transmissions that consists of pairs of data: first the payload data (DATA) have to be set and second a unique command (CMD) has to be given; this command is assigned to the payload data and causes the transmission of both, the other communication partner will receive them as a pair
- Plain – every data block is transferred immediately, the CMD input behaves like the char payload data; this is also a formatted transmission mode, the binary data are transmitted using a special header that describes the data, all other data are transmitted as text values and are terminated by a CR
- Raw – every data block is transferred immediately and the data are not formatted in any way

Char IN0 (CMD) – input for a command value

Digital IN1 (DATA) – input for a digital value that has to be sent

Numeric IN2 (DATA) – input for a numeric value that has to be sent

Character IN3 (DATA) – input for text that has to be sent

Binary IN4 (DATA) – input for binary data that have to be sent

Char OUT0 (CMD) – output for a command that was received, this output emits data only in operation mode "Command/Value" and always together with a data value that was assigned to this command

Digital OUT1 (DATA) – received digital data

Numeric OUT1 (DATA) – received numeric data

Char OUT1 (DATA) – received text data

Binary OUT1 (DATA) – received binary data

This plug-in is located in flowplugins/libio_network_client.

Network Receiver:

Using this plug-in a server can be set up where several clients can connect to. It provides a uni-directional connection and emits data received from these clients. These data can be formatted depending on the operation mode the client is running with. So this plug-in expects clients that are able to handle a specific network protocol (like the "Network Client" or "Network Transmitter" PlugIns).

Within the configuration panel the IP and port number this plug-in has to listen at have to be specified together with the maximum number of incoming connections the plug-in will handle at the same time. Additionally a user name and password can be given. In case there are some authentication data set here, a connecting client has to authenticate at this plug-in. In case user name and password do not match, the incoming connection is rejected. A data transfer mode has not to be specified here, this plug-in automatically switches to the mode of the connecting client.

Char OUT0 (CMD) - output for a command that was received, this output emits data only when the

connected client uses operation mode "Command/Value" and always together with a data value that was assigned to this command

Digital OUT1 (DATA) – received digital data

Numeric OUT1 (DATA) – received numeric data

Char OUT1 (DATA) – received text data

Binary OUT1 (DATA) – received binary data

This plug-in is located in flowplugins/libio_network_out.

Network Server:

This plug-in can be used to establish bi-directional communication via a TCP/IP network. It is able to accept several incoming client connections and to exchange data with them. These data can be formatted depending on the mode the clients are using or they can be plain and unformatted

Within the configuration panel the IP and port number the server has to bind at have to be specified together with the maximum number of incoming connections the plug-in will handle at the same time. Additionally a user name and password can be given. These authentication data are valid only with the formatted transfer modes and require clients that are able to understand this format (for an example the "Network Transmitter" or "Network Client" plug-in are able to handle this format, other examples can be found within the SDK's OpenAPC interface implementations). In case there are some authentication data set here, a connecting client has to authenticate at this plug-in When user name and password do not match, the incoming connection is rejected. A data transfer mode has not to be specified here, this plug-in automatically switches to the mode of the connecting client.

Char IN0 (CMD) - input for a command value

Digital IN1 (DATA) input for a digital value that has to be sent

Numeric IN2 (DATA) input for a numeric value that has to be sent

Character IN3 (DATA) input for text that has to be sent

Binary IN4 (DATA) input for binary data that have to be sent

Char OUT0 (CMD) - output for a command that was received, this output emits data only in operation mode "Command/Value" and always together with a data value that was assigned to this command

Digital OUT1 (DATA) - received digital data

Numeric OUT1 (DATA) - received numeric data

Char OUT1 (DATA) - received text data

Binary OUT1 (DATA) - received binary data

This plug-in is located in flowplugins/libio_network_server.

Network Transmitter:

This plug-in works and behaves exactly like the "Network Client", except one major difference: it provides an uni-directional connection only, means it can only send data to a server it connects with and it doesn't provides any outputs. For a detailed description of the functionality and parameters of this plug-in please refer to description of "Network Client" plug-in

This plug-in is located in flowplugins/libio_network_in.

Optris LT Pyrometer:

This plug-in can be used together with an Optris LT pyrometer with serial interface to read temperature data from it and to react on the measured values accordingly. The behaviour of the plug-in can be configured. The first configuration-panel “Basic” provides possibilities to set-up the serial interface, here the same parameters have to be specified the Optris LT serial port is running with. Also to be configured in the first panel is the protocol format to be used when communicating with the pyrometer, here the option to be selected depends

on the exact device which is used.

The second panel “Thresholds” gives the possibility to define the behaviour of the plug-in via following options:

- Lower Temperature – here a default temperature can be set which is used to turn on an external device (e.g. a heater), when measured temperature falls below of the limit specified here, the device is turned on; within a BeamConstruct environment this value can be overridden by pen parameters, within ControlRoom Numeric IN0 can be used to set a new lower temperature value during operation
- Upper Temperature – here a default temperature can be set which is used to turn off an external device (e.g. a heater), when measured temperature rises above the limit specified here, this device is turned off; within a BeamConstruct environment this value can be overridden by pen parameters, within ControlRoom Numeric IN1 can be used to set a new lower temperature value during runtime
- Additional Digital Output Port – this parameter can be used within BeamConstruct only to configure the digital output port of a connected scanner controller card to be used for switching an external device; within a ControlRoom environment always Digital OUT6 is used for this
- Additional Digital Output Bit – this parameter can be used within BeamConstruct only to configure the pin number of the digital output port configured with previous parameter
- Stop Process on Temperature Deviation >10% - this parameter is working within BeamConstruct only, when this option is set, a running marking process is stopped, when the temperature is more than 10% above the limit specified by "Upper Temperature" parameter, the process is continued only when temperature falls below the "Upper Temperature" value

When this plug-in is used within BeamConstruct, the pen parameters are extended by an additional tab-pane “Heating” that offers some parameters that can be set to individual values for each pen. During marking these related values are sent to the plug-in whenever then pen changes:

- Lower Temperature – a temperature which is used to turn on an external device (like a heater), when measured temperature falls below of the limit specified here, the device is turned on; this device is toggeled via the output port and output bit specified in configuration panel “Thresholds”
- Upper Temperature – a temperature which is used to turn off an external device when measured temperature rises above the limit specified here; this device is toggeled via the output port and output bit specified in configuration panel “Thresholds”

Within a ControlRoom environment following IO's can be used:

Numeric IN0 (L) - to specify a new lower temperature that overrides the default value specified in settings

Numeric IN1 (H) - to specify a new upper temperature that overrides the default value specified in settings

Numeric OUT0 (CURR) - the current temperature measured with the pyrometer

Numeric OUT5 (ERR) - when this value returns a value not equal 0, there is a problem with the plug-in, it is no longer able to read data from the Optris pyrometer and therefore can't control switching of connected device any more

Digital OUT6 (ON) - using this output a device like a heater can be controlled, it goes to HIGH signal as soon as measured temperature falls below of the given lower temperature and switches back to LOW as soon as it exceeds the higher temperature threshold

Binary OUT7 (CTRL) - this output emits several additional control data in binary format signalling the state of the plug-in and the measured values

This plug-in is located in flowplugins/libio_optris_lt.

Parallel Interface:

This plug-in offers a cheap solution for digital IO's using standard PC hardware. Here the parallel interface (the Centronics/printer port) is used for setting up to 8 different output bits and for getting up to 5 digital inputs. This plug-in works well with a real parallel port hardware, the proper operation is not guaranteed when a USB-to-Parallel converter is used, here it depends on the converter and its driver software if this plug-in can be used. On some operation systems it may be necessary that the complete application is executed with superuser privileges in order to get access to the parallel port hardware.

Within the configuration panel only the base address of the parallel port has to be defined.

Digital OUT0..OUT7 (D0..D7) – the digital outputs that correspond to the lines D0..D7 of the parallel port

Digital IN0 (ACK) – digital input that emits the signal of the ACK-input whenever its state changes

Digital IN1 (BSY) – digital input that emits the signal of the BSY-input whenever its state changes

Digital IN2 (PE) – digital input that emits the signal of the PE-input whenever its state changes

Digital IN3 (SEL) – digital input that emits the signal of the SEL-input whenever its state changes

Digital IN4 (ERR)- digital input that emits the signal of the ERR-input whenever its state changes

This plug-in is located in flowplugins/libio_parallel_inout.

Serial Interface:

This plug-in can be used to establish bi-directional communication via a serial interface. The transferred data can be formatted depending on the configured mode or they can be plain and unformatted as handed over to the plug-in

Within the configuration panel the name of the serial interface and the serial connection parameters have to be set. Additionally a user name and password can be given. These authentication data are valid only with the formatted transfer modes and require a software on the opposite side of the connection that is able to understand this format (for an example an other "Serial Interface" plug-in would be able to handle this format correctly). Next one of the following operation modes has to be selected:

- Command/Value – a mode for formatted data transmissions that consists of pairs of data: first the payload data (DATA) have to be set and second a unique command (CMD) has to be given; this command is assigned to the payload data and causes the transmission of it, the other communication partner will receive them as pairs
- Plain – every data block is transferred immediately, the CMD input behaves like the “char” payload data; this is also a formatted transmission mode, the binary data are transmitted using a special header that describes the data, all other data are transmitted as text values and are terminated by the given termination character sequence
- Raw – every character data block is transferred immediately and the data are not formatted in any way, here the character and command input are the same
- Raw/E5AK checksum – every character data block is transferred immediately and the data are not formatted in any way but a XOR-based checksum is appended according to the specification of E5AK-controller; here the character and command input are the same

The following selection gives the possibility to specify the character that have to be added to the data before transmission, here it can be decided between CR (\r, 0x0D), LF (\n, 0x0A), CRLF (\r\n, 0x0D0A) and no termination character. This selection influences transmission only, for reception of data it is always checked if at least one of the characters \r or \n (or combinations of both) are detected.

The plug-in provides the following IOs:

Char IN0 (CMD) – input for a command value

Digital IN1 (DATA) – input for a digital value that has to be sent

Numeric IN2 (DATA) – input for a numeric value that has to be sent

Character IN3 (DATA) – input for text that has to be sent

Binary IN4 (DATA) – input for binary data that have to be sent

Char OUT0 (CMD) – output for a command that was received, this output emits data only in operation mode "Command/Value" and always together with a data value that was assigned to this command

Digital OUT1 (DATA) – received digital data

Numeric OUT1 (DATA) – received numeric data

Char OUT1 (DATA) – received text data

Binary OUT1 (DATA) – received binary data

This plug-in is located in flowplugins/libio_serial_inout.

Speak:

This plug-in offers a text to speech converter, texts and characters given to the plug-in are converted to spoken voice and are send to the audio hardware.

This plug-in is able to handle two independent voices that can be configured separately:

- Pitch – the pitch of the voice
- Speed – the speed the text is spoken with
- Word Gap – the delay in milliseconds that is done between every word
- Gender – the gender of the voice, here male, female or neutral can be chosen
- Language – this text to speech converter supports different native languages and tries to speak texts as natural as possible, here the language has to be set that fits to the input texts. PLEASE NOTE: the Speak-plug-in of course does not perform any translation, the language setting is responsible for a good, native pronunciation of the given texts!

Char IN0 - the text that has to be spoken using voice 1

Char IN4 - the text that has to be spoken using voice 2

This plug-in is located in flowplugins/libio_speak.

5.3.3.7.2.11 IO Operation Macros

Modbus RTU Master:

This macro contains the "Modbus RTU Master" plug-in and offers the possibility to access a Modbus slave. After the Modbus data access highly depends on the purpose of the slave and the set-up of the plug-in this macro offers access to a slave using one specific configuration. Depending on the real requirements it could be possible to change this macro.

To use it the internally used plug-in has to be configured first, here the serial connection parameters and the slave node address have to be set.

Digital IN DIN00..DIN31 – these inputs send single bits to the connected slave and change the state of the related outputs

Numeric NIN00..NIN03 – these inputs send numeric values to the connected slave and change the state of its related outputs

Digital OUT DOUT00..DOUT31 – these outputs return the state of the inputs of the connected slave

Numeric NOUT00..NOUT03 – these outputs return the state of the related values of the connected slave

Modbus TCP Master:

This macro uses the "Modbus TCP Master" plug-in and offers the possibility to access a Modbus slave. After the Modbus data access highly depends on the purpose of the slave and the set-up of the plug-in this macro offers access to a slave using one specific configuration. Depending on the real requirements it could be possible to change this macro.

To use it the internally used plug-in has to be configured first, here the network connection parameters have to be set.

Digital IN DIN00..DIN31 – these inputs send single bits to the connected slave and change the state of the related outputs

Numeric NIN00..NIN03 – these inputs send numeric values to the connected slave and change the state of its related outputs

Digital OUT DOUT00..DOUT31 – these outputs return the state of the inputs of the connected slave

Numeric NOUT00..NOUT03 – these outputs return the state of the related values of the connected slave

Weecoboard 4M IO:

This plug-in is designed to access the Weecoboard 4M by Aptasys. Since this is an ARM embedded board only the Linux ARM-runtime of the OpenAPC software package contains a fully functional variant of this plug-in. Packages for all other operating systems contain a skeleton if this plug-in without any functionality. There it can be used to design a project that has to be executed on an Weecoboard later. The IOs of this plug-in correspond to the digital IOs of the board directly:

Digital IN0 (1) – sets the digital output 1

Digital IN1 (2) – sets the digital output 2

Digital IN6 (red) – switches the red/orange LED of the board

Digital IN7 (green) – switches the green LED of the board

Digital OUT0 (1) – watches the digital input 1

…

Digital OUT7 (8) – watches the digital input 8

This plug-in is located in flowplugins/libio_weecoio.

Weecoboard LCD IO:

This plug-in is designed to access the Weecoboard LCD by Aptasys. Since this is an ARM embedded board only the Linux ARM-runtime of the OpenAPC software package contains a fully functional variant of this plug-in. Packages for all other operating systems contain a skeleton if this plug-in without any functionality. There it can be used to design a project that has to be executed on an Weecoboard later. The IOs of this plug-in correspond to the digital IOs of the board directly:

Digital IN0 (1) – sets the digital output 1

Digital IN1 (2) – sets the digital output 2

Digital IN6 (red) – switches the red/orange LED of the board

Digital IN7 (green) – switches the green LED of the board

Digital OUT0 (1) – watches the digital input 1

…

Digital OUT7 (4) – watches the digital input 4

This plug-in is located in flowplugins/libio_weecolcd.

5.3.3.7.2.12 Laser Flow Objects

There exist no application-internal flow elements of this type, all motion related functionalities are provided by external plug-ins.

5.3.3.7.2.13 External Laser Flow Objects

3rdEye(R) PSC140P:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to

putting a system into operation after a software update was installed.

This plug-in can be used to access the 3rdEye PSC140P scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Use Board Number – in case more than only one board is used within the same host computer this value specifies which of the boards has to be used by this plug-in
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the PSC140P and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Correction file – the .pcf correction file to be used for operation (provided by the vendor of the scan head)
- Firmware file – the .hex firmware file to be used to operate the scanner controller card (provided by 3rdEye)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the PSC140P manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.
The PSC140P scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X and Y inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the PSC140P plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not

cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_psc140p.

Coherent(R) Avia Ethernet Laser Controller:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured.

Using this plug-in a Coherent Avia laser can be controlled via Ethernet. To access the serial interface of the target laser an Ethernet to serial converter is required. The plug-in provides a basic configuration panel where the network parameters can be set according to the configuration of the TCP/IP to serial converter. A second panel gives the possibility to enter several laser-specific parameters. By default the plug-in can be used to set fixed frequency and power that are used all the time. On “Basic” configuration panel this behaviour can be changed. There are checkboxes to influence the behaviour of the plug-in:

- Control power during process – when this checkbox is set, the power-values defined in pen parameters are sent to the plug-in before the related vector data are sent to the scanner card
- Control frequency during process – when this checkbox is set, the frequency-values defined in pen parameters are sent to the plug-in before the related vector data are sent to the scanner card

The plug.in provides following in- and outputs to be used within the Flow Editor

Digital IN3 (L) – can be used to turn the laser on and off

Numeric IN4 (PWR) – controls the laser power

Numeric IN5 (FREQ) – controls the lasers output frequency

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to control the laser

Digital OUT6 (BSY) – signals if the plug-in is still communicating with the laser (HIGH) or if it has finished all operations so that other components can start its operation; this input has to be connected to the BSY-input of plug-in that is used to generate the control data

This plug-in is located in flowplugins/libio_coherent_avia.

Dummy scanner controller

This plug-in does not communicate with an external hardware but is a dummy for testing or demonstration purposes. From its configuration possibilities and parameters it is identical to the “HALaser E1803D Scanner Controller Card” but comparing to it, it does not communicate with the E1803D hardware. Instead all its functions return with “OK” or “ready” immediately.

This plug-in is located in flowplugins/libio_sc_dummy.

G-Code Controller:

WARNING: This plug-in is designed to control laser and other equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured.

This plug-in can be used to access laser-controller or other similar devices that accept G-Code (CNC) data transferred via Ethernet. This is a generic plug-in which does not expect a specific device but provides generic G-Code commands conform to the standard which normally should work with all G-Code-compatible devices. Following parameters can be configured:

- IP and Port – IP and port number of the device to connect with
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode
- Laser type – here the type of used laser has to be set; this parameter corresponds to the option of custom M7xx commands described below
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Custom initialisation commands – here it is possible to add own, device-specific G-Code commands that are sent during initialisation of the device
- Jump mode – here a mode can be chosen to perform fast jumps to a position while the laser or tool is turned off. It can be chosen between G-Code command "G00" which performs an as fast as possible movement, and "Jump speed" which performs a normal, linear movement using the jump speed specified below
- Include custom M7xx laser commands – when this option is set, some laser-specific commands are included into the generated G-Code which contain parameters and settings which are required for and specific to laser marking operations. Following commands are supported:
  M700 – the laser type, here a decimal number has to be set which specifies the laser type in detail

Laser type	M700 decimal parameter	Corresponding hexadecimal value
CO2	1207959553	0x48000001
YAG1 (Q-Switch and FPK at the beginning)	3355443202	0xC8000002
YAG2 (Q-Switch after FPK at the beginning)	2818572291	0xA8000003
YAG3 (Q-Switch after variable Q-Time)	2818572292	0xA8000004
CRF (Continuously running frequency)	1073741829	0x40000005
IPG	1207959560	0x48000008

  M701 – laser frequency in unit Hz (generic laser types only)
  M702 – laser power in unit %*1000 (generic and IPG lasers only)
  M703 – laser on delay (A) and laser off delay (B) in unit µsec

M704 – jump speed (A) and mark speed (B) unit mm/min
M705 – jump delay (A), mark delay (B) and in-polygon delay (C) in unit µsec
M706 – reserved for future use
M707 – working area position X and Y in unit specified by G20/G21 command
M708 – jump delay (A), mark delay (B) and variable in-polygon delay (C) in unit usec (maximum delay at 180 degrees angle)
M709 – working area size in X, Y and Z direction in unit specified by G20/G21 command
M710 – pulselength in unit nsec (IPG and CRF lasers with variable pulse width only)
M711 – wobble amplitude in X-direction (X) in unit µm, wobble amplitude in Y-direction (Y) in unit µm, wobble frequency (C) in unit mHz (Hz*1000)
M712 – reserved for future use
M713 – FPK (first pulse) in unit nsec (YAG lasers only)
M714 – simmer current in unit %*1000 (SPI laser only)
M715 – stand-by frequency (A) in unit Hz and stand-by pulsewidth (B) in unit nsec (CO_2 and YAG lasers only)
M716 – reserved for future use
M717 – waveform number (SPI laser only)
M718 – laser frequency in unit Hz (A) and laser pulsewidth in unit µsec (B, needs to fit to period of current laser frequency)
M719 – switch master oscillator (IPG only), 1 for on, 0 for off
M720 .. M799 – reserved for future use

- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates in relation to output

In Default-panel several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information:

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off (in case related jump mode is selected)
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on; this value is used only in ControlRoom-context and when the plug-in does not get binary control data

Following in- and outputs are available for the G-Code plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y. When laser is off, jump speed or command G00 is used for this movement, when it is turned on, mark speed is used.

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes, setting this input starts the movement to the coordinates at x,y,z. When laser is off, jump speed or command G00 is used for this movement, when it is turned on, mark speed is used.

Digital IN3 (L) – turns the laser on and off

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are currently supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in or within the scanner card

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this output has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

This plug-in is located in flowplugins/libio_sc_gcode.

Generic Laser Controller:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured.

Using this plug-in a laser can be controlled via serial interface ASCII commands. It is able to connect to a laser controller via COM-port/serial interface or TCP/IP, send predefined commands to it and receive a response from the laser. The received data can be interpreted in a minimum way. The plug-in provides a basic configuration panel where the serial interface parameters can be set according to the specifications of the laser to be connected.
As parameter “Interface” following values can be set:

- “COMx” or “/dev/ttyx” in case a local serial interface has to be used, here “x” is the number of the serial port
- “U1” in case the serial interface of a connected scanner controller card has to be used, in this case all data are routed to and from this controller card
- IP in style “aaa.bbb.ccc.ddd” in case network connection via TCP/IP has to be used

A second panel gives the possibility to enter the commands that have to be sent to the laser. Here several commands can be submitted. Every command has to be separated by specific characters that specify carriage return and/or line feed. Here for a carriage return the special character sequence “`[CR]`” has to be added for a line feed “`[LF]`” can be used. Please note: also in case only one laser command has to be sent, the CR/LF termination characters have to be set after the command. Which characters have to be used depends on the requirements of the used laser. The “Commands” configuration panel offers following configuration possibilities:

- Response End Character – specifies which characters are expected in every response from the laser; if “none” is chosen, the plug-in tries to receive some data after every command transmission but does not check the contents
- Device Open Command – here the command can specified that is sent to the laser when everything is initialised and when all connected devices are opened
- Device Open Delay – this is an optional delay that can be specified to let the plug-in wait after the Device Open Command is sent and before all other operations (like opening of other devices) are continued; this can be useful to let the laser start up/settle completely
- Ready for Marking Command – this command is sent to the laser whenever it has to be ready for marking, it is issued e.g. in case BeamConstruct opens the Mark dialogue
- Ready for Marking Delay – this is an optional delay that can be specified to let the plug-in wait after the Ready for Marking Command is sent and before all other operations (like start of marking operation) are continued; this can be useful to let the laser start up/settle completely
- Job Start Command – this command is sent to the laser prior to submission of first vector data to the scanner card
- Job Start Delay – this is an optional delay that can be specified to let the plug-in wait after the Job Start Command is sent and before sending of marking/vector data begins
- Laser On Command – this is a very specific command that should be used only in some very special applications, it is issued whenever the laser has to be turned on; this is done by the scanner card normally, so the user has to ensure that both commands don't overlap. Please note: Since serial communication is quite slow and not very responsive this way of turning on the laser is quite inaccurate, it should be used only when exact laser timing does not matter
- Laser On Delay – this is an optional delay that can be specified to let the plug-in wait after the Laser On Command is sent and before sending of marking/vector data begins
- Laser Off Command – this corresponds to the Laser On Command and is a very specific one that should be used only in some very special applications, it is issued whenever the laser has to be turned off; this is done by the scanner card normally, so the user has to ensure that both commands don't overlap. Please note: Since serial communication is quite slow and not very responsive this way of turning off the laser is quite inaccurate, it should be used only when exact laser timing does not matter
- Job End Command – this command is sent to the laser after submission of vector data to the

scanner card has finished

- End Ready for Marking Command – this command is sent to the laser whenever it no longer needs to be ready for marking, it is issued e.g. in case BeamConstruct closes the Mark dialogue
- Device Close Command – here the command can specified that is sent to the laser when everything is deinitialised and when all connected devices are closed; this command is invoked e.g. when the main application is closed

The plug.in provides following in- and outputs to be used within the Flow Editor

Digital IN3 (L) – can be used to turn the laser on and off

Numeric IN4 (PWR) – controls the laser power; this functionality is currently not supported and does not result in data sent to the laser

Numeric IN5 (FREQ) – controls the lasers output frequency; this functionality is currently not supported and does not result in data sent to the laser

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to control the laser

Digital OUT6 (BSY) – signals if the plug-in is still communicating with the laser (HIGH) or if it has finished all operations so that other components can start its operation; this input has to be connected to the BSY-input of plug-in that is used to generate the control data

Binary OUT7 (CTRL) – here control data are emitted which are used when the plug-in sends data to be used with the embedded serial port of a scanner controller card

This plug-in is located in flowplugins/libio_lc_generic.

HALaser E1701A Scanner Controller:

WARNING: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after any software update.

This plug-in can be used to access the HALaser Systems E1701A scanner controller cards for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.

Basic configuration parameters:

- IP or Serial Interface - here it has to be configured how and where the controller is accessible. In case of Ethernet connection the IP (in style aaa.bbb.ccc.ddd) has to be entered here, in case of USB-connection the name of the serial interface has to be given ("COMx" for Windows, "/dev/ttyACMx" for Linux where "x" is the number of the serial interface the controller was connected with)
- Ethernet Password – this parameter has to be given only in case Ethernet connection is used; here a password can be specified in order to identify connection to a controller. The same password has to be set within the related E1701 scanner controller. When password specified here and password configured at connected controller do not fit to each other, connection is closed and plug-in is not able to use this controller.
- Analogue voltage range – the E1701A is used to control analogue scanheads. Using this parameter the voltage range this scanhead works with can be selected in range 2,5..10 V (equal to +-2,5 V +-10 V control voltage at the cards outputs)
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of E1701 controller and to the manuals of the laser device vendor; within the plug-in the following laser types are available which make use of different parameters:

Laser Type	Signals
CO_2	• stand-by frequency and length are used for initialisation (tickle-pulse after opening the device) • default power control and laser on/off via PWM control (longer pulses cause laser emission, short tickle pulses keep laser in stand-by) • additional definitions in "CO_2"-tab-pane
YAG Mode 1/2/3	• stand-by frequency and length are used for initialisation (tickle-pulse after opening the device) • default power control and laser on/off via PWM control (longer pulses cause laser emission, short tickle pulses keep laser in stand-by) • first pulse killer supported at laser on (mode description see manual of E1701)
Laser Mode 4	• no predefined power control • definition of continuously running frequency in "Laser Mode 4"-tab-pane • continuously running frequency can be changed by parameter at beginning of mark
SPI (HS)	• stand-by frequency is used for initialisation and running independent from jump/mark (continuously running frequency) • frequency can be changed by parameter at beginning of mark • default power control via A0 analogue output • simmer control requires laser controller plug-in and serial interface communication • additional definitions in "SPI"-tab-pane
IPG (MO, Latch)	• stand-by frequency is used for initialisation and running independent from jump/mark (continuously running frequency with 50% pulse-width at LaserA output) • frequency can be changed by parameter at beginning of mark • default power control via latched LP8 digital outputs • additional definitions in "IPG/JPT"-tab-pane

- Additional power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here; this is an additional value which does not override or replace the default of the selected laser type. PLEASE NOTE: the possible outputs listed here require specific variants of E1701 scanner card and/or specific extension boards which are optional. Please refer to E1701 scanner controller manual for details which outputs are available together with base- and extension boards.
- Bitmap marking mode – using this combobox the marking mode for scanner bitmaps can be chosen; "Fast" uses the jump speed as long as the laser is turned off within a bitmap and moves with mark speed continuously elsewhere, "Jump'n'shoot" jumps to a pixel position, then shoots with the laser and jumps to the next, "Laser gate modulated" does not make use of any power outputs but turns on/off the laser gate (black and white marking only)
- A0 Correction Factor: this parameter applies to analogue output A0 of LP8 Extensionboard only. It can be used for analogue output voltages in range 0..5V but requires an external power supply which provides 5V exactly (voltages provided via USB are often below of 5V). This factor can be used to adjust the output voltage by giving a correction factor. It of course can not pull the A0 output up to %v when power supply delivers less than 5V but it can be used to adjust voltages below this limit.
- Correction file – a correction file to be used for operation (provided by the vendor of the scan head).

Here several formats are supported by the card directly and can be used without any conversion.

- Stand By Frequency – frequency the laser has to be driven with in standby mode (depends on selected laser type); if applicable this frequency is emitted after initialisation of the plug-in
- Stand By Length – pulse length the laser has to be driven with in standby mode (depends on selected laser type); if applicable this pulse width is used after initialisation of the plug-in
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates in relation to output
- Gain X / Gain Y – linear correction factor in X and Y direction to stretch or shrink the output
- Rotation – rotates the output by the given angle (in unit degree) using the centre of the working area as rotation point
- Slant X / Slant Y – trapezoidal correction angle along X and Y axis
- Use A3 as Digi Out – when this option is set, the analogue output A3 is operated as it would be a digital output

3D:

- Use third axis – this checkbox enables 3D marking mode
- Field depth – the Z-size of the field in unit mm; this value is used for displaying the correct working area in BeamConstruct and for calculation of the additional Z correction
- Z distance – the distance between last mirror of the scanhead and middle position of the working area in unit mm; this value is used for calculation of the additional Z correction

Signals:

- Mark-In-Progress output – specifies a digital output to be used as "Mark in progress" signal. When this option is set, the related output is set to HIGH as long as a marking operation is running. This option can be used only together with Digi I/O Extension board.
- Wait-External-Trigger output – specifies a digital output to be used as "Wait for external trigger" signal. When this option is set, the related output is set to HIGH as long as the controller has halted operation because it is waiting for an ExtStart signal.
- Laser Ready Input – specifies a digital input to be used as "Laser Ready" signal. When this option is set, marking is possible only in case a HIGH signal is detected at the input specified here. This option can be used only together with Digi I/O Extension board.
- Invert LaserGate – inverts the output logic of the LaserGate output
- Invert LaserA – inverts the output logic of the LaserA output
- Invert LaserB – inverts the output logic of the LaserB output
- Marking on-the-fly – can be used to enable marking on-the-fly operations, here the desired mode can be chosen; “1D” makes use of one encoder input (two signal lines) and compensates in one direction, here the direction and strength is specified by one of the following parameters, “2D” makes use of two encoder inputs (four signal lines) and compensates in both directions X and Y, here the strength is specified by the two following parameters, “Simulated” is intended to be used for testing purposes, here MOTF-functionality is available without an external encoder signal, it is generated by the controller card internally
- On-the-fly factor X / On-the-fly factor Y – specifies the encoder factor to be used along the related axis, the value has to be given in unit µm/inc

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and

Z are used or in case control data are used that do not contain motion/speed/delay information. These default parameters are used only in ControlRoom-projects and therefore do not appear within BeamConstruct:

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position, while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to E1701 manual.

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.

Within ControlRoom projects E1701 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the Control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the Control input is used, the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be started.

Following in- and outputs are available for the E1701 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y. When laser is off, jump speed is used for this movement, when it is turned on, mark speed is used.

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes, setting this input starts the movement to the coordinates at x,y,z. When laser is off, jump speed is used for this movement, when it is turned on, mark speed is used.

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the laser power value (using unit percent)

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are currently supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in or within the scanner card

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this output has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card (requires Digi I/O Extensionboard). These data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_e1701a.

HALaser E1701D Scanner Controller:

WARNING: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations

including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after any software update.

This plug-in can be used to access the HALaser Systems E1701D scanner controller cards for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.

Basic configuration parameters:

- IP or Serial Interface - here it has to be configured how and where the controller is accessible. In case of Ethernet connection the IP (in style aaa.bbb.ccc.ddd) has to be entered here, in case of USB-connection the name of the serial interface has to be given ("COMx" for Windows, "/dev/ttyACMx" for Linux where "x" is the number of the serial interface the controller was connected with)
- Ethernet Password - this parameter has to be given only in case Ethernet connection is used; here a password can be specified in order to identify connection to a controller. The same password has to be set within the related E1701 scanner controller. When password specified here and password configured at connected controller do not fit to each other, connection is closed and plug-in is not able to use this controller.
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of E1701 controller and to the manuals of the laser device vendor; within the plug-in the following laser types are available which make use of different parameters:

Laser Type	**Signals**
CO_2	• stand-by frequency and length are used for initialisation (tickle-pulse after opening the device) • default power control and laser on/off via PWM control (longer pulses cause laser emission, short tickle pulses keep laser in stand-by) • additional definitions in “CO_2”-tab-pane
YAG Mode 1/2/3	• stand-by frequency and length are used for initialisation (tickle-pulse after opening the device) • default power control and laser on/off via PWM control (longer pulses cause laser emission, short tickle pulses keep laser in stand-by) • first pulse killer supported at laser on (mode description see manual of E1701)
Laser Mode 4	• no predefined power control • definition of continuously running frequency in “Laser Mode 4”-tab-pane • continuously running frequency can be changed by parameter at beginning of mark
SPI (HS)	• stand-by frequency is used for initialisation and running independent from jump/mark (continuously running frequency) • frequency can be changed by parameter at beginning of mark • default power control via A0 analogue output • simmer control requires laser controller plug-in and serial interface communication • additional definitions in “SPI”-tab-pane
IPG (MO,	• to be used for IPG and compatible laser types which require master oscillator

Laser Type	Signals
Latch)	and power control via latched LP8 port • stand-by frequency is used for initialisation and running independent from jump/mark (continuously running frequency with 50% pulse-width at LaserA output) • frequency can be changed by parameter at beginning of mark • default power control via latched LP8 digital outputs • additional definitions in “IPG/JPT”-tab-pane

- Additional power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here; this is an additional value which does not override or replace the default of the selected laser type. PLEASE NOTE: the possible outputs listed here require specific variants of E1701 scanner card and/or specific extension boards which are optional. Please refer to E1701 scanner controller manual for details which outputs are available together with base- and extension boards.
- Bitmap marking mode – using this combobox the marking mode for scanner bitmaps can be chosen; “Fast” uses the jump speed as long as the laser is turned off within a bitmap and moves with mark speed continuously elsewhere, “Jump'n'shoot” jumps to a pixel position, then shoots with the laser and jumps to the next, “Laser gate modulated” does not make use of any power outputs but turns on/off the laser gate (black and white marking only)
- A0 Correction Factor: this parameter applies to analogue output A0 of LP8 Extensionboard only. It can be used for analogue output voltages in range 0..5V but requires an external power supply which provides 5V exactly (voltages provided via USB are often below of 5V). This factor can be used to adjust the output voltage by giving a correction factor. It of course can not pull the A0 output up to 5V when power supply delivers less than 5V but it can be used to adjust voltages below this limit.
- Correction file – a correction file to be used for operation (provided by the vendor of the scan head). Here several formats are supported by the card directly and can be used without any conversion.
- Stand By Frequency – frequency the laser has to be driven with in standby mode (depends on selected laser type); if applicable this frequency is emitted after initialisation of the plug-in
- Stand By Length – pulse length the laser has to be driven with in standby mode (depends on selected laser type); if applicable this pulse width is used after initialisation of the plug-in
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates in relation to output
- Gain X / Gain Y - linear correction factor in X and Y direction to stretch or shrink the output
- Rotation - rotates the output by the given angle (in unit degree) using the centre of the working area as rotation point
- Slant X / Slant Y - trapezoidal correction angle along X and Y axis

3D:

- Use third axis – this checkbox enables 3D marking mode
- Field depth – the Z-size of the field in unit mm; this value is used for displaying the correct working area in BeamConstruct and for calculation of the additional Z correction
- Z distance – the distance between last mirror of the scanhead and middle position of the working area in unit mm; this value is used for calculation of the additional Z correction

Signals:

- Mark-In-Progress output – specifies a digital output to be used as "Mark in progress" signal. When this option is set, the related output is set to HIGH as long as a marking operation is running. This option can be used only together with Digi I/O Extension board.
- Wait-External-Trigger output – specifies a digital output to be used as "Wait for external trigger" signal. When this option is set, the related output is set to HIGH as long as the controller has halted operation because it is waiting for an ExtStart signal.
- Laser Ready Input – specifies a digital input to be used as "Laser Ready" signal. When this option is set, marking is possible only in case a HIGH signal is detected at the input specified here. This option can be used only together with Digi I/O Extension board.
- Invert LaserGate – inverts the output logic of the LaserGate output
- Invert LaserA – inverts the output logic of the LaserA output
- Invert LaserB – inverts the output logic of the LaserB output
- Marking on-the-fly – can be used to enable marking on-the-fly operations, here the desired mode can be chosen; “1D” makes use of one encoder input (two signal lines) and compensates in one direction, here the direction and strength is specified by one of the following parameters, “2D” makes use of two encoder inputs (four signal lines) and compensates in both directions X and Y, here the strength is specified by the two following parameters, “Simulated” is intended to be used for testing purposes, here MOTF-functionality is available without an external encoder signal, it is generated by the controller card internally
- On-the-fly factor X / On-the-fly factor Y – specifies the encoder factor to be used along the related axis, the value has to be given in unit µm/inc

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information. These default parameters are used only in ControlRoom-projects and therefore do not appear within BeamConstruct:

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position, while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to E1701 manual.

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.

Within ControlRoom projects E1701 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the Control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the Control input is used, the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be started.

Following in- and outputs are available for the E1701 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y. When laser is off, jump speed is used for this movement, when it is turned on, mark speed is used.

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to

use three axes, setting this input starts the movement to the coordinates at x,y,z. When laser is off, jump speed is used for this movement, when it is turned on, mark speed is used.

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the laser power value (using unit percent)

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are currently supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in or within the scanner card

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this output has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card (requires Digi I/O Extensionboard). These data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_e1701.

HALaser E1803D Scanner Controller:

WARNING: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after any software update.

This plug-in can be used to access the HALaser Systems E1803D scanner controller cards for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.

Basic configuration parameters:

- IP or Serial Interface – here it has to be configured how and where the controller is accessible. In case of Ethernet connection the IP (in style aaa.bbb.ccc.ddd) has to be entered here, in case of USB-connection the name of the serial interface has to be given ("COMx" for Windows, "/dev/ttyACMx" for Linux where "x" is the number of the serial interface the controller was connected with)
- Ethernet Password – this parameter has to be given only in case Ethernet connection is used; here a password can be specified in order to identify connection to a controller. The same password has to be set within the related E1803 scanner controller configuration file. When password specified here and password configured at connected controller do not fit to each other, connection is closed and plug-in is not able to use this controller.
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of E1803 controller and to the manuals of the laser device vendor; within the plug-in the following laser types are available which make use of different parameters:

Laser Type	Signals
CO_2	• stand-by frequency and length are used for initialisation (tickle-pulse after opening the device) • default power control and laser on/off via PWM control (longer pulses cause laser emission, short tickle pulses keep laser in stand-by) • additional definitions in "CO_2"-tab-pane
YAG Mode 1/2/3	• stand-by frequency and length are used for initialisation (tickle-pulse after opening the device) • default power control and laser on/off via PWM control (longer pulses cause laser emission, short tickle pulses keep laser in stand-by) • first pulse killer supported at laser on (mode description see manual of E1803)
Laser Mode 4	• no predefined power control • definition of continuously running frequency in "Laser Mode 4"-tab-pane • continuously running frequency can be changed by parameter at beginning of mark
SPI (HS)	• stand-by frequency is used for initialisation and running independent from jump/mark (continuously running frequency) • frequency can be changed by parameter at beginning of mark • default power control via A0 analogue output • default simmer control via A1 analogue output • additional definitions in "SPI"-tab-pane
IPG (MO, Latch)	• to be used for IPG and compatible laser types which require master oscillator and power control via latched LP8 port • stand-by frequency is used for initialisation and running independent from jump/mark (continuously running frequency with 50% pulse-width at LaserA output) • frequency can be changed by parameter at beginning of mark • default power control via latched LP8 digital outputs • additional definitions in "IPG/JPT"-tab-pane
IPG (MO, A0)	• to be used for IPG and compatible laser types which require master oscillator and power control via analogue interface • default power control via A0 analogue output • RS232 serial interface connection used to read and reset error states

- Additional power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here; this is an additional value which does not override or replace the default of the selected laser type.
- Bitmap marking mode – using this combobox the marking mode for scanner bitmaps can be chosen; "Fast" uses the jump speed as long as the laser is turned off within a bitmap and moves with mark speed continuously elsewhere, "Jump'n'shoot" jumps to a pixel position, then shoots with the laser and jumps to the next, "Laser gate modulated" does not make use of any power outputs but turns on/off the laser gate (black and white marking only)
- Correction file – a correction file to be used for operation (provided by the vendor of the scan head). Here several formats are supported by the card directly and can be used without any conversion.
- Stand By Frequency – frequency the laser has to be driven with in standby mode (depends on

selected laser type); if applicable this frequency is emitted after initialisation of the plug-in

- Stand By Length – pulse length the laser has to be driven with in standby mode (depends on selected laser type); if applicable this pulse width is used after initialisation of the plug-in
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates in relation to output
- Gain X / Gain Y - linear correction factor in X and Y direction to stretch or shrink the output
- Rotation - rotates the output by the given angle (in unit degree) using the centre of the working area as rotation point
- Slant X / Slant Y - trapezoidal correction angle along X and Y axis

3D:

- Use third axis – this checkbox enables 3D marking mode
- Field depth – the Z-size of the field in unit mm; this value is used for displaying the correct working area in BeamConstruct and for calculation of the additional Z correction
- Z distance – the distance between last mirror of the scanhead and middle position of the working area in unit mm; this value is used for calculation of the additional Z correction

Signals:

- Mark-In-Progress output – specifies a digital output to be used as "Mark in progress" signal. When this option is set, the related output is set to HIGH as long as a marking operation is running.
- Wait-External-Trigger output – specifies a digital output to be used as "Wait for external trigger" signal. When this option is set, the related output is set to HIGH as long as the controller has halted operation because it is waiting for an ExtStart signal.
- Laser Ready Input – specifies a digital input to be used as "Laser Ready" signal. When this option is set, marking is possible only in case a HIGH signal is detected at the input specified here.
- Invert LaserGate – inverts the output logic of the LaserGate output
- Invert LaserA – inverts the output logic of the LaserA output
- Invert LaserB – inverts the output logic of the LaserB output
- Marking on-the-fly – can be used to enable marking on-the-fly operations, here the desired mode can be chosen; “1D” makes use of one encoder input (two signal lines) and compensates in one direction, here the direction and strength is specified by one of the following parameters, “2D” makes use of two encoder inputs (four signal lines) and compensates in both directions X and Y, here the strength is specified by the two following parameters, “Simulated” is intended to be used for testing purposes, here MOTF-functionality is available without an external encoder signal, it is generated by the controller card internally
- On-the-fly factor X / On-the-fly factor Y – specifies the encoder factor to be used along the related axis, the value has to be given in unit µm/inc
- A0/A1 0% voltage – these fields can be used to define the allowed voltage range for A0 and A1 analogue outputs. Here 0% defines the output voltage which has to be set at 0% power values.
- A0/A1 100% voltage – these fields can be used to define the allowed voltage range for A0 and A1 analogue outputs. Here 0% defines the output voltage which has to be set at 100% (means full) power values.

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information. These

default parameters are used only in ControlRoom-projects and therefore do not appear within BeamConstruct:

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position, while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to E1803 manual.

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.

Within ControlRoom projects E1803 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the Control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the Control input is used, the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be started.

Following in- and outputs are available for the E1803 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y. When laser is off, jump speed is used for this movement, when it is turned on, mark speed is used.

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes, setting this input starts the movement to the coordinates at x,y,z. When laser is off, jump speed is used for this movement, when it is turned on, mark speed is used.

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the laser power value (using unit percent)

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are currently supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in or within the scanner card

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this output has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card (requires Digi I/O Extensionboard). These data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_e1803.

Makeblock XY-Plotter:

Warning: This plug-in is designed to control laser or other dangerous equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured.

This plug-in can be used to control a Makeblock XY-plotter by sending appropriate G-Code commands to it via serial interface. It is independent from the tool used at this plotter, which can be a laser or any other hardware that can be toggled by straight on/off signals. The plug-in provides a basic configuration panel where the serial interface parameters can be set according to the specifications of the laser to be connected.

Within a second panel the default mark-speed to be used can be configured.

Beside of this it provides following in- and outputs to control the plug-in out of a ControlRoom application:

Numeric IN0 (X) - x-coordinate to move the plotter to, setting this value does not result in any movement

Numeric IN1 (Y) - y-coordinate to move the plotter to, when a value is given at this input, a movement to the given y-position and the previously set x-position is started; movement speed depends on input "L" which specifies jump/fast speed (L=0) or mark speed (L=1)

Digital IN3 (L) - turns on and off the laser/tool connected to the plotter

Binary IN7 (CTRL) - here a (continuous) stream of control data is accepted that contains data to fully control the plotter

Numeric OUT5 (ERR) - an error code specifying the current operational/error state of the plug-in and the connected plotter

Digital OUT7 (BSY) - signals if the plug-in is still communicating with the plotter (HIGH) or if it has finished all operations so that other components can start its operation; this input has to be connected to the BSY-input of plug-in that is used to generate the control data

This plug-in is located in flowplugins/libio_makeblock_xy.

Printer driver controlled:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after a software update was installed.

This plug-in can be used for all scanner controllers that can be accessed via a standard system printer driver. So within the configuration dialogue no laser/scanner related parameters can be set but values for line width, orientation and others:

- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates

The printer driver based scanner controller plug-in can be controlled via its inputs by transmitting a stream of movement data to the control input:

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control

data, it is necessary to synchronize several movement operations

This plug-in is located in flowplugins/libio_sc_printer.

Raylase(R) SP-ICE2:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after a software update was installed.

This plug-in can be used to access the Raylase SP-ICE2 scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- IP – the IP of the board to be controlled by this plug-in
- Port – the port number of the board
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the SP-ICE2 and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .gcd correction file to be used for operation (provided by the vendor of the scan head)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the SP-ICE2 manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.
The SP-ICE2 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the SP-ICE2 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in

Please note: the auto-calibration commands depend on the capabilities of the used scanner controller card and the used scan head.

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_spice2.

SAMLight(R) CCI:

This plug-in gives access to the SAMLight Client Control Programming Interface. It offers the possibility to remote-control the SAMLight application, to encapsulate its functionalities within a user interface that is much easier to handle and to automate processes and operations within the application. To use this plug-in a license for SAMLight may be necessary which is not part of this software package. Please contact the vendor of SAMLight for such a license.

There is also a macro available that encapsulates this plug-in and provides several pre-defined inputs and functionalities.

To use this plug-in SAMLight has to be configured for remote access via TCP/IP connection, the direct connection via function calls can't be used. Next within the configuration panel of the plug-in the IP and port number where SAMLight is listening at has to be set.

Access to the CCI and all it's functions happen via the inputs of the plug-in The Client Control Interface of SAMLight and therefore this plug-in act in handshake-mode, means after execution of a command the application has to wait for a response that is given at OUT0. When a new command is given before this response is available, that command is rejected. This fact is noticed within the OpenDebugger, when it

happens within the OpenPlayer the untimely command is rejected without any further action or notification.

Numeric IN0 (CMD) - whenever a number is set at this input the related command is executed via the CCI; some of the commands require additional parameters that can be set using the other inputs. Here only the mapping between the command number and the executed command is described, for a detailed documentation about the purpose of every command please refer to the manual of SAMLight. Together with the command the parameters are given, these parameter names correspond to the names of the inputs where the related values are expected at.

So when a description

99 - ScCciDoSometing(X,Y,TXT)

is given, that means, that the command "ScCciDoSomething" is executed whenever the number 99 is received at IN0. This command is executed using the values set for X (IN1), Y (IN2) and TXT (IN5). The commands parameters (the values at the other inputs than IN0) have to be set before the command is given.

Currently following commands are supported by this plug-in:

1 – ScCciTest(NAME)
2 – ScCciShowApp(A)
3 – ScCciGetWorkingArea(A)
4 – ScCciOpticMatrixTranslate(X,Y,A)
5 – ScCciOpticMatrixRotate(X,Y,A)
6 – ScCciOpticMatrixScale(X,Y)
7 – ScCciOpticMatrixReset()
8 – ScCciGetOpticMatrix(A)
9 – ScCciSetHead(A)
10 – ScCciGetHead()
11 – ScCciIsMarking()
12 – ScCciStopMarking()
13 – ScCciSwitchLaser(EN)
14 – ScCciMoveAbs(X,Y,A)
15 – ScCciSetPen(A)
16 – ScCciGetPen()
17 – ScCciResetSequence()
18 – ScCciResetCounter()
19 – ScCciResetSerialNumbers()
20 – ScCciIncSerialNumbers()
21 – ScCciDecSerialNumbers()
22 – ScCciResplitJob()
23 – ScCciSetMarkFlags(A)
24 – ScCciMarkEntityByName(NAME,EN)
25 – ScCciChangeTextByName(NAME,TXT)
26 – ScCciGetEntityOutline(NAME,A)
27 – ScCciTranslateEntity(NAME,X,Y,A)
28 – ScCciRotateEntity(NAME,X,Y,A)
29 – ScCciScaleEntity(NAME,X,Y,A)
30 – ScCciDeleteEntity(NAME)
31 – ScCciLoadJob(TXT,1,EN,1)
32 - ScCciImport(NAME,TXT,[auto-generated extension],X,A)
33 – ScCciSaveJob(TXT,A)
34 – ScCciNewJob()
35 – ScCciFitViewToWorkingArea()
36 – ScCciFitViewToEntities()
37 – ScCciFitViewToSelectedEntities()
38 – ScCciAutoCompensateOff()
39 – ScCciAutoCompensateRef()
40 – ScCciAutoCompensateCal()
41 – ScCciSetMode(A)
42 – ScCciGetMode()
43 - automatic command, retrieves number and name of entities in current job
44 - ScCciSetDoubleValue(1,A); override speed

45 - ScCciSetDoubleValue(3,A); override frequency
46 - ScCciSetDoubleValue(2,A); override power
49 - ScCciSetLongValue(4,A); set digital outputs of scanner card
51 - ScCciGetLongValue(4); get digital inputs of scanner card
52 - ScCciSetStringValue(6,TXT); save a screen shot with a width of 160 pixels
53 - ScCciSetStringValue(7,TXT); save a screen shot with a width of 320 pixels
53 - ScCciSetStringValue(9,TXT); save a screen shot with the full width

Numeric IN1 (X) – sets a value used as X parameter when a command is executed

Numeric IN2 (Y) – sets a value used as Y parameter when a command is executed

Numeric IN3 (A) – sets a value used as A parameter when a command is executed

Char IN4 (TXT) – sets a value used as TXT parameter when a command is executed

Char IN5 (NAME) – sets a value used as NAME parameter when a command is executed

Digital IN6 (EN) – sets a value used as EN parameter when a command is executed

Digital OUT0 – returns a HIGH signal when the last command could be executed successfully, LOW otherwise; only when this output was set the next command can be send to IN0, elsewhere it will be rejected when an other command is still in progress

Char OUT1 – when a command could not be executed successfully here the original error message (that is given from SAMLight) is emitted

Numeric OUT2 – some commands retrieve numeric values from SAMLight, this value is given at this output when the command could be executed successfully

Char OUT3 – some commands retrieve text values from SAMLight, this value is given at this output when the command could be executed successfully

This plug-in is located in flowplugins/libio_samcci.

SCANLAB(R) RTC ScanAlone:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and whenever a software update was installed.

This plug-in can be used to access the SCANLAB RTC ScanAlone scanner controller card for controlling a scan head and a connected laser in real-time. This plug-in supports the direct operation mode only, no writing of laser marking data onto the card or operating in stand-alone mode is supported.
Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the RTC ScanAlone and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .ctb correction file to be used for operation (provided by the vendor of the scanhead)
- Stand By Frequency – frequency the laser has to be driven with in standby mode

- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used when the direct axis control inputs X, Y and Z are used or when control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the position has to be changed with when the laser is turned off
- Mark Speed – the speed the position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the RTC ScanAlone manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.

The RTC ScanAlone scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time without additional synchronisation. Here the results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the RTC ScanAlone plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking (when the laser is turned on)

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; following commands are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in
- 1 – auto-calibration command, turn auto-calibration on and perform referencing for head 1
- 2 – auto-calibration command, turn auto-calibration on and perform referencing for head 2
- 5 – auto-calibration command, perform calibration at head 1

- 6 – auto-calibration command, perform calibration at head 2
- 9 – turn auto-calibration off for head 1
- 10 – turn auto-calibration off for head 2

Please note: the auto-calibration commands depend on the capabilities of the used scanner controller card and the used scan head

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_rtcscanalone.

SCANLAB(R) RTC3:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and whenever a software update was installed.

This plug-in can be used to access the SCANLAB RTC3 scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Use Board Number – in case more than only one board is used within the same host computer this value specifies which of the boards has to be used by this plug-in
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the RTC3 and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .ctb correction file to be used for operation (provided by the vendor of the scanhead)
- Firmware file – the .hex firmware file to be used to operate the scanner controller card (provided by SCANLAB)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked

- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used when the direct axis control inputs X, Y and Z are used or when control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the position has to be changed with when the laser is turned off
- Mark Speed – the speed the position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the RTC3 manual
  All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.

The RTC3 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time without additional synchronisation. Here the results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the RTC3 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking (when the laser is turned on)

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; following commands are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in
- 1 – auto-calibration command, turn auto-calibration on and perform referencing for head 1
- 2 – auto-calibration command, turn auto-calibration on and perform referencing for head 2
- 5 – auto-calibration command, perform calibration at head 1
- 6 – auto-calibration command, perform calibration at head 2
- 9 – turn auto-calibration off for head 1
- 10 – turn auto-calibration off for head 2

Please note: the auto-calibration commands depend on the capabilities of the used scanner controller card and the used scan head

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the

scanner and to control the laser, such a stream can contain complex marking operation

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_rtc3.

SCANLAB(R) RTC4:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after a software update was installed.

This plug-in can be used to access the SCANLAB RTC4 scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Use Board Number – in case more than only one board is used within the same host computer this value specifies which of the boards has to be used by this plug-in
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the RTC4 and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .ctb correction file to be used for operation (provided by the vendor of the scan head)
- Firmware file – the .hex firmware file to be used to operate the scanner controller card (provided by SCANLAB)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and

Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the RTC4 manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.
The RTC4 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the RTC4 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in
- 1 – auto-calibration command, turn auto-calibration on and perform referencing for head 1
- 2 – auto-calibration command, turn auto-calibration on and perform referencing for head 2
- 5 – auto-calibration command, perform calibration at head 1
- 6 – auto-calibration command, perform calibration at head 2
- 9 – turn auto-calibration off for head 1
- 10 – turn auto-calibration off for head 2

Please note: the auto-calibration commands depend on the capabilities of the used scanner controller card and the used scan head.

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_rtc4.

SCANLAB(R) RTC5:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after a software update was installed.

This plug-in can be used to access the SCANLAB RTC5 scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Use Board Number – in case more than only one board is used within the same host computer this value specifies which of the boards has to be used by this plug-in
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the RTC5 and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .ct5 correction file to be used for operation (provided by the vendor of the scan head)
- Firmware data directory – the directory that contains the firmware files to be used to operate the scanner controller card (provided by SCANLAB)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the RTC5 manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.
The RTC5 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the RTC5 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in
- 1 – auto-calibration command, turn auto-calibration on and perform referencing for head 1
- 2 – auto-calibration command, turn auto-calibration on and perform referencing for head 2
- 5 – auto-calibration command, perform calibration at head 1
- 6 – auto-calibration command, perform calibration at head 2
- 9 – turn auto-calibration off for head 1
- 10 – turn auto-calibration off for head 2

Please note: the auto-calibration commands depend on the capabilities of the used scanner controller card and the used scan head.

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_rtc5.

SCANLAB(R) RTC6:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to

putting a system into operation after a software update was installed.

This plug-in can be used to access the SCANLAB RTC6 scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Use Board Number – in case more than only one board is used within the same host computer this value specifies which of the boards has to be used by this plug-in
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the RTC6 and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .ct5 correction file to be used for operation (provided by the vendor of the scan head)
- Firmware data directory – the directory that contains the firmware files to be used to operate the scanner controller card (provided by SCANLAB)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the RTC6 manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.
The RTC6 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered.

Following in- and outputs are available for the RTC6 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in
- 1 – auto-calibration command, turn auto-calibration on and perform referencing for head 1
- 2 – auto-calibration command, turn auto-calibration on and perform referencing for head 2
- 5 – auto-calibration command, perform calibration at head 1
- 6 – auto-calibration command, perform calibration at head 2
- 9 – turn auto-calibration off for head 1
- 10 – turn auto-calibration off for head 2

Please note: the auto-calibration commands depend on the capabilities of the used scanner controller card and the used scan head.

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_rtc6.

SCAPS(R) FEB:

Using this plug-in the stand-alone option of the USC scanner card can be used. The communication is done via serial interface and gives the possibility to control the marking process of the scanner card while it is running in stand alone mode and without any other external applications that are controlling it. To use this plug-in or to initially put data to the scanner card a license may be necessary which is not part of this software package. Please contact the vendor of the USC/FEB stand alone option for such a license.

There is also a macro available that encapsulates this plug-in and provides several pre-defined inputs and functionalities.

To use this plug-in the scanner card has to run in stand-alone mode and with no other PC connected where a software is running at that influences the scanner card. Within the configuration panel of the plug-in the port name of the local serial interface and the serial communication parameters have to be set that correspond to the settings stored at the scanner card.

When the plug-in is initialised, the command "INIT" is sent to the card automatically. This command puts the stand-alone option into a proper operation mode. After it requires a nameable time to finish its operation, the scanner card is not able to execute any other command for several seconds.

Access to the scanner card and all it's functions happen via the inputs of the plug-in The communication protocol of the FEB stand-alone option works using handshake-mode, means after execution of a command the application has to wait for a response that is given at OUT0. When a new command is given before this response is available, that command is rejected. This fact is noticed within the OpenDebugger only, when it happens within the OpenPlayer the untimely command is rejected without any further notification.

Numeric IN0 (CMD) – whenever a number is set at this input the related command is sent to the scanner card; some of the commands require additional parameters that can be set using the following inputs. Here only the mapping between the command number and the executed command is described, for a detailed documentation about the purpose of every command please refer to the manual of the USC scanner card and the stand-alone option. Together with the command the parameters are given, these parameter names correspond to the names of the plug-ins inputs where the related values are expected at.

So when a description

99 – FX X Y TXT

is given, that means, that the command "FX" is executed whenever the number 99 is received at IN0. Here only the first word is the original USC/FEB command, that can be found within the manual. This command is executed using the values set for X (IN1), Y (IN2) and TXT (IN5). These values have to be set before the command is given.

Currently following commands are supported by this plug-in:

4 – TRB X Y A (transform the complete job bitwise)
5 – RT A (rotate the complete job)
6 – SC X Y A (scale the complete job)
31 – JN A (select a new job)
23 – ET EN (enable/disable triggering)
24 – M EN (start/stop marking)
25 – TX NAME TXT (set a new text to a serial number entity)
44 – OS A (overwrite speed)
45 – OF A (overwrite frequency)
46 – OP A (overwrite power)
47 – EM EN (enable/disable marking on-the-fly)
48 – LC A (set global loop count)
49 – OOU A (set digital outputs of the USC scanner card)
50 – OOF A (set digital outputs of the FEB stand alone option)
51 – OIU (get the digital inputs of the USC scanner card)
52 – OIF (get the digital inputs of the FEB stand alone option)

Numeric IN1 (X) – sets a value used as X parameter when a command is executed

Numeric IN2 (Y) – sets a value used as Y parameter when a command is executed

Numeric IN3 (A) – sets a value used as A parameter when a command is executed

Char IN4 (TXT) – sets a value used as TXT parameter when a command is executed

Char IN5 (NAME) – sets a value used as NAME parameter when a command is executed

Digital IN6 (EN) – sets a value used as EN parameter when a command is executed

Digital OUT0 – returns a HIGH signal when the last command could be executed successfully, LOW otherwise; only when this output was set the next command can be send to IN0, elsewhere it will be rejected because the preceding command is still in progress

Char OUT1 – when a command could not be executed successfully here an error message is returned that corresponds to the error code returned from the FEB stand-alone option

Numeric OUT2 – some commands retrieve numeric values from the scanner card, this value is given at this output when the command could be executed successfully

This plug-in is located in flowplugins/libio_samfeb.

SCAPS(R) USC-1/2:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may

otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after a software update was installed.

This plug-in can be used to access the SCAPS USC1 and USC-2 scanner controller cards via the generic Scanner Controller Interface for controlling a scan head and a connected laser in real-time. The usage of this interface may require additional licenses provided by the vendor of the USC scanner controller cards.
Within the configuration dialogue the plug-in, the scanner controller card and the interface to the card have to be configured according to the connected devices.
Basic:

- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the SCAPS manuals and to the manuals of the laser device vendor
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card and on the available SCAPS licenses
- Correction file – the .ucf correction file to be used for operation (provided by the vendor of the scan head or by SCAPS)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly factor X / On-the-fly factor Y – specifies the factor in X or Y direction for marking on-the-fly operation; this option depends on the capabilities of the used scanner controller card and on the available SCAPS licenses

Default:
Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the SCAPS manuals

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.

The SCI scanner controller card plug-in and thus the connected USC-1/USC-2 scanner controller card can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options

at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used, the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be started. Following in- and outputs are available for the SCI plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the percentual laser power

Numeric IN5 (FREQ) – the frequency in unit Hz the laser has to be driven with during marking

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; curently there is one command value supported:
0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_samsci.

SAMLight CCI Controller:

This macros encapsulates functionalities to access SCAPS SAMLight via the Client Control Interface. Internally it uses the "SAMLight CCI" plug-in As a first step this plug-in has to be configured to access SAMLight correctly, here the IP and port number where SAMLight is available at have to be set. Within SAMLight the same connection parameter and the TCP/IP access method need to be enabled for the Client Control Interface too.

The macro itself provides inputs of two types: parameter inputs that do not cause a data transmission when a value is set and command inputs that send a command to SAMLight, wait for the reaction and provide the result at the output. Some of these command inputs require values that have to be set at the parameter inputs before:

Numeric IN X – input parameter for X coordinates as it is used by several commands; this input only sets a parameter for later use, it does not invoke a command

Numeric IN Y – input parameter for Y coordinate values as it is used by some of the CCI commands; this input only sets a parameter for later use, it does not cause communication with the scanner card

Numeric IN Z – input parameter for Z coordinates as it is used by several commands Numeric IN ANGLE - angle input parameter; this input only sets a parameter for later use and does not invoke a command

Numeric IN VALUE – generic input parameter for numeric values used by some of the Client Control Interface commands

Char IN ENTITY_NAME – input to set a name of an entity for later use with a command

Char IN VALUE – generic input parameter for text values used by some of the CCI commands

Char IN LOAD_JOB – here a pathname is expected to a SJF-jobfile; this value is used to load the job specified by this path into SAMLight; as a second parameter this command uses the value of numeric input VALUE, if it is set to 1 a currently loaded job is dropped, in case it was set to 0 before, the previously loaded jobs are not removed but the new one is added to them

Digital IN MARK_TRIGG – this input enables the mark-trigger-mode using the currently selected job, after setting this input the current job is marked as soon as the scanner card receives an external trigger

Digital IN MARK_START – starts marking the currently selected job immediately

Digital IN ENTITY_ROT – this command rotates a single entity of the current job and requires four values that have to be set before it is used: the name of the entity (input ENTITY_NAME), the X- and Y-coordinates (input X and Y) of the centre where to rotate around and the rotation angle (input ANGLE)

Digital IN ENTITY_SCA – scales the entity specified by ENTITY_NAME in its X, Y and Z direction (these scaling factors have to specified by using inputs X, Y and Z before)

Digital IN ENTITY_TRA – translates the entity specified by ENTITY_NAME in given X, Y and Z direction (the transformation distances have to specified by using inputs X, Y and Z before)

Digital IN MATRIX_ROT – rotates the output matrix around the coordinates given at X and Y input using the value of the ANGLE input

Digital IN MATRIX_SCA – scales the complete output in X and Y direction using the values set at the inputs X and Y

Digital IN MATRIX_TRA – translates the output matrix using the values that have been set at the inputs X, Y and Z before

Digital IN MATRIX_RES – modifications of the output matrix stay active also when a new job is loaded; they can be reset to its default state (no scaling, no rotation and no translation) with the command that is sent to SAMLight when this input is triggered with a HIGH signal

Digital IN CHG_TEXT – this command changes the text of an Text2D object within the currently loaded job; the entity that has to be changed is specified by the value given at input ENTITY_NAME and the new text has to be set at character input VALUE before

Digital IN GET_ENTITI – this command does not require additional parameters; it retrieves the names of all entities that are currently loaded. When a job is loaded into SAMLight and a HIGH-signal is triggered at this input, first the number of available entities is retrieved and returned at output NUM_ENTITIES. Second the names of all these entities are returned at the output ENTITY_NAMES. When an entity of the current job does not have a name, simply an empty string is returned. The names provided here can be used to modify the jobs using the commands that require the name of an entity at input ENTITY_NAME

Digital IN USC_OUT0..USC_OUT5 – these inputs set the related digital optical output of the USC scanner card directly and immediately to 0 or 1 (depending on the signal set at these inputs)

Digital IN GET_USC_IN – when this input is set to HIGH a state update for the opto-inputs of the USC scanner card is requested, the inputs are read and the new state of the opto-inputs is given at the output lines USC_IN0..USC_IN5 of the macro

Digital IN GET_SCR_32 – this command creates a screen shot of the View2D of SAMLight and returns it at the binary output SCREEN; the shot of the View2D has a size of 320 pixels

Digital IN GET_SCR_FU – this command creates a screen shot of the View2D of SAMLight and returns it using the binary output SCREEN; here the View2D is given using its original size

Digital OUT CMD_OK – this output informs about the result of every command that was given, when a HIGH signal is given at this output the last command could be executed successfully, LOW means an error happened. In case of an error a plain error text is given at the ERROR-output. Please note: this output is updated only in case a command input is set, when values are set that do not invoke a command to the scanner card, CMD_OK is not used

Char OUT ERROR – the plain error text that is returned by SAMLight in case a command could not be executed successfully

Numeric OUT NUM_ENTITIES – this output is used together with the GET_ENTITI command input and returns the number of entities that are used by the current job

Char OUT ENTITY_NAMES – this output is used together with the GET_ENTITI command input and returns

the names of the entities that are used by the current job

Digital OUT USC_IN0..USC_IN5 – the state of the digital inputs of the USC scanner card, these inputs are updated every time the input GET_USC_IN of this macro is set

Binary OUT SCREEN – this output gives raw image data containing a screen shot of SAMLight's View2D

SCAPS FEB Controller:

Using this macro it is possible to implement a connection and communication with the FEB stand alone option of the SCAPS scanner cards. Internally this macro uses the "SCAPS(R)" FEB plug-in. As a first step this plug-in has to be set up, here the used serial parameters have to be configured (including the name of the used serial port).

The macro itself gives access to most of the commands supported by the plug-in via its IO's:

Numeric IN X – input parameter for X coordinates as it is used by several commands; this input only sets a parameter for later use, it does not invoke a command

Numeric IN Y – input parameter for Y coordinate values as it is used by some of the FEB commands; this input only sets a parameter for later use, it does not cause communication with the scanner card

Numeric IN Z – input parameter for Z coordinates as it is used by several commands

Numeric IN ANGLE – angle input parameter; this input only sets a parameter for later use and does not invoke a command

Numeric IN VALUE – generic input parameter for numeric values used by some of the FEB commands

Char IN ENTITY_NAME – input to set a name of an entity for later use with a command

Char IN VALUE – generic input parameter for text values used by some of the FEB commands

Digital IN LOAD_JOB – when a HIGH signal is set to this input a new job is selected using the job number of the VALUE input; as a result the outputs CMD_OK and - in case an error occurred during selecting a job - the output ERROR is set

Digital IN MARK_TRIGG – this input enables the mark-trigger-mode using the currently selected job, after setting this input the current job is marked as soon as the scanner card receives an external trigger

Digital IN MARK_START – starts marking the currently selected job immediately

Digital IN MARK_STOP – stops a running marking process immediately and disables the mark trigger mode in case it is active

Digital IN MATRIX_ROT – rotates the complete job around the centre of the working field using the value of the ANGLE input

Digital IN MATRIX_SCA – scales the complete job in X, Y and Z direction using the values set at the inputs X, Y and Z

Digital IN MATRIX_TRA – translates the current job within the range of the working field using the values that have been set at the inputs X, Y and Z before

Digital IN CHG_TEXT – when a HIGH signal is set at this input the contents of a serial number entity within the currently loaded job are changed; the entity that has to be changed is specified by the value set at input ENTITY_NAME, the new value it has to be set to is defined by the character input VALUE

Digital IN USC_OUT0..USC_OUT5 – these inputs set the related digital output of the USC scanner card directly and immediately to 0 or 1 (depending on the signal set at these inputs)

Digital IN GET_USC_IN – when this input is set to HIGH a state update for the digital inputs of the USC scanner card is requested, the inputs are read and the new state of these inputs is given at the output lines USC_IN0..USC_IN5 of the macro

Digital IN FEB_OUT0..USC_OUT7 – these inputs set the related digital output of the FEB stand alone option directly and immediately to 0 or 1 (depending on the signal set at these inputs)

Digital IN GET_FEB_IN – when this input is set to HIGH a state update for the digital inputs of the FEB stand alone option is requested, the inputs are read and the new state of these inputs is given at the output lines FEB_IN0..FEB_IN7 of the macro

Digital OUT CMD_OK – this output informs about the result of every command that was given, when a HIGH signal is given at this output the last command could be executed successfully, LOW means an error happened. In case of an error a plain error text is given at the ERROR-output. Please note: this output is updated only in case a command input is set, when values are set that do not invoke a command to the scanner card, CMD_OK is not used

Char OUT ERROR – the plain error text in case a command could not be executed successfully

Digital OUT USC_IN0..USC_IN5 – the state of the digital inputs of the USC scanner card, these inputs are updated every time the input GET_USC_IN of this macro is set

Digital OUT FEB_IN0..USC_IN7 – the state of the digital inputs of the FEB stand alone option, these inputs are updated every time the input GET_FEB_IN of this macro is set

Sintec(R) ETH6608:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured. Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and prior to putting a system into operation after a software update was installed.

This plug-in can be used to access the Sintec ETH6608 scanner controller card for controlling a scan head and a connected laser in real-time. Within the configuration dialogue the plug-in and the scanner controller card have to be configured according to the connected devices.
Basic:

- Board IP – the IP of the board that has to be accessed by this plug-in
- Laser type – the type of the connected laser, the field that was selected here directly corresponds to the laser tab panes where some additional, laser-specific settings can be done; for a detailed description of the available options please refer to the manual of the ETH6608 and to the manuals of the laser device vendor
- Scanner Output – the output that has to be used for accessing the scanner, here it can be chosen between analogue output and XY2/100 interface; this has to be set to the output the scanner is connected with
- Power output – the laser output power can be controlled via an additional output; the kind of output to be used can be defined here
- Number of Axes – specifies if the scanner controller has to operate in 2D or 3D mode, this option depends on the capabilities of the used scanner controller card
- Correction file – the .bco, .ctb, .ct5, .gcd or .ucf correction file to be used for operation (provided by the vendor of the scan head)
- Stand By Frequency – frequency the laser has to be driven with in standby mode
- Stand By Length – pulse length the laser has to be driven with in standby mode
- Field Left Position / Field Upper Position – defines the upper left coordinate of the field that has to be used by the scanner
- Field Size – the total size of the field that is used by the scanner, this parameter together with the left and upper position specifies the coordinates input vectors have to be located within, in case there are input data which define a position outside this field they will not be marked
- Swap X and Y – swaps the input X and Y coordinate to exchange X and Y axis values of the scanner
- Mirror X / Mirror Y – mirror the input coordinates
- On-the-fly speed X / On-the-fly speed Y – specifies the used static speed in X or Y direction for marking on-the-fly operation

Default:

Here several default parameters can be defined that are used in case the direct axis control inputs X, Y and Z are used or in case control data are used that do not contain motion/speed/delay information

- Jump Speed – the speed the scanners position has to be changed with when the laser is turned off
- Mark Speed – the speed the scanners position has to be changed with when the laser is turned on
- Laser Off Delay / Laser On Delay / Jump Delay / Mark Delay / Polygon Delay – these delays are used during marking operation, they become active when the laser is turned off or on, when the scanner jumps to an other position, when it moves to an other position while marking or when the movement direction is changed during marking; for a detailed description of these values, their usage and their influence to the marking result please refer to the ETH6608 manual

All following tab-panes contain laser-specific definitions that are used depending on the selected laser type. For information about these parameters and their usage please refer to the manuals and specifications of the used lasers.
The ETH6608 scanner controller card plug-in can be controlled via its inputs directly by transmitting coordinate values to the X, Y and Z inputs or by using the control-input where a stream of movement data will be accepted. It is recommended not to use both options at the same time or without additional synchronisation. The results may be undefined when direct and control stream commands are set at the same time. In case the control input is used the BSY-output has to be connected with the source of the control data, it signalises if the scanner controller card is still working or if it has finished so that other operations can be triggered. Following in- and outputs are available for the ETH6608 plug-in:

Numeric IN0 (X) – the X-position the scanner has to be moved to, setting a new value to this input does not cause any movement, it will be invoked only after the Y or Z input was set

Numeric IN1 (Y) – the Y-position the scanner has to be moved to, only in case the plug-in was configured to use two axes setting this input starts the movement to the coordinates at x,y

Numeric IN2 (Z) – the Z-position the scanner has to be moved to, only in case the plug-in was configured to use three axes setting this input starts the movement to the coordinates at x,y,z

Digital IN3 (L) – turns the laser on and off

Numeric IN4 (PWR) – the power the laser has to be driven with, here a value in range 0..100 is expected that corresponds to the as a percentage laser power value

Numeric IN6 (CMD) – this input can be used to set additional commands and to perform additional control operations using them; here following command values are supported:

- 0 – stop marking immediately and flush the list of control commands that may be already stored within the plug-in

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to move the scanner and to control the laser, such a stream can contain complex marking operation information

Digital OUT6 (BSY) – signals if marking is still active (HIGH) or if the scanner controller card has finished all operations; this input has to be connected to the BSY-input of plug-in that is used to generate the control data, it is necessary to synchronize several movement operations

Binary OUT7 (IN) – binary data that reflect the state of the digital inputs of this scanner controller card, these data are required by some plug-ins that need to react on these inputs and there have to be connected to them (like the scanner controller stepper motor plug-in)

This plug-in is located in flowplugins/libio_eth6608.

SPI G4 Laser RS232 Controller:

Warning: This plug-in is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured.

Using this plug-in a SPI G4 laser can be controlled via serial interface ASCII commands. It is able to connect to such a laser controller via COM-port/serial interface, send predefined and freely definable commands to it and receive responses from the laser.

The plug-in provides a basic configuration panel where the serial interface parameters can be set according

to the communication parameters of the lasers serial interface.

A second panel gives the possibility to specify behaviour of the plug-in and to enter custom commands that have to be sent to the laser. Every of these custom commands has to be separated by specific characters that specify carriage return and/or line feed. Here for a carriage return the special character sequence "`[CR]`" has to be added for a line feed "`[LF]`" can be used. Please note: also in case only one laser command has to be sent, the "`[CR][LF]`" termination characters have to be appended.

The "Commands" configuration panel offers following configuration possibilities:

- Device Open Command – here a custom command can specified that is sent to the laser when everything is initialised and when all connected devices are opened
- Device Open Delay – this is an optional delay that can be specified to let the plug-in wait after the Device Open Command is sent and before all other operations (like opening of other devices) are continued; this can be useful to let the laser start up/settle completely
- Ready for Marking Command – this custom command is sent to the laser whenever it has to be ready for marking, it is issued e.g. in case BeamConstruct opens the Mark dialogue
- Ready for Marking Delay – this is an optional delay that can be specified to let the plug-in wait after the Ready for Marking Command is sent and before all other operations (like start of marking operation) are continued; this can be useful to let the laser start up/settle completely
- Job Start Command – this command is sent to the laser prior to submission of first vector data to the scanner card
- Job Start Delay – this is an optional delay that can be specified to let the plug-in wait after the Job Start Command is sent and before sending of marking/vector data begins
- Control power during process - when this checkbox is set, the plug-in controls the power of the laser, means it sends the related command to it automatically on every change of output power and before vector data are sent that would use this new power value
- Control waveform during process - when this checkbox is set, the plug-in controls the used waveform, means it sends the related command to it automatically on every change of waveform and before vector data are sent that would use it
- Job End Command – this optional, custom command is sent to the laser after submission of vector data to the scanner card has finished
- End Ready for Marking Command – this optional command is sent to the laser whenever it no longer needs to be ready for marking, it is issued e.g. in case BeamConstruct closes the Mark dialogue
- Device Close Command – here an optional, custom command can specified that is sent to the laser when everything is deinitialised and when all connected devices are closed; this command is invoked e.g. when the main application is closed or when a user opens the global settings dialogue in BeamConstruct

The plug-in provides following in- and outputs to be used within the Flow Editor of ControlRoom:

Digital IN3 (L) – can be used to turn the laser on and off

Numeric IN4 (PWR) – controls the laser power

Binary IN7 (CTRL) – here a (continuous) stream of control data is accepted that contains data to control the laser

Digital OUT6 (BSY) – signals if the plug-in is still communicating with the laser (HIGH) or if it has finished all operations so that other components can start its operation; this input has to be connected to the BSY-input of plug-in that is used to generate the control data

This plug-in is located in flowplugins/libio_lc_spi.

5.3.3.7.2.14 Motion Flow Objects

There exist no application-internal flow elements of this type, all motion related functionalities are provided by external plug-ins.

5.3.3.7.2.15 External Motion Flow Objects

HALaser E1701M Focus Shifter Motor Controller:

Using this plug-in up one axis of E1701M stepper motor controller by HALaser Systems/OpenAPC Project Group can be accessed. It is intended to be used as Z-shifter/focus shifter primarily. For independent axis control of up to four axes please check description of E1701M Stepper Motor controller.

The controller can be accessed either via USB serial interface or via Ethernet.

Within the configuration panels of the plug-in there are two major set-up possibilities. The first panel provides general settings and parameters that apply to the plug-in and controller globally:

- IP or Serial Interface - here it has to be configured how and where the controller is accessible. In case of Ethernet connection the IP (in style aaa.bbb.ccc.ddd) has to be entered here, in case of USB-connection the name of the serial interface has to be given ("COMx" for Windows, "/dev/ttyACMx" for Linux where "x" is the number of the serial interface the controller was connected with)
- Ethernet Password - this parameter has to be given only in case Ethernet connection is used; here a password can be specified in order to identify connection to a controller. The same password has to be set within the related E1701M motion controller. When password specified here and password configured at connected controller do not fit, connection is closed and plug-in is not able to use this controller.

The settings within the axis-panel applies for the Z-axis:

- Resolution Factor – this value has to be used to specify conversion factor from shown position value (mm in planar mode, degrees in angular mode) to incremental position of the used motor (depending on its incremental resolution, step-configuration of used driver and used gearbox)
- Low Limit – the lower limit (in unit mm) is a soft limit specifying the minimum position the axis is allowed to move to in planar mode; in case of angular mode this value is ignored.
  PLEASE NOTE: when an axis was not referenced yet, this limit is useless since the real position of axis is not known to E1701M controller. So as very first an axis always should be referenced!
- High Limit – the higher limit (in unit mm) is a soft limit specifying the maximum position the axis is allowed to move to in planar mode; in case of angular mode this value is ignored.
  PLEASE NOTE: when an axis was not referenced yet, this limit is useless since the real position of axis is not known. So as very first an axis always should be referenced!
- Maximum Speed – this parameter can be used to limit maximum speed an axis can move with (in unit mm/sec for planar mode and deg/sec in angular mode). Independent from which speed value is specified during operation, this limit is never exceeded
- Acceleration Mode – specifies the shape of the ramp used during acceleration, here it can be chosen between linear (smooth acceleration), exponential (strong but not so smooth mode which may overshoot at end of acceleration phase) or s-shaped acceleration (very smooth but not very fast accelerating)
- Acceleration – factor that specifies how strong acceleration is for chosen acceleration mode
- Deceleration Mode – specifies the shape of the ramp used during deceleration, here it can be chosen between linear (smooth deceleration), exponential (strong but not so smooth mode which may lose increments at beginning of deceleration phase) or s-shaped deceleration (very smooth but not very fast)
- Deceleration – factor that specifies how strong deceleration is for chosen deceleration mode
- Stop Deceleration – factor that specifies strength of deceleration for all stop events (stop by limit switch, reference switch or by stop-command)
- Invert Direction – all axis use a "positive" and "negative" movement direction which are used to describe the motion direction. When this option is set, "positive" and "negative" are exchanged which may be useful to make UI-layout/appearance of these directions more convenient to user
- Reference Mode – after first start-up or when a motion error occurred where increments may got

lost, referencing is required for an axis. To do this, a reference switch is required at a defined mounting position (refer to related parameter below). Using reference mode parameter a sequence of movements in relation to this reference switch can be chosen that has to be performed on referencing.

- Reference Timeout – here a value in unit seconds can be specified. When referencing could not be finished after this time has elapsed, it will be cancelled since it seems axis is not able to find reference switch and/or movement is not possible for some reasons
- Reference Signal Input – here an input can be specified that is used as switch defining the referencing position
- Invert input logic – inverts the logic of the reference input to react on LOW level instead of HIGH
- Reference Speed 1/2/3 – referencing is done in up to three different steps (depending on used reference mode). Speeds for these steps should be slow in order to not to overshoot when reference position is reached. In case of more than one referencing step, these speed values should decrease for every step. Using these three parameters reference speed values can be specified. When a speed is given for a step that is not used in chosen reference mode, this value is ignored.
- Referenced Position – after referencing was finished successfully, the current axis position is set to the position value specified here
- Limit Switch 1/2 Input – beside the soft limits it is possible to define hardware-controlled position limits via limit switches. When such limit switches are used and connected to E1701M controller, the inputs they are connected with can be specified here. Whenever one of these switches sends a HIGH signal, current movement is stopped using stop-deceleration
- Invert input logic – inverts the logic of the limit switch input to react on LOW level instead of HIGH
- Auto-leave Limit Switch – when a limit switch is hit, normally no more motions are allowed until this switch is left. When this option is set and an axis hits a limit switch, it automatically moves back until this switch is left.
  PLEASE NOTE: when this option is set, a possibly unexpected motion may start in opposite direction!
- Encoder – E1701M controller board supports up to two external quadrature encoders that provide position information. When such an encoder is connected to decoder input 0 (digital inputs 0 and 1) or decoder input 1 (digital inputs 2 and 3) these inputs can be enabled here for an axis. When an external encoder is activated newly. the axis position on start-up is set to 0. To specify a different position, reference mode "No referencing" has to be chosen, a referencing position has to be set and referencing has to be performed. This does not cause any referencing operation but immediately sets the current, encoder-based position to the one specified as "Referenced Position". Alternatively a standard referencing operation as described above can be performed too.
  PLEASE NOTE: an encoder can be used for one axis only, when the same decoder input is configured for more than one axis, the results are undefined.
- Encoder Factor – this factor is used for calculation of axis incremental position, here the ratio between encoder pulses and motor increments has to be specified

When integrated into a ControlRoom-project following flow I/Os are available:

Numeric IN0 (POS) – absolute position in unit millimetres to move the axis to; here the factor out of the configuration is used to calculate the incremental position used by the drive out of the given metric distance; when a value is set here the movement starts immediately using the last speed value that was set at the MODE-input

Numeric IN1 (MODE) – at this input several other parameters can be specified depending on the given value; when a value greater than 0 is set it is taken as speed information for the next movement of axis 1, when a value of 0 is set the axis is stopped independent from its current position, a value of -1 causes a reference movement

Binary IN7 (CTRL) – input for 1D (Z) control data to manage the movement; here the related number of axes has to be configured properly. The related output OUT7 emits motion-synchronous Control data that can be used by a tool for machinery.

Numeric OUT0 (POS) – this output gives the current position of the axis, during a movement operation it changes permanently

Numeric OUT1 (MODE) – this is a state output that emits a 0 when the drive has stopped or the current speed otherwise

Digital OUT6 (BSY) – this output signals the busy state of the plug-in, it is set to HIGH as soon as a motion command arrives and is started to be executed and goes back to LOW only in case the axies does not move and no more motion commands are available; this output has to be connected with the BSY-input of the plug-in that provides the binary motion data

Binary OUT7 (CTRL) – output for the Control data delivered at IN7; here the control data are emitted synchronously dependent on the current movement position so that a tool can be controlled correctly dependent on the current position. This output also pays attention to possible tool on delays or tool off delays, as long as they are smaller than 0 the control data are emitted the given time before the related movement starts so that negative on/off delays are handled correctly. This plug-in is located in flowplugins/libio_e1701m_zshifter.

HALaser E1701M Stepper Motor Controller:

Using this plug-in up to four axes of E1701M stepper motor controller by HALaser Systems/OpenAPC Project Group can be accessed. These axes can be moved independently or synchronously via the control-input. In second case it is possible to perform flat, linear X/Y movements or three-dimensional X/Y/Z movements. It can be used e.g. together with an XY-Table and a tool for machining purposes. The controller can be accessed either via USB serial interface or via Ethernet.

Within the configuration panels of the plug-in there are two major set-up possibilities. The first panel provides general settings and parameters that apply to the plug-in and controller globally:

- IP or Serial Interface - here it has to be configured how and where the controller is accessible. In case of Ethernet connection the IP (in style aaa.bbb.ccc.ddd) has to be entered here, in case of USB-connection the name of the serial interface has to be given ("COMx" for Windows, "/dev/ttyACMx" for Linux where "x" is the number of the serial interface the controller was connected with)
- Ethernet Password - this parameter has to be given only in case Ethernet connection is used; here a password can be specified in order to identify connection to a controller. The same password has to be set within the related E1701M motion controller. When password specified here and password configured at connected controller do not fit, connection is closed and plug-in is not able to use this controller.
- Swap X and Y - this option is valid only in case the plug-in is fed with coordinates and not with separate position values for each axis, when it is set, X and Y coordinate are exchanged. When plug-in is used out of BeamConstruct, this option is inoperable

The settings within the axis-panels apply for one specific axis only. Here different settings for each axis can be given:

- Enable Axis – an axis is used and can be moved by this plug-in only in case this checkbox is set. When plug-in has to be used in coordinate mode (access via Control-input where XY or XYZ position coordinates are sent to it instead of separate axis position data) axis 1 and 2 have to be enabled for 2D coordinates (X and Y) and axis 1, 2 and 3 have to be enabled for 3D coordinates (X, Y and Z)
- Axis Mode – an axis can be operated in two modes: planar or radial. In planar mode it moves over a distance (using unit mm), in angular mode it moves to an angle (in unit degrees)
- Resolution Factor – this value has to be used to specify conversion factor from shown position value (mm in planar mode, degrees in angular mode) to incremental position of the used motor (depending on its incremental resolution, step-configuration of used driver and used gearbox)
- Low Limit – the lower limit (in unit mm) is a soft limit specifying the minimum position the axis is allowed to move to in planar mode; in case of angular mode this value is ignored.
  PLEASE NOTE: when an axis was not referenced yet, this limit is useless since the real position of axis is not known to E1701M controller. So as very first an axis always should be referenced!
- High Limit – the higher limit (in unit mm) is a soft limit specifying the maximum position the axis is allowed to move to in planar mode; in case of angular mode this value is ignored.
  PLEASE NOTE: when an axis was not referenced yet, this limit is useless since the real position of axis is not known. So as very first an axis always should be referenced!

- Maximum Speed – this parameter can be used to limit maximum speed an axis can move with (in unit mm/sec for planar mode and deg/sec in angular mode). Independent from which speed value is specified during operation, this limit is never exceeded
- Acceleration Mode – specifies the shape of the ramp used during acceleration, here it can be chosen between linear (smooth acceleration), exponential (strong but not so smooth mode which may overshoot at end of acceleration phase) or s-shaped acceleration (very smooth but not very fast accelerating)
- Acceleration – factor that specifies how strong acceleration is for chosen acceleration mode
- Deceleration Mode – specifies the shape of the ramp used during deceleration, here it can be chosen between linear (smooth deceleration), exponential (strong but not so smooth mode which may lose increments at beginning of deceleration phase) or s-shaped deceleration (very smooth but not very fast)
- Deceleration – factor that specifies how strong deceleration is for chosen deceleration mode
- Stop Deceleration – factor that specifies strength of deceleration for all stop events (stop by limit switch, reference switch or by stop-command)
- Invert Direction – all axis use a "positive" and "negative" movement direction which are used to describe the motion direction. When this option is set, "positive" and "negative" are exchanged which may be useful to make UI-layout/appearance of these directions more convenient to user
- Reference Mode – after first start-up or when a motion error occurred where increments may got lost, referencing is required for an axis. To do this, a reference switch is required at a defined mounting position (refer to related parameter below). Using reference mode parameter a sequence of movements in relation to this reference switch can be chosen that has to be performed on referencing.
- Reference Timeout – here a value in unit seconds can be specified. When referencing could not be finished after this time has elapsed, it will be cancelled since it seems axis is not able to find reference switch and/or movement is not possible for some reasons
- Reference Signal Input – here an input can be specified that is used as switch defining the referencing position
- Invert input logic – inverts the logic of the reference input to react on LOW level instead of HIGH
- Reference Speed 1/2/3 – referencing is done in up to three different steps (depending on used reference mode). Speeds for these steps should be slow in order to not to overshoot when reference position is reached. In case of more than one referencing step, these speed values should decrease for every step. Using these three parameters reference speed values can be specified. When a speed is given for a step that is not used in chosen reference mode, this value is ignored.
- Referenced Position – after referencing was finished successfully, the current axis position is set to the position value specified here
- Limit Switch 1/2 Input – beside the soft limits it is possible to define hardware-controlled position limits via limit switches. When such limit switches are used and connected to E1701M controller, the inputs they are connected with can be specified here. Whenever one of these switches sends a HIGH signal, current movement is stopped using stop-deceleration
- Invert input logic – inverts the logic of the limit switch input to react on LOW level instead of HIGH
- Auto-leave Limit Switch – when a limit switch is hit, normally no more motions are allowed until this switch is left. When this option is set and an axis hits a limit switch, it automatically moves back until this switch is left.
  PLEASE NOTE: when this option is set, a possibly unexpected motion may start in opposite direction!
- Encoder – E1701M controller board supports up to two external quadrature encoders that provide position information. When such an encoder is connected to decoder input 0 (digital inputs 0 and 1) or decoder input 1 (digital inputs 2 and 3) these inputs can be enabled here for an axis. When an external encoder is activated newly. the axis position on start-up is set to 0. To specify a different position, reference mode "No referencing" has to be chosen, a referencing position has to be set and referencing has to be performed. This does not cause any referencing operation but immediately

sets the current, encoder-based position to the one specified as "Referenced Position". Alternatively a standard referencing operation as described above can be performed too. PLEASE NOTE: an encoder can be used for one axis only, when the same decoder input is configured for more than one axis, the results are undefined.

- Encoder Factor – this factor is used for calculation of axis incremental position, here the ratio between encoder pulses and motor increments has to be specified

When integrated into a ControlRoom-project following flow I/Os are available:

Numeric IN0 (POS) – absolute position in unit millimetres to move axis 1 to; here the factor out of the configuration is used to calculate the incremental position used by the drive out of the given metric distance; when a value is set here the movement starts immediately using the last speed value that was set at the MODE-input

Numeric IN1 (MODE) – at this input several other parameters can be specified depending on the given value; when a value greater than 0 is set it is taken as speed information for the next movement of axis 1, when a value of 0 is set the axis is stopped independent from its current position, a value of -1 causes a reference movement

Numeric IN2 (POS) – performs a movement with axis 2 similar to IN0

Numeric IN3 (MODE) – defines the speed for axis 2 or controls its movement similar to IN1

Numeric IN4 (POS) – performs a movement with axis 3 similar to IN0

Numeric IN5 (MODE) – defines the speed for axis 3 or controls its movement similar to IN1

Binary IN7 (CTRL) – input for 2D (X and Y) or 3D (X, Y and Z) Control data to manage the movement; here the related number of axes has to be configured properly. The related output OUT7 emits motion-synchronous Control data that can be used by a tool for machinery.

Numeric OUT0 (POS) – this output gives the current position of axis 1, during a movement operation it changes permanently

Numeric OUT1 (MODE) – this is a state output that emits a 0 when the drive has stopped or the current speed otherwise

Numeric OUT2 (POS) – this is the position output for axis 2 (similar to OUT0)

Numeric OUT3 (MODE) – here the current state of axis 2 is given (similar to OUT1)

Numeric OUT4 (POS) – this is the position output for axis 3 (similar to OUT0)

Numeric OUT5 (MODE) – here the current state of axis 3 is given (similar to OUT1)

Digital OUT6 (BSY) – this output signals the busy state of the plug-in, it is set to HIGH as soon as a motion command arrives and is started to be executed and goes back to LOW only in case none of the axes moves and no more motion commands are available; this output has to be connected with the BSY-input of the plug-in that provides the binary motion data

Binary OUT7 (CTRL) – output for the Control data delivered at IN7; here the control data are emitted synchronously dependent on the current movement position so that a tool can be controlled correctly dependent on the current position. This output also pays attention to possible tool on delays or tool off delays, as long as they are smaller than 0 the control data are emitted the given time before the related movement starts so that negative on/off delays are handled correctly. This plug-in is located in flowplugins/libio_e1701m_stepper.

Schneider/IMS MDrive+:

This plug-in is able to control up to three axes of an Schneider/IMS MDrive+; there is also a macro available that encapsulates the low level functionalities described here. These axes can be moved independently or synchronously via the control-input to perform flat, linear X/Y movements (or three-dimensional X/Y/Z movements). This mode can be used e.g. together with an XY-Table and a tool for machining purposes.

This drive can be controlled via a serial interface where all axes are connected to this line. Independent from the fact if one or more axes are used, all of the used drives have to be pre-configured in the following way before operating them within a ControlRoom or BeamConstruct environment. Following the configuration steps are described together with the commands that have to be sent to the drive to do this configuration:

1. A unique device identifier has to be set for each axis (`DN=″X″` where “X” is the identifier).
2. The echo mode has to be set to mode "2" (`EM=2`).
3. The party mode has to be enabled (`PY=1`).
4. These parameters have to be stored to the drives permanently (`XS^J`).
5. When the homing/referencing functionality has to be used, the input S1 of the drive has to be connected to the home switch.

When this configuration was done successfully, the drives can be set up within the configuration panel of the plug-in. First of all the data of the serial interface have to be defined. Here the correct port name and the interface settings have to be given in order to establish the communication to the drive. Next for every used axis several parameters have to be given:

- Axis Mode – specifies the operational mode of the drive, here out of the two options planar (expects linear input data) and rotational (expects angular input values in unit degree) can be chosen
- Factor - specifies a factor that is used to calculate a metric value in millimetres or a angular value in degrees out of the drive-internal position information that is given in unit "increments", the exact factor depends on your hardware set-up and the used gear
- Low Limit - specifies the lower limit in unit millimetres where the axis is allowed to move to at minimum
- High Limit - specifies the upper limit in unit millimetres where the drive is allowed to move to at maximum
- Axis Name - is the device name that was stored within the drive in order to identify it
- Axis Home Timeout - is a time-out value in seconds that described after what time referencing has to be stopped when no home switch can be found

When a movement is running at an axis it can't be overwritten by a new position value immediately, such a new position information would be rejected by the plug-in. In this case the axis first has to be stopped and after the related speed-output indicates that this axis really stopped (by emitting a speed value of 0) the new position can be set at the POS-input. Within the debugger such an invalid positioning request would be indicated, within the OpenPlayer it is ignored without any further notice.

Numeric IN0 (POS) – absolute position in unit millimetres to move axis 1 to; here the factor out of the configuration is used to calculate the incremental position used by the drive out of the given metric distance; when a value is set here the movement starts immediately using the last speed value that was set at the MODE-input

Numeric IN1 (MODE) – at this input several other parameters can be specified depending on the given value; when a value greater than 0 is set it is taken as speed information for the next movement of axis 1, when a value of 0 is set the axis is stopped independent from its current position, a value of -1 causes a movement to the home switch

Numeric IN2 (POS) – performs a movement with axis 2 similar to IN0

Numeric IN3 (MODE) – defines the speed for axis 2 or controls its movement similar to IN1

Numeric IN4 (POS) – performs a movement with axis 3 similar to IN0

Numeric IN5 (MODE) – defines the speed for axis 3 or controls its movement similar to IN1

Binary IN7 (CTRL) – input for 2D (X and Y) or 3D (X, Y and Z) Control data to manage the movement; here the related number of axes has to be configured properly. The related output OUT7 emits motion-synchronous Control data that can be used by a tool for machinery.

Numeric OUT0 (POS) – this output gives the current position of axis 1, during a movement operation it changes permanently

Numeric OUT1 (MODE) – this is a state output that emits a 0 when the drive has stopped or the current speed otherwise

Numeric OUT2 (POS) – this is the position output for axis 2 (similar to OUT0)

Numeric OUT3 (MODE) – here the current state of axis 2 is given (similar to OUT1)

Numeric OUT4 (POS) – this is the position output for axis 3 (similar to OUT0)

Numeric OUT5 (MODE) – here the current state of axis 3 is given (similar to OUT1)

Digital OUT6 (BSY) – this output signals the busy state of the plug-in, it is set to HIGH as soon as a motion command arrives and is started to be executed and goes back to LOW only in case none of the axes moves and no more motion commands are available; this output has to be connected with the BSY-input of the plug-in that provides the binary motion data

Binary OUT7 (CTRL) – output for the Control data delivered at IN7; here the control data are emitted synchronously dependent on the current movement position so that a tool can be controlled correctly dependent on the current position. This output also pays attention to possible tool on delays or tool off delays, as long as they are smaller than 0 the control data are emitted the given time before the related movement starts so that negative on/off delays are handled correctly.

This plug-in is located in flowplugins/libio_ims_mdrive2.

Sill(R) Focus Ethernet:

This is a one-axis motion plug-in to access a Sill focus shifter via Ethernet. Since the hardware is controlled via serial interface an Ethernet to serial converter is required in order to use this plug-in. In first configuration panel the TCP/IP configuration of that converter has to be set. In second panel the motion parameters can be set the focus shifter has to be operated with. Here a conversion factor from distance (in unit µm) to internal representation of the focus shifter has to be given. Beside of that the upper and lower movement limit (in unit µm) can be specified as well.

Since this is a motion plug-in that can control only one axis, only following in- and outputs are available:

Numeric IN0 (POS) – absolute position in unit micrometers to move the axis to; here the factor out of the configuration is used to calculate the incremental position used by the focus shifter out of the given metric distance; when a value is set here the movement starts immediately

Numeric IN1 (MODE) – at this input several other parameters can be specified depending on the given value; to start a movement a value greater than 0 has to be set here prior to specifying the target positiion at input 0, when a value of 0 is set the axis is stopped independent from its current position, a value of -1 causes a movement to the home position

Binary IN7 (CTRL) – input for binary motion control data

Numeric OUT0 (POS) – this output gives the current position of axis 1, during a movement operation it changes permanently

Numeric OUT1 (MODE) – this is a state output that emits a 0 when the drive has stopped or the current speed otherwise

Digital OUT6 (BSY) – this output signals the busy state of the plug-in, it is set to HIGH as soon as a motion command arrives and is started to be executed and goes back to LOW only in case the axis stopped moving and no more motion commands are available; this output has to be connected with the BSY-input of the plug-in that provides the binary motion data

This plug-in is located in flowplugins/libio_sill_focus.

IO Sync Motor Trigger (Scannercard):

This plug-in is not a motor driver in common sense. Instead of submitting motion and position information, it sends a digital signal to a connected device which then can be used to trigger external equipment. This digital signal stays at HIGH until the external device responds with an other signal. As soon as this signal is received, the software assumes the motion is completed and continues with other operations. This plug-in sends and receives the digital signals via a connected scanner controller card and its digital IOs. Thus the binary output OUT7 has to be connected with the binary input of the scanner controller cards plug-in (in case of a ControlRoom environment). Beside of that a feedback from the scanner controller is necessary, so the binary output of the scanner controller card needs to be connected with IN7 of this plug-in in order to get feedback about the digital inputs on that scanner controller.

The plug-in is able to control up to three axes via these digital on/off signals These axes will operate sequentially, two or more axes will not be triggered at the same time.

As global parameter only the digital output port and the digital input port of the connected scanner controller card have to be chosen. Here is is up to the user to select IO ports that really exist for the used scanner controller, in case an IO is specified that is not available, no data can be send to it or no input information can be received. Which bit of the selected IO's is used for which purpose and which axis, can be specified within the axis-related configuration panels:

- Motion start output – the output bit at the previously chosen output port type that is used to send the command to an external device to start a motion operation. This output stays at HIGH until the external device signals the motion operation was completed
- Motion done input – the input bit at the previously chosen input port where the external device signals the end of a motion operation
- Reference start output – the output bit at the previously chosen output port type that is used to send the command to an external device to start a referencing operation. This output stays at HIGH until the external device signals the referencing operation was completed
- Reference Timeout – a timeout value in seconds after what a referencing operation will be cancelled no matter if the connected device signalled a “motion done” or not
- Referenced Position – the metric position (in unit millimetres) that has to be set at the referenced position

When a movement is running at an axis it can't be overwritten by a new position value immediately, such a new position information would be rejected by the plug-in. In such a case the axis first has to be stopped and after the related speed-output indicates that this axis really has stopped (by emitting a speed of 0) the new position can be set at the POS-input. Within the debugger such an invalid positioning request would be indicated, within the OpenPlayer it is ignored without any further notice.

Numeric IN0 (MOVE) – the absolute position (in unit mm or degrees) to move the axis 1 to, for this movement the speed is used that was set last at the MODE input; when a value is set here and no other movement operation is in progress at this axis the motor is moved to the related position immediately; the position value given here is not used to set a position somewhere but only returned as current position after the external device has signalled the motion operation was done

Numeric IN1 (MODE) – this input is used for different control commands depending on the value of the number that is set here; when the given value is greater than 0 it is interpreted as speed and the value is used for the next movement command given at the preceding input, when the input value is equal to 0 the motor is stopped immediately (stop-command), a value of -1 causes a movement to the home/referencing position (here the reference input has to be connected and configured properly); a speed information given with a value greater than 0 is not used to drive a motor with but only returned as current speed until the external device has signalled the motion operation was done

Numeric IN2 (POS) – performs a movement with axis 2 similar to IN0

Numeric IN3 (MODE) – defines the speed for axis 2 or controls its movement similar to IN1

Numeric IN4 (POS) – performs a movement with axis 3 similar to IN0

Numeric IN5 (MODE) – defines the speed for axis 3 or controls its movement similar to IN1

Binary IN7 (CTRL) – input for 2D (X and Y) or 3D (X, Y and Z) control data to manage the movement; here the related number of axes has to be configured properly. The related output OUT7 emits motion-synchronous Control data that can be used by a tool for machinery.

Digital OUT6 (BSY) – busy-signal, specifies if at least one axis is still moving; this output is set to LOW as soon as all current operations have been finished by this plug-in

Binary OUT7 (CTRL) – output for the Control data delivered at IN7; here the control data are emitted synchronously dependent on the current movement position so that a tool can be controlled correctly dependent on the current position.

This plug-in is located in flowplugins/libio_scanctrl_iosync.

IWH Robot:

Using this plug-in an Isel Wafer Handling Robot and compatible devices can be controlled via ASCII commands that are transmitted using a serial interface. Thus the serial port and its parameters have to be

set up according to the used settings. As additional parameter the home time-out has to be specified: this time-out is active when the robot references and/or moves to the home position. When this operation lasts longer as the specified home time-out value (in unit seconds) the operation is cancelled.

Char IN0 (CMD) – an ASCII-command that is sent to the robot; depending on the kind of command the outputs of this plug-in will change their state

Digital IN1 (STP) – a HIGH signal at this input stops any movement of the robot immediately

Digital IN2 (HOM) – when a HIGH signal is sent to this input the robot performs a movement to its home position; depending on the kind of encoder that is used that operation includes referencing for all axes

Numeric OUT0 (R) – current radius of the robots position in 1/1000 inches

Numeric OUT1 (T) – current angular position of the robots arm in unit 1/100 degrees

Numeric OUT2 (Z) – current height of the robots arm in unit 1/1000 inches; all these three values change its state permanently during a movement of the robot

Digital OUT3 (VCMD) – this output signals the current state of the chucks vacuum valve, it signals if it is turned on (vacuum enabled, wafer gripped) or of

Digital OUT4 (VSEN) – here the state of the vacuum sensor is given, this output emits a HIGH as soon as the vacuum valve is turned on and a wafer is detected on the chuck

Digital OUT5 (MOV) – this output is set to HIGH as long as the robot moves, a new movement can be sent to the robot only when LOW is given here

Numeric OUT6 (MAP) – this output is related to an (optional) wafer scanner: after a SCN command is sent to the robot and the scanning process is finished, the scanning result is emitted at this output using a sequence of numbers that specifies the state of every wafer (these numbers are the result of a MAP-command)

This plug-in is located in flowplugins/libio_iwh_robot.

Panasonic Minas:

With this plug-in Panasonic Minas servo controllers can be accessed and used for motion operations. There is also a macro available that encapsulates the low level functionalities described here and gives more comfortable possibilities to access the drive. The plug-in is able to handle up to four axes that are connected to the same RS485 line. The Minas-controllers itself need to have a basic configuration in order to operate with this plug in. So the motion position at index 1 has to be configured for absolute movements, single operation and to use the speed 1. All other motion position indices (in range 2..60) can be used and configured freely. Beside of that every connected drive needs to be set up with a unique address so that the plug-in is able to identify and access the axes correctly.

Within the configuration panel different parameters have to be set up. First of all the serial interface has to be configured, here the interface name and the connection parameters need to be set according to the parameters that are configured for the controller. Next for every used axis several parameters have to be given:

- Factor – specifies a factor that is used to calculate a metric value in millimetres out of the drive-internal position information that is given by the position encoder, the exact factor depends on your hardware set-up and the gear you are using
- Low Limit – specifies the lower limit in unit millimetres where the axis is allowed to move to at minimum
- High Limit – specifies the upper limit in unit millimetres where the drive is allowed to move to at maximum
- Axis Address – the address that is set at the controller in order to identify it at the serial line
- Axis Home Timeout – is a time-out value in seconds that described after what time referencing has to be stopped when no home switch could be found

When a movement is running at an axis it can't be overwritten by a new position value immediately. Such a new position information would be rejected by the plug-in. In such a case the axis first has to be stopped and after the related speed-output indicates that the axis really has stopped (by emitting a speed of 0) the new position can be set at the POS-input. Within the debugger an invalid positioning request would be indicated,

within the OpenPlayer it is ignored without any further notice. Every of the up to four axes is controlled via a pair of inputs:

Numeric IN0 (POS) – the absolute position (in unit mm) to move the axis 1 to, for this movement the speed is used that was set last at the MODE input; when a value is set here the motor is moved to the related position immediately

Numeric IN1 (MODE) – this input is used for different control commands depending on the value of the number that is set here; when the given value is greater than 0 it is interpreted as speed and the value is used for the next movement command given at the preceding input, when the input value is equal to 0 the motor is stopped immediately, a value of -1 causes a movement to the home/referencing position (here the reference input has to be connected and configured properly) and values in range -2..-60 perform a movement to the indexed positions that are stored within the controllers internal configuration for positions 2..60

Numeric IN2 (POS) – performs a movement with axis 2 similar to IN0

Numeric IN3 (MODE) – defines the speed for axis 2 or controls its movement similar to IN1

Numeric IN4 (POS) – performs a movement with axis 3 similar to IN0

Numeric IN5 (MODE) – defines the speed for axis 3 or controls its movement similar to IN1

Numeric IN6 (POS) – performs a movement with axis 4 similar to IN0

Numeric IN7 (MODE) – defines the speed for axis 4 or controls its movement similar to IN1

Numeric OUT0 (POS) – this output gives the current position of axis 1, during a movement operation it changes permanently

Numeric OUT1 (MODE) – this is a state output that emits a 0 when the drive has stopped, the current speed with values greater than 0 and the current drives error code with values smaller than 0; when the drive is in error state this error is reset automatically with the next movement command

Numeric OUT2 (POS) – this is the position output for axis 2 (similar to OUT0)

Numeric OUT3 (MODE) – here the current state of axis 2 if given (similar to OUT1)

Numeric OUT4 (POS) – this is the position output for axis 3 (similar to OUT0)

Numeric OUT5 (MODE) – here the current state of axis 3 if given (similar to OUT1)

Numeric OUT6 (POS) – this is the position output for axis 4 (similar to OUT0)

Numeric OUT7 (MODE) – here the current state of axis 4 if given (similar to OUT1)

This plug-in is located in flowplugins/libio_panasonic_minas.

Stepper Motor Driver (Scannercard):

Using this plug-in a stepper motor can be controlled via a connected scanner controller card. This plug-in does not control such a card directly but it generates output data that can be handled by a scanner controller card. Thus the binary output OUT7 has to be connected with the binary input of the scanner controller cards plug-in (in case of a ControlRoom environment). Beside of that a feedback from the scanner controller is necessary, so the binary output of the scanner controller card needs to be connected with IN7 of this plug-in in order to get feedback about the digital inputs on that scanner controller.

The plug-in is able to control up to three axes via two output signals "Step" and "Direction". These axes will operate sequentially, two or more axes will not move at the same time.

As global parameter only the digital output port and the digital input port of the connected scanner controller card have to be chosen. Here is is up to the user to select IO ports that really exist for the used scanner controller, in case an IO is specified that does not exists, no data can be send to it or no input information can be received. Which bit of the selected IO's is used for which purpose and which axis can be specified within the axis-related configuration panels:

- Axis Mode – specifies the operational mode of the drive, here out of the two options planar (expects linear input data) and rotational (expects angular input values in unit degree) can be chosen
- Factor – specifies a factor that is used to calculate a metric value in millimetres or a rotational value

in degrees out of the drive-internal position information that is given in unit "increments", the exact factor depends on your hardware set-up and the gear you are using

- Low Limit – specifies the lower limit in unit millimetres where the axis is allowed to move to at minimum
- High Limit – specifies the upper limit in unit millimetres where the drive is allowed to move to at maximum
- Acceleration – a value that specifies how smooth the drive should start its movements
- Deceleration – a value that specifies how smooth the drive should stop its movements
- Step Signal Output – the output bit at the previously chosen output port type that is used for the pulsed step signal; here it has to be made sure that an output is used only once, overlapping mappings would cause problems during operation
- Direction Signal Output – the output bit at the output port that was chosen in general configuration panel, this output bit is used to define the movement direction of the axis; here it has to be made sure that an output is used only once, overlapping mappings would cause problems during operation
- Invert Direction – inverts the direction output / the movement direction for this axis
- Invert input logic – inverts the logic of the limit switch input to react on LOW level instead of HIGH
- Lazy input switch – when stepper motors are operated via this software control and digital in/outputs of a controller card instead of a native motion controller, all operations are done somewhat slower because of the non-realtime behaviour of the operating system BeamConstruct is running at. Thus the whole system can't react as fast as an embedded controller e.g. when a reference switch was hit and therefore may not find it successfully. In this case the user has the possibility to slow down the reference speed and/or to set this option in order to tolerate more variances during referencing. To still get an accurate reference position under these conditions, the “Reference Speed (Leave)” needs to be set to lower values.
- Reference Mode – here it can be specified if a home/referencing operation has to be done or not and if yes which method of referencing has to be used; the chosen mode defines using which direction the software shall search for the reference switch and using which direction the switch shall be left afterwards
- Reference Signal Input – the input bit at the previously chosen input port where the reference switch is connected to
- Reference Speed (Enter) – the speed that has to be used to search for the reference switch
- Reference Speed (Leave) – the speed that has to be used to leave the reference switch once it was found; the correct referenced position is that position, where the reference switch just turns off; this part of referencing has an direct influence on the accuracy on the resulting position, the lower it is, the more exact the position will be. Please also refer to option “Lazy input switch” described above.
- Reference Timeout – a timeout value in seconds after what a referencing operation will be cancelled no matter if the reference switch was hit or the referencing sequence was completed
- Referenced Position – the metric position (in unit millimetres) that has to be set at the referenced position

When a movement is running at an axis it can't be overwritten by a new position value immediately, such a new position information would be rejected by the plug-in. In such a case the axis first has to be stopped and after the related speed-output indicates that this axis really has stopped (by emitting a speed of 0) the new position can be set at the POS-input. Within the debugger such an invalid positioning request would be indicated, within the OpenPlayer it is ignored without any further notice.

Numeric IN0 (POS) – the absolute position (in unit mm or degrees) to move the axis 1 to, for this movement the speed is used that was set last at the MODE input; when a value is set here and no other movement operation is in progress at this axis the motor is moved to the related position immediately

Numeric IN1 (MODE) – this input is used for different control commands depending on the value of the number that is set here; when the given value is greater than 0 it is interpreted as speed and the value is used for the next movement command given at the preceding input, when the input value is equal to 0 the motor is stopped immediately (stop-command), a value of -1 causes a movement to the home/referencing

position (here the reference input has to be connected and configured properly)

Numeric IN2 (POS) – performs a movement with axis 2 similar to IN0

Numeric IN3 (MODE) – defines the speed for axis 2 or controls its movement similar to IN1

Numeric IN4 (POS) – performs a movement with axis 3 similar to IN0

Numeric IN5 (MODE) – defines the speed for axis 3 or controls its movement similar to IN1

Binary IN7 (CTRL) – input for 2D (X and Y) or 3D (X, Y and Z) Control data to manage the movement; here the related number of axes has to be configured properly. The related output OUT7 emits motion-synchronous Control data that can be used by a tool for machinery.

Digital OUT6 (BSY) – busy-signal, specifies if at least one axis is still moving; this output is set to LOW as soon as all current operations have been finished by this plug-in

Binary OUT7 (CTRL) – output for the Control data delivered at IN7; here the control data are emitted synchronously dependent on the current movement position so that a tool can be controlled correctly dependent on the current position. This output also pays attention to possible tool on delays or tools off delays, as long as they are smaller than 0 the Control data are emitted the given time before the related movement starts so that negative on/off delays are handled correctly.

This plug-in is located in flowplugins/libio_scanctrl_stepper.

5.3.3.7.2.16 Motion Macros

Schneider/IMS MDrive+ Controller:

Using this macro it is possible to handle Schneider/IMS MDrive and MDrive+ motors easily. Internally it uses the "Schneider/IMS MDrive" plug-in: as a first step it has to be configured, the serial port parameters and the axes device names have to be set properly in order to use the macro.

Numeric IN AX1_POS – this is a command input that lets axis 1 move immediately to the given position using the speed value that is set at input AX1_SPD

Numeric IN AX1_SPD – sets a new speed that is used for the next movement of axis 1

Digital IN AX1_STOP – stops a movement of axis 1 immediately

Digital IN AX1_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric IN AX2_POS - this is a command input that lets axis 2 move immediately to the given position using the speed value that is set at input AX2_SPD

Numeric IN AX2_SPD – sets a new speed that is used for the next movement of axis 2

Digital IN AX2_STOP – stops a movement of axis 2 immediately

Digital IN AX2_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric IN AX3_POS – this is a command input that lets axis 3 move immediately to the given position using the speed value that is set at input AX3_SPD

Numeric IN AX3_SPD – sets a new speed that is used for the next movement of axis 3

Digital IN AX3_STOP – stops a movement of axis 3 immediately

Digital IN AX3_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric OUT AX1_CPOS – the current position of axis 1, during a movement this output returns changing position data permanently

Digital OUT AX1_MOVING – this output is set to HIGH as long as the axis 1 is moving, LOW indicates that the movement of the axis has stopped completely

Numeric OUT AX2_CPOS – the current position of axis 2, during a movement this output returns changing position data permanently

Digital OUT AX2_MOVING – this output is set to HIGH as long as the axis 2 is moving, LOW indicates that the movement of the axis has stopped completely

Numeric OUT AX3_CPOS – the current position of axis 3, during a movement this output returns changing position data permanently

Digital OUT AX3_MOVING – this output is set to HIGH as long as the axis 3 is moving, LOW indicates that the movement of the axis has stopped completely

IWH Robot Controller:

This is a macro that incorporates the IWH Robot Plug In as described above and offers a more comfortable access to its functions via its extended IO possibilities:

Char IN GET_CAS – this is the first part of the two-step command to GET a wafer out of a cassette. Here the station name of the cassette has to be set. Setting this name does not cause any action and a station name can be overridden by any other name as long as no data are set to the next input:

Numeric IN GET_CAS_SL – this is the second part of the two-step command to GET a wafer out of a cassette, here the slot number to fetch the wafer from has to be set. The robot starts its movement as soon as the number is set at this input using the station name of the previous command and the slot number given here

Char IN PUT_CAS – this is the first part of the two-step command to PUT a wafer into a cassette. Here the station name of the cassette has to be set. Setting this name does not cause any action as long as no data are set to the next input:

Numeric IN PUT_CAS_SL – this is the second part of the two-step command to PUT a wafer out of a cassette, here the slot number to put the wafer into has to be set. The robot starts its movement as soon as the number is set at this input using the station name of the previous command and the slot number given here

Char IN GET_STAT – this command can be used to GET a wafer from the default slot of the specified station, the movement of the robot starts immediately after the station name is set at this input

Char IN PUT_STAT – this command can be used to PUT a wafer into the default slot of the specified station, the movement of the robot starts immediately after the station name is set at this input

Digital IN STOP – stop a running movement of the robot immediately; this input can to be used to interrupt a currently active movement and it has to be used when a motion command has to be sent to the robot while an other motion is stile active

Digital IN HOME – a HIGH signal at this input causes the robot to move to its home position; in case of a volatile position encoder position referencing is done too

Char IN SCAN_STAT – this command can be used to scan a station for wafers, the movement of the robot starts immediately after the station name is set at this input and the resulting wafer mapping is submitted at numeric output SCN_MAP

Digi IN VAC_CMD – this input controls the vacuum valve for the chuck, a HIGH signal turns on the vacuum to hold a wafer, a LOW signal turns of the vacuum; the state of this input influences the state of the outputs VAC_CMD (feedback if the vacuum could be enabled/disabled) and VAC_SENS (feedback if the wafer could be gripped in case of enabled vacuum)

Char IN RAW_CMD – using this input any other commands can be sent to the robot directly

Numeric OUT POS_R – this output gives a feedback about the robots position, here the excursion of the robots arm is given in unit 1/1000 inch

Numeric OUT POS_T – this output gives a feedback about the robots position, here the angular position of the robots arm is given in unit 1/10 degrees

Numeric OUT POS_Z – this output gives a feedback about the robots position, here the height of the robots arm is given in unit 1/1000 inch

Digi OUT VAC_CMD – feedback for the current state of the vacuum command; this output gives a HIGH signal in case the vacuum valve of the chuck could be turned on successfully (using input VAC_CMD)

Digi OUT VAC_SENS - feedback for the current state of the vacuum of the chuck, this output gives a HIGH

signal in case the vacuum valve is turned on (input VAC_CMD) and there is a wafer on the chuck which could be gripped successfully

Digi OUT MOVING – this is a state output that tells if the robot is currently moving (HIGH) or not (LOW)

Numeric OUT SCN_MAP – here the result of a wafer scanning operation (robot command “SCN” or input SCAN_STAT) is emitted as a sequence of numbers which belong to the scanned wafer slots and the scanning result

Steppermotor Controller:

Using this macro it is possible to handle a stepper motor that is controlled via step and direction signals given at the parallel port. Internally it uses the "Stepper Motor Driver (Parport)" plug-in: as a very first step there all required IO- and controlling parameters have to be set.

Numeric IN AX1_POS – this is a command input that lets axis 1 move immediately to the given position using the speed value that is set at input AX1_SPD

Numeric IN AX1_SPD – sets a new speed that is used for the next movement of axis 1

Digital IN AX1_STOP – stops a movement of axis 1 immediately

Digital IN AX1_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric IN AX2_POS – this is a command input that lets axis 2 move immediately to the given position using the speed value that is set at input AX2_SPD

Numeric IN AX2_SPD – sets a new speed that is used for the next movement of axis 2

Digital IN AX2_STOP – stops a movement of axis 2 immediately

Digital IN AX2_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric IN AX3_POS – this is a command input that lets axis 3 move immediately to the given position using the speed value that is set at input AX3_SPD

Numeric IN AX3_SPD – sets a new speed that is used for the next movement of axis 3

Digital IN AX3_STOP – stops a movement of axis 3 immediately

Digital IN AX3_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric IN AX4_POS – this is a command input that lets axis 4 move immediately to the given position using the speed value that is set at input AX4_SPD

Numeric IN AX4_SPD – sets a new speed that is used for the next movement of axis 4

Digital IN AX4_STOP – stops a movement of axis 4 immediately

Digital IN AX4_HOME – this input can be used when the axis has stopped, it lets the axis do a reference movement to find its home position

Numeric OUT AX1_CPOS – the current position of axis 1, during a movement this output returns changing position data permanently

Digital OUT AX1_MOVING – this output is set to HIGH as long as the axis 1 is moving, LOW indicates that the movement of the axis has stopped completely

Numeric OUT AX2_CPOS – the current position of axis 2, during a movement this output returns changing

position data permanently

Digital OUT AX2_MOVING – this output is set to HIGH as long as the axis 2 is moving, LOW indicates that the movement of the axis has stopped completely

Numeric OUT AX3_CPOS – the current position of axis 3, during a movement this output returns changing position data permanently

Digital OUT AX3_MOVING – this output is set to HIGH as long as the axis 3 is moving, LOW indicates that

the movement of the axis has stopped completely

Numeric OUT AX4_CPOS – the current position of axis 4, during a movement this output returns changing position data permanently

Digital OUT AX4_MOVING – this output is set to HIGH as long as the axis 4 is moving, LOW indicates that the movement of the axis has stopped completely

5.3.3.7.2.17 Data Flow Objects

There exist no application-internal flow elements of this type, all data related functionalities are provided by external plug-ins.

5.3.3.7.2.18 External Data Flow Objects

Clock:

The Clock plug-in provides several time-related information. So there are outputs that send pulses in definable time periods, outputs that send numerical information and one output that gives a formatted time string.

Numeric IN2 – the Clock plug-in always uses the current time of the hosting system. To work with other times this input can be used to define an offset to the current time. This offset has to be given in seconds and can be positive or negative

Digital OUT0 – this output sends HIGH signals; Within the configuration the delay between two pulses can be defined, it is possible to have one every second, every minute, every hour or every day

Digital OUT1 – this output works similar but independent from OUT0

Numeric OUT2 – here a numerical information is given about the configured part of the time information; within the configuration it can be specified if the current second, minute, hour, day, month, year or Unix-time stamp has to be emitted as numerical value

Numeric OUT3..OUT7 – these outputs works similar to OUT2 but are independent from it and can be configured for a different time portion to be given

Char OUT7 – using this output a formatted time string can be created. The time string itself can consist of normal characters and some additional format information that are place holders for special portions of the time handled by this plug-in; these place holders always start with a "%" sign. To have the real "%" character within the format string, a "%%" has to be entered. Following format information are supported:

- %A – is replaced by national representation of the full weekday name
- %a – is replaced by national representation of the abbreviated weekday name
- %B – is replaced by national representation of the full month name
- %b and %h – are replaced by national representation of the abbreviated month name
- %C – is replaced by the year divided by 100
- %c – is replaced by national representation of time and date
- %d – is replaced by the day of the month as a decimal number (01..31)
- %G – is replaced by a year as a decimal number with century. This year is the one that contains the greater part of the week (Monday as the first day of the week)
- %g – is replaced by the same year as in "%G", but as a decimal number without century (00..99)
- %H – is replaced by the hour (24-hour clock) as a decimal number (00..23)
- %I – is replaced by the hour (12-hour clock) as a decimal number (01..12)
- %j – is replaced by the day of the year as a decimal number (001..366)
- %M – is replaced by the minute as a decimal number (00..59)

- %m – is replaced by the month as a decimal number (01..12)
- %n – is replaced by a newline
- %p – is replaced by national representation of either "ante meridiem" ("AM") or "post meridiem" ("PM") as appropriate
- %r – is equivalent to "%I:%M:%S %p"
- %S – is replaced by the second as a decimal number (00..60)
- %v – is equivalent to "%e-%b-%Y"
- %W – is replaced by the week number of the year (Monday as the first day of the week) as a decimal number (00..53)
- %w – is replaced by the weekday (Sunday as the first day of the week) as a decimal number (0..6)
- %X – is replaced by national representation of the time
- %x – is replaced by national representation of the date
- %Y – is replaced by the year with century as a decimal number
- %y – is replaced by the year without century as a decimal number (00..99)
- %Z – is replaced by the time zone name
- %z – is replaced by the time zone offset from UTC; a leading plus sign stands for east of UTC, a minus sign for west of UTC, hours and minutes follow with two digits each and no delimiter between them
- %+ - is replaced by national representation of the date and time

This plug-in is located in flowplugins/libio_clock.

CSV to Control:

This plug-in converts 2D or 3D movement and tool data that are given in a CSV file to the related binary control data structure. These control data then can be handled by motion elements (like the Schneider/IMS MDrive+ plug-in) and to access tools (like a mill or a laser) to process materials.

The structure of the CSV files is partially fixed, it in every case has to consist of X and Y coordinates which are mandatory. An additional Z value is required only in case a 3D CSV file is read in. This can be used to generate three-dimensional tool data. All other values are optional. In case they do not exist, the related fields of the binary Control structure are set to 0 or to a value that was found within the CSV file preceding. The given additional parameters always apply for the current coordinates. As an example: there is a (new) power value set within a row. This value becomes active before the movement to the given coordinate is started, means the new power is valid for the way to the coordinate that was given in the same line like the new power value.

Following CSV-rows are possible:

X (mandatory) – the X value of the current position coordinate in unit mm

Y (mandatory) – the Y value of the current position coordinate in unit mm

Z (mandatory in case X, Y and Z values have been configured as input data) – the Z value of the current position coordinate in unit mm

ON (optional) – specifies if the tool has to be turned on (1) or off (0) for the movement to the coordinates given in the same line; if no value is specified here the preceding state is used or 0 in case no ON-value was set until now

Power (optional) – the power a connected tool has to be used with, here a value in range 0%..100% is possible; if no value is specified here the preceding state is used or 0% in case no power-value was set within the current file

Frequency (optional) – the frequency in unit Hz a connected tool has to be used with for the movement to the coordinates given in the same line of the CSV file; if no value is specified here the preceding state is used or 0 Hz in case no Frequency-value was set within the current file

Off Speed (optional) – this is the speed a movement has to be performed with when the tool is turned off, it is given in unit mm/sec and has to be specified at least once before a movement with the tool turned off (ON=0) is performed

On Speed (optional) – this is the speed a movement has to be performed with when a used tool is turned on, it is given in unit mm/sec and has to be specified at least once before a movement with the tool turned on (ON=1) is performed

Off Delay (optional) – here a delay value can be specified in unit usec (microseconds) that defines a delay when the tool is turned off: it turns off the tool using the given delay. This delay can be negative, in this case the tool is turned off the given time before the related position is reached.

On Delay (optional) – here a delay value can be specified in unit usec (microseconds) that defines a delay when the tool is turned from off to on: it turns on the tool using the given delay. This delay can be negative, in this case the tool is turned on the given time before the related position is reached.

Additional Parameter 1 (optional) – this is a parameter that can be used freely for some additional values that might be interesting for the movement or the tool, here the data within the CSV file have to fit to the capabilities of the used motion element or tool

Additional Parameter 2 (optional) – this is a parameter that can be used freely for some additional values that might be interesting for the movement or the tool

Additional Parameter 3 (optional) – this is a parameter that can be used freely for some additional values

Additional Parameter 4 (optional) – this is a parameter that can be used freely for some additional values

Additional Parameter 5 (optional) – this is a parameter that can be used freely for some additional values

Additional Parameter 6 (optional) – this is a parameter that can be used freely for some additional values

The rows of such a CSV line have to consist of numbers, as soon as any other character is detected parsing of the current line is stopped dropping all the data (in case the operation failed during parsing of the mandatory coordinates) or it is stopped using the data that could be fetched until this position (in case failure happened during parsing the optional values).

Before using this plug-in following parameters have to be set within the configuration panel:

Default Filename – path to a predefined CSV-file that has to be loaded and converted

Input Data – here it has to be chosen if the CSV-file contains only two rows for 2D (X and Y) coordinates or three rows for 3D (X, Y and Z) coordinates that have to be parsed; in case this value is set accidentially wrong, all following parameters are assigned to the wrong field within the binary control structure, no logic check can be performed here to avoid such a problem.

Column Separator – specifies which character is used within the CSV-file to separate the columns of data

Output Data Structure – this option can be used to specify if the created binary control structure contains 2D or 3D information; this option normally should be set to the same value like the “Input Data” parameter but does not have to, it can be used to perform a simple data conversion between 2D and 3D too.

Output 7 Value – the numeric output 7 of this plug-in is not assigned to a fixed value of the CSV-file, here it can be chosen which of the additional rows have to be used to emit data at OUT7

This conversion plug-in uses the following in- and outputs:

Digital IN0 (CLK) – start loading and parsing of the current CSV file, as soon as a HIGH value is given here the complete file is converted so that the related output emit a continuous stream of data according to the values found within the file

Char IN1 (FILE) – specifies a new CSV file that will be loaded when the next HIGH signal is given to IN0

Char IN2 (CSV) – input for a single line of CSV data that is processed immediately and independent form the CLK input

Binary OUT0 (CTRL) – output for binary Control data that contain movement and tool information according to the values out of the loaded file or according to the values out of the given CSV line; here a stream of data is emitted until all lines of a given file are processed, an exact timing that would result out of the movement speeds and the position is ignored, this has to be handled by the plug-in that is responsible for the motion.

Numeric OUT1 (ERR) – this output emits error codes in case the specified CSV-file could not be loaded.

Digital OUT2 (ON) – whenever the ON-state of the tool changes this output emits a new value LOW or HIGH that signals the new state

Numeric OUT3 (PWR) – whenever the power-value changes this output emits a new value in range 0..100 that signals the new power state

Numeric OUT4 (FREQ) – whenever the frequency-value changes this output emits a new value that signals the new frequency

Numeric OUT5 (OFFSPD) – whenever a new off speed is set this output emits a new value that signals the new speed-value

Numeric OUT6 (ONSPD) – whenever a new on speed is set this output emits a new value that signals the new speed-value

Numeric OUT6 (PRM) – here a new value is emitted whenever the parameter changes that was configured for this output within the settings panel of the plug-in

This plug-in is located in flowplugins/libio_csv2ctrl

E-Mail Notification:

When an event occurs on a system where no user sits in front of it to take notice of it, this plug-in can be used to send the related information via e-mail. It takes dynamically or statically data and submits it via the network.

Within the configuration panel following parameters have to be defined:

- Mailserver - the IP or host name of the server that has to deliver the mail; this server has to be configured to accept incoming connections and incoming e-mails from the host where this project is running at
- To - the recipient of the mail (the recipients mail address)
- Subject - the subject of the mail
- Text - the body of the mail

Subject and body can be overwritten and replaced by dynamic data:

Digital IN0 (SEND) – sends an e-mail using the current data

Char IN1 (SUB) – overwrites the predefined subject with this text

Char IN2 (BDY) – overwrites the predefined text with this new one

This plug-in is located in flowplugins/libio_mailnotifer.

HPGL to Control:

This plug-in converts 2D vector data that are given in a HPGL (.PLT) file to the related binary control data structure. These control data then can be handled by motion elements (like the Schneider/IMS MDrive+ plug-in) and to access tools (like a mill or a laser) to process materials. The plug-in supports the HPGL 1 standard completely plus some parts of the HPGL 2 format specification.

Since HPGL does contain pen-information but no data that could be used somehow to manage a tool or that define movement information these data have to be configured within the plug-ins configuration panel.

Within the general plug-in settings a default file name to a .PLT file has to be set which is used for conversion. Since there are no measurement units specified within the file itself the “HPGL Unit” defines the size of one unit within the file. Depending of the file version and the accuracy of the exporting application this can be 0.025 mm, 0.01 mm or any other value.

Next for every pen that may exist within the HPGL file movement and tool parameters have to be set. These parameters are used for every vector information of the related pen. Thus different pens can be used to define different speed and/or tool values. Following pen parameters are required:

Power – the power the tool has to be handled with in range 0%..100%

Frequency – the frequency in unit Hz the tool has to be controlled with (tool-dependent parameter)

Off Speed – the speed that is used for all movements where the tool is turned off, means where the pen is defined as “up” within the HPGL file and where no visible vector data are defined within it

On Speed – the speed that is used for all movements where the tool is turned on, means where the pen is defined as “down” within the HPGL file and where visible vector data are defined within it

Off Delay – the delay that becomes active when the tool is turned off, this delay will be done by later processing steps between start of the movement and turning the tool off really, this value might be negative

On Delay – the delay that becomes active when the tool is turned on, this delay will be done by later processing steps between start of the movement and turning the tool on really, this value might be negative

Additional Parameter 1..6 – here some additional, tool-specific values can be set

The plug-in supports the following IO's:

Digital IN0 (CLK) – start loading and parsing of the current HPGL file, as soon as a HIGH value is given here the complete file is converted so that the related output emit a continuous stream of data according to the values found within the file

Char IN1 (FILE) – specifies a new HPGL file that will be loaded when the next HIGH signal is given to IN0

Binary OUT0 (CTRL) – output for binary Control data that contain movement and tool information according to the values out of the loaded file and the pen-definitions done within the plug-ins configuration; here a stream of data is emitted until all vector data of a given file are processed, a timing that would results out of the movement speeds and the position is ignored, this has to be handled by the plug-in that is responsible for the motion.

Numeric OUT1 (ERR) – this output emits error codes in case the specified HPGL-file could not be loaded.

This plug-in is located in flowplugins/libio_hpgl2ctrl

HTTP Client:

Using this plug-in data can be retrieved via the network using the HTTP-protocol. After HTTP is a stateless and connectionless protocol, every attempt to retrieve data results in a new connection to the specified server. Thus a connection problem can't be detected during application start-up when the plug-in is initialised but only during runtime when new data are requested.

Within the configuration settings a default IP, a port number and a resource where the data have to be retrieved from can be specified. IP is something in style xxx.xxx.xxx.xxx or a domain name like openapc.com. Port typically is 80 and the resource is the exact document that has to be fetched (e.g. "/index.php").

Currently this plug-in supports following mime types:

- image/* - loads an image and converts it to a raw binary bitmap
- text/plain - loads a plain text

Following IO's are supported by this plug-in:

Char IN0 (IP) – overwrites the default IP or host name

Numeric IN1 (PORT) – overwrites the default port number specified within the set-up

Char IN2 (RES) – overwrites the default resource information and tries to get it immediately

Digital IN3 (GET) – when this input is set to 1

Binary OUT0 – when some data like images have been downloaded, they are emitted at this output

Char OUT1 – after downloading a plain text file the resulting lines of text are emitted at this output line by line

Numeric OUT2 (ERR) – this output returns an result code; here positive numbers are equal to the HTTP error codes (200 means everything is fine), negative numbers point to internal reasons for a failure,all other codes not equal to 200 point to an other kind of (HTTP) error.

This plug-in is located in flowplugins/libio_http_client.

Load Image:

This plug-in is able to load a picture and to provide the loaded image as binary data. These binary data can be used e.g. to dynamically change images within the user interface. Different HMI elements support that functionality (like the Image Button or the Static Image). When the binary data loaded by this plug-in are sent to a flow element that is not able to handle images (because it expects a different kind of binary data), the image data are dropped. In this case within the OpenDebugger an error message is given, when the project is executed in OpenPlayer the user is not informed about this illegal operation.

The configuration panel of the plug-in gives the possibility to pre-define up to four image files. Please note: when the project is moved to a different platform or system it has to be made sure that the paths given for the images are still valid!

Digital IN0 – loads the image that is predefined with image file 1

Char IN1 – overwrites the current predefinition of image file 1 to be loaded and loads the new one immediately

Digital IN2 – loads the image that is predefined with image file 2

Char IN3 – overwrites the current predefinition of image file 2 to be loaded and loads the new one immediately

Digital IN4 – loads the image that is predefined with image file 3

Char IN5 – overwrites the current predefinition of image file 3 to be loaded and loads the new one immediately

Digital IN6 – loads the image that is predefined with image file 4

Char IN7 – overwrites the current predefinition of image file 4 to be loaded and loads the new one immediately

Binary OUT0 – the loaded image data of file 1

Numeric OUT1 – when loading of the image 1 failed an error code is given here

Binary OUT2 – the loaded image data of file 2

Numeric OUT3 – when loading of the image 2 failed an error code is given here

Binary OUT4 – the loaded image data of file 3

Numeric OUT5 – when loading of the image 3 failed an error code is given here

Binary OUT6 – the loaded image data of file 4

Numeric OUT7 – when loading of the image 4 failed an error code is given here

This plug-in is located in flowplugins/libio_loadimg.

Load Text:

This plug-in is able to load a text file line by line. Comparing to similar PlugIns here a HIGH signal at the digital input does NOT cause the complete file to be loaded but it reads the next line out of it. The configuration panel of the plug-in gives the possibility to pre-define up to four text files where the contents have to be loaded.
PLEASE NOTE: when the project is moved to a different platform or system it has to be made sure that the paths given for the text files are still valid, possibly these paths have to be adapted due to different file name conventions and drive naming restrictions!

Digital IN0 (CLK1) – loads the next line of the text that is predefined with file 1

Char IN1 (FILE1) – overwrites the current predefinition of text file 1; this operation stops reading the previous file and opens the new one; the first line of this file is read only when a HIGH signal is received at IN0

Digital IN2 (CLK2) – loads the next line of the text that is predefined with file 2

Char IN3 (FILE2) – overwrites the current predefinition of text file 2; this operation stops reading the previous file and opens the new one; the first line of this file is read only when a HIGH signal is received at IN2

Digital IN4 (CLK3) – loads the next line of the text that is predefined with file 3

Char IN5 (FILE3) – overwrites the current predefinition of text file 3; this operation stops reading the previous file and opens the new one; the first line of this file is read only when a HIGH signal is received at IN4

Digital IN6 (CLK4) – loads the next line of the text that is predefined with file 4

Char IN7 (FILE4) – overwrites the current predefinition of text file 4; this operation stops reading the previous file and opens the new one; the first line of this file is read only when a HIGH signal is received at IN6

Char OUT0 (DATA1) – the next text line out of file 1

Numeric OUT1 (ERR1) – when the end of the file 1 was reached or when an error occurred no text is emitted but the related error code is given here; following error codes are used:

- 1 - the file could not be opened
- 3 - the file could not be read
- 4 - the end of the file was reached, no more data are available
- 5 - no file name was specified

Char OUT2 (DATA2) – the next text line out of file 2

Numeric OUT3 (ERR2) – when the end of the file 2 was reached or when an error occurred no text is emitted but the related error code is given here

Char OUT4 (DATA3) – the next text line out of file 3

Numeric OUT5 (ERR3) – when the end of the file 3 was reached or when an error occurred no text is emitted but the related error code is given here

Char OUT6 (DATA4) – the next text line out of file 4

Numeric OUT7 (ERR4) – when the end of the file 4 was reached or when an error occurred no text is emitted but the related error code is given here

This plug-in is located in flowplugins/libio_loadtxt.

MySQL Access:

With this plug-in it is possible to communicate with a MySQL database server, to send data to a database and to retrieve data out of it. Within the configuration panel the IP and port number of the server, the user name and password and the name of the database have to be specified.

Char IN0 (QUER) – this input has to be used for SQL-statements where no response is expected (statements that set data like "INSERT" or "UPDATE")

Char IN1 (REQ) – this input has to be used for SQL-statements that may cause a response (statements that request data like "SELECT")

Char OUT1 (RES) – the result of a request to input 1; the resulting columns of a row are returned as CSV (comma separated values), when more than row is returned every row is emitted with a separate text string at this output

This plug-in is located in flowplugins/libio_mysql.

PostgreSQL Access:

With this plug-in it is possible to communicate with a PostgreSQL database server, to send data to a database and to retrieve data out of it. Within the configuration panel the IP and port number of the server, the user name and password and the name of the database have to be specified.

Char IN0 (QUER) – this input has to be used for SQL-statements where no response is expected (statements that only set or modify data like "INSERT" or "UPDATE")

Char IN1 (REQ) – this input has to be used for SQL-statements that may cause a response (statements that request data like "SELECT")

Char OUT1 (RES) – the result of a request to input 1; the resulting columns of a row are returned as CSV (comma separated values), when more than row is returned every row is emitted with a separate text string

at this output

This plug-in is located in flowplugins/libio_postgresql.

Random Generator:

This plug-in creates random data of type digital and numeric. Whenever the related digital input is triggered the assigned output emits a random signal. The digital output signal is randomly set to LOW or HIGH. The valid range of the numeric output can be configured within the configuration panel, there the maximum value can be specified.

Digital IN0 – trigger signal to emit a new digital output value at OUT0

Digital IN1 – trigger signal to emit a new numeric output value at OUT1 in configured range from 0 to the parameter set in plug-in configuration

Save Image:

Using this plug-in binary image data can be saved to disk as image file. Such an image file can be opened by other applications later. This plug-in supports different common file formats to save the image with. Depending on the kind of image data and depending on the intended usage of the pictures an appropriate image format has to be chosen. This plug-in is able to handle up to four independent parameter sets for images that can be set within the configuration panel separately:

- Default Filename – the default file name that has to be used to create the picture file
- File Format – the file format of the picture, here one of the formats PNG, GIF, PNM, XBM, XPM, BMP or JPEG can be chosen

PLEASE NOTE: when the project is moved to a different platform or system it has to be made sure that the default paths given for the images are still valid according to the naming conventions of the target system!

Binary IN0 (IMG1) – the binary image data that have to be saved to disk; the file is written using the current default file name 1 as soon as binary data arrive at this input

Char IN1 (FILE1) – overwrites the default file name 1

Binary IN2 (IMG2) – the binary image data that have to be saved to disk; the file is written using the current default file name 2 as soon as binary data arrive at this input

Char IN3 (FILE2) – overwrites the default file name 2

Binary IN4 (IMG3) – the binary image data that have to be saved to disk; the file is written using the current default file name 3 as soon as binary data arrive at this input

Char IN5 (FILE3) – overwrites the default file name 3

Binary IN6 (IMG4) – the binary image data that have to be saved to disk; the file is written using the current default file name 3 as soon as binary data arrive at this input

Char IN7 (FILE4) – overwrites the default file name 4

This plug-in is located in flowplugins/libio_saveimg.

Save Text:

This plug-in behaves a bit different than the other file operations. Here the data are not written once, instead of it this plug-in opens a file and writes all text data into this file line by line as soon as it gets them.

The "Save Text" plug-in can handle up to four independent streams that save data into four separate files. Within the global configuration panel default names for these four files can be set.

This plug-in can be used e.g. to save logging information to disk as they are created by the application-internal log functionalities.

PLEASE NOTE: when the project is moved to a different platform or system it has to be made sure that

the default paths given for the text files are still valid!

Char IN0 (TXT1) – whenever a new text is send to this input a new line is written into the corresponding file 1

Char IN1 (FILE1) – here the default file name 1 can be changed; setting a new file name closes the preceding text file and creates a new one using this name, when this new name is equal to the preceding one the old text file is overwritten

Char IN2 (TXT2) – whenever a new text is send to this input a new line is written into the corresponding file 2

Char IN3 (FILE2) – here the default file name 2 can be changed; setting a new file name closes the preceding text file and creates a new one using this name, when this new name is equal to the preceding one the old text file is overwritten

Char IN4 (TXT3) – whenever a new text is send to this input a new line is written into the corresponding file 3

Char IN5 (FILE3) – here the default file name 3 can be changed; setting a new file name closes the preceding text file and creates a new one using this name, when this new name is equal to the preceding one the old text file is overwritten

Char IN6 (TXT4) – whenever a new text is send to this input a new line is written into the corresponding file 4

Char IN7 (FILE4) – here the default file name 4 can be changed; setting a new file name closes the preceding text file and creates a new one using this name, when this new name is equal to the preceding one the old text file is overwritten

This plug-in is located in flowplugins/libio_savetxt.

5.3.3.7.2.19 Miscellaneous Flow Objects

Interlock Server Connection:

Instead of implicit mapping of outputs to the Interlock Server and instead of allowing the Interlock Server to modify the inputs of a local element directly, this flow element offers an alternative, very specific possibility to access data nodes. Normally it should not be necessary to use this direct communication path but in some seldom cases it might be useful – especially when direct communication is necessary with plug-ins that are handled by the external OpenPlugger.
It offers eight inputs and outputs of different data types that can be assigned to data nodes within the Interlock Server. Whenever data are sent to an input of this flow element these data are forwarded to the configured node and its input. Whenever a configured node changes within the Interlock Server at a output that is set up within the configuration panel of this flow element, the changed data are emitted at the related output.
So within the configuration panel for every input and output two things have to be configured: the unique name of the node within the Interlock Server the flow element has to connected with and the number of the input or output of that node to assign the IO of the flow element with.
When an input field for a node name is left empty within the configuration dialogue, this input/output is unused and will not affect anything/will not be affected by anything.
PLEASE NOTE: This flow element is not able to modify or to receive data from nodes that belong to an other element within the same OpenPlayer, it only can access nodes that exist within the Interlock Server but it can't access nodes that already belong to mapped elements within the player.

Digi IN0 – input for digital data to be forwarded to a node and its input number as configured

Digi IN1 – input for digital data to be forwarded to a node and its input number as configured

Numeric IN2 – input for numerical data to be forwarded to a node and its input number as configured

Numeric IN3 – input for numerical data to be forwarded to a node and its input number as configured

Char IN4 – input for text data to be forwarded to a node and its input number as configured

Char IN5 – input for text data to be forwarded to a node and its input number as configured

Binary IN6 – input for text data to be forwarded to a node and its input number as configured

Binary IN7 – input for text data to be forwarded to a node and its input number as configured

Digi OUT0 – output that is mapped to a node within the Interlock Server and that emits a digital value whenever the configured output of this node changes

Digi OUT1 – output that is mapped to a node within the Interlock Server and that emits a digital value

whenever the configured output of this node changes

Numeric OUT2 – output that is mapped to a node within the Interlock Server and that emits a numerical value whenever the configured output of this node changes

Numeric OUT3 – output that is mapped to a node within the Interlock Server and that emits a numerical value whenever the configured output of this node changes

Char OUT4 – output that is mapped to a node within the Interlock Server and that emits a text whenever the configured output of this node changes

Char OUT5 – output that is mapped to a node within the Interlock Server and that emits a text whenever the configured output of this node changes

Binary OUT6 – output that is mapped to a node within the Interlock Server and that emits binary data whenever the configured output of this node changes

Binary OUT7 – output that is mapped to a node within the Interlock Server and that emits binary data whenever the configured output of this node changes

Log Output:

This is a very specific flow object. Different to the other objects it is allowed to exists exactly once within an project. When it exists, it receives all logging information that may be generated by other elements. When there are logging events defined in other elements but no Log Output object exists, these logging information are dropped. When a Log Output element exists and logging information are generated, it is essential that all outputs of this element that belong to used log types are used. When a output of this object is unused but the related log is used within the application that may lead to a memory overflow (or at least to excessive but useless memory usage) because the logged data are not flushed out of this flow element. Beside a logging information that is given as plain, human readable text also a unique identifier is given. This identifier specifies the element that has issued the logging event. The identifier is a number that can be found within the property dialogue of such an element (for an HMI element right beside the name input field, for an Flow element within the dialogue title)..

Char OUT0 – output for the error log texts that are generated by other objects

Numeric OUT1 – output for the unique identifier of the object that has caused the error log event

Char OUT2 – output for the warning log texts that are generated by other objects

Numeric OUT3 – output for the unique identifier of the object that has caused the warning log event

Char OUT4 – output for the information log texts that are generated by other objects

Numeric OUT5 – output for the unique identifier of the object that has caused the information log event

Char OUT6 – output for the event log texts that are generated by other objects

Numeric OUT7 – output for the unique identifier of the object that has caused the additional log event

Log Recorder:

Most of the HMI objects offer the possibility to watch their data directly and to cause log events if necessary. These elements can create log events directly if required. When data have to be watched within a more complex flow structure or where no elements are involved that are able to generate log data for its own, the Log Recorder flow element can be used. It can be attached to data flows to watch their values and to create log information if necessary. There can exist several Log Recorders within the same project, all of them send their data to the same Log Output object like it is used by the HMI objects. Within the definition dialogue – that is very similar to the one known from the HMI objects – of this flow object it can be defined which values or ranges of values have to cause logging events for which type of log.

Digital IN0, digital IN1 – inputs to watch digital data

Numerical IN2, numerical IN3 – inputs to watch numerical data

Char IN4, char IN5 – inputs to watch text data

Log-In User:

This element belongs to the internal user management functionality and can be used only when this feature is enabled and used. Using this flow element a users login-name and password can be set. When both are correct and belong to an active user all user interface elements of the current project are set to a state that belongs to the privileges of this user. So this element is mandatory when user management functionality has to be implemented for a project. As a minimum configuration a text fields data flow and a password fields data flow have to be connected with this element so that a user is able to enter a login-name and a password initially. Then the comparison of these user data and the modification of the state of all HMI elements is done by this flow element automatically.
For some more details please refer to the section “User Management” below.

Char IN0 (USR) – character input for the login name of a user; sending data to this input does not invoke a log-in operation, this happens only when a password is given

Char IN1 (PWD) – character input for the password of a user, this input should be connected with a Password Field in order to hide the password while it is entered; as soon as new data are given here the log-in operation starts automatically: when both, the log-in name from IN0 and the password belong to a valid user this user is logged in and all HMI elements of the current project are set to a state that belongs to the privileges granted to this user

Digi IN2 (CHG) – opens a password change dialogue for the currently logged in user so that he is able to set a new password for his account

Digi IN3 (OUT) – logs out the current user and sets all HMI elements to their “default” state

Char OUT0 (Name) – as soon as a user was logged in successfully this output gives the full name of this user as configured

Digi OUT1 (OK) – this output signalises if a user log-in attempt could be done successfully or not; here a HIGH signal is emitted as soon as a valid user logged in using the correct password, in al other cases (user not valid, user disabled, password not correct,...) the output emits a LOW signal. Only in case a HIGH is given logging in was successful and only in this case all the HMI elements of the current project could be updated to a new state.

5.3.3.7.2.20 External Miscellaneous Flow Objects

Execute Program:

With this plug-in an external program can be started. Within the configuration panel of the plug-in up to four applications can be predefined including their command line parameters. Please note: after the programs, that can be defined here, depend on the underlying platform, it might be necessary to modify a project manually before it is moved to a different system. So when a project was created e.g. on a Windows system and is set up to execute a specific .EXE file there, the path and the name of this program have to be adapted before the project is put e.g. to Linux (which does not know .EXE files). That's necessary because the application paths and program names are different on these systems.

The plug-in can handle up to four applications using the four input possibilities of it. Nevertheless it is not able to start an application for one of the inputs where the preceding application is already running, in this case the second call to start the program will be rejected. To avoid such collisions applications have to be used that run in background automatically, they have to be configured to go in background or it has to be avoided that it is started while the previous program is still active.

Within the debugger such a collision is indicated, within the OpenPlayer the execution of the second application isn't done without any further notification to the end user.

Digital IN0 - executes the predefined application 1 when a HIGH signal is set

Char IN1 - overwrites the name and parameters of the predefined application 1 and executes the new one

Digital IN2 - executes the predefined application 2 when a HIGH signal is set

Char IN3 - overwrites the name and parameters of the predefined application 2 and executes the new one

Digital IN4 - executes the predefined application 3 when a HIGH signal is set

Char IN5 - overwrites the name and parameters of the predefined application 3 and executes the new one

Digital IN6 - executes the predefined application 4 when a HIGH signal is set

Char IN7 - overwrites the name and parameters of the predefined application 4 and executes the new one

Numeric OUT1 - gives the return value of application 1 as soon as it exits

Numeric OUT3 - gives the return value of application 2 as soon as it exits

Numeric OUT5 - gives the return value of application 3 as soon as it exits

Numeric OUT7 - gives the return value of application 4 as soon as it exits

This plug-in is located in flowplugins/libio_execute.

5.3.4 The Plugged Devices List of the OpenEditor

Within the Flow Editor it is possible to add plug-ins that access external devices and communicate with them. This direct communication where no separation is done between the logical layers of an application is not recommended for larger projects, or for projects that need to be future-proof.

Instead of this direct communication it is recommended to execute such plug-ins separately within an own logical layer and to communicate with them via the Interlock Server. To do that first of all the Interlock Server needs to be activated in global configuration. Then a list becomes active in tab-pane “Plugged Devices” where plug-ins can be configured for external use.

The usage of this list is very similar to the Flow Editor: right-clicking into it opens a context menu where plug-ins can be added. Here only these flow categories are available where it makes sense to let the related plug-ins run externally. When such a plug-in is selected, it will be added to the list. Now the symbol of this plug-in is shown, its unique name, an unique number and the category it belongs to. The number can't be changed, it is set by the editor. This number is important when the project file is opened using the OpenPlugger, it is used to identify the plug-in that has to be used by the OpenPlugger.

An alternative way of adding plugged flow elements is provided by the fold bar on the right hand side of the list. It provides the same categories of elements than the context menu described above. These categories can be opened by double-clicking the bar titles. Then the list of flow elements is shown where one of it can be appended to the list by double-clicking it. Comparing to the fold bars of HMI Editor and Flow Editor this one does not select the elements, here a single click does not cause any action while a double-click adds the element to the list immediately.

The elements in the list can be selected by single clicking them with the left mouse button. A selected element can be deleted via the context menu. When such a list element is double-clicked the configuration dialogue of the plug-in is opened where its element-specific parameters can be set and changed. For a description of all plug-ins, their purpose, usage and parameters please refer to section “The Flow Objects of the Flow Editor” subsections “External Flow Objects”.

Different to the Flow Editor it is not possible to define any flow connections within this list. Flow objects that are added here will run within an own OpenPlugger instance each and will communicate via the Interlock Server only. Their input data are sent from that server to the OpenPlugger/plug-in and output data are sent from the plug-in/OpenPlugger back to the server. So the player or debugger can communicate with these plug-ins only via this Interlock Server.

5.4 Using the OpenDebugger

The OpenDebugger is a specific execution environment for an ControlRoom project. Similar to the OpenPlayer here a project can be loaded and executed. But different to it within the OpenDebugger there are several possibilities to analyse the flow of data detailed in order to find problems within the configuration:

- conditions can be defined at which the program flow has to be stopped
- the flow can be executed in single step mode where the user has to press a button for every data transfer cycle
- errors and warnings are displayed together with the name of the flow element that caused this warning; that might be important for the user to find out where a program flow does not act as expected

- the outputs of flow elements can be watched to see the changes and values of data

The OpenDebugger itself has to be started with a project file name as parameter – when it is executed out of the OpenEditor this is done automatically using the current project. Then it opens two windows: the HMI interface window as it would appear on the target system when the project file is executed with the OpenPlayer and an additional debugging window. This debugging window is opened right beside the HMI window, so make sure that there is enough space on your screen.

5.4.1 Debugging a Project

All the debugging possibilities and functionalities can be reached via the tool bar of the debuggers main window. The first two buttons can be used to open an other project file or to exit the debugger – so the last one brings the user back to the OpenEditor when the debugger is started from there.

The next button – that symbolises a small bug in its original sense – can be used for starting a program flow in debug mode. This behaviour is different to the OpenPlayer that starts a flow immediately after loading a project, in OpenDebugger that has to be done manually. That's necessary in case the user wants to change the debugging settings before the project is executed.
When a flow is started in debugging mode and the check box “Stop on warning” is selected, the program flow will be stopped automatically whenever a warning condition is met (please see below).

The next tool bar button opens the watch window that can be used to inspect the output values of flow elements, it is symbolised by a magnifier. Within that watch window flow elements can be added to display their output states.

Right beside the watch window button the functionality for performing a singe step can be found. This button is symbolised by a foot. Whenever the debugger is in run mode “stopped” and this button is pressed, one single operation is performed. Here “single operation” means

1. the outputs of all flow elements where a data flow is active are read
2. the data read from there are transferred to the input of the next connected flow object(s) and
3. they are set there at the output.

That's true for all flow elements that currently have to handle some data, so a single step performs this operation for all available flow elements in parallel.
PLEASE NOTE: in singe stepping mode all time-dependent flow elements may behave not in the same way than in continuous execution mode. This happens because total execution time in single stepping mode is much longer than in normal operation mode, thus internal delays and timers will finish/become active at a completely different state of the total flow environment. As an example: imagine a timer that emits pulses/data every 500 milliseconds. In normal execution mode these 500 milliseconds are enough to perform several complex operations and calculations so that for the internal processing such an event appears once for every several hundred to several thousand total flow cycles. In single-stepping mode this is quite different: here the time between every manually triggered flow cycle may exceed the 500 milliseconds of your timer. Resulting from that here the timer emits pulses/data now for every flow cycle! Resulting from that the complete project may behave different.

The next button within the tool bar can be used to configure the debugger and to define what conditions and warnings should cause a stop of the program flow. This can be done within a separate configuration dialogue where these options can be selected or unchecked.

The last two check boxes within the tool bar are used to fully disable or enable the stop of the program flow in case of a stop condition and to display the names of the HMI elements within the HMI window – that last option can be important to find a specific HMI element.

Below of this tool bar a panel can be found where the current running state and all warnings that occurred are listed. If a condition is met that causes such a warning the name of the flow or HMI element is given where this happened. Using these information it can be analysed in detail what happens where and which data are sent.

5.4.1.1 Watching the Number of Program Flows

At the lower end of the window some additional information are displayed. Here the current run state can be seen and the number of program flows that are currently active are given. The latter information is an important one: normally when no flows are started cyclically and when no user interaction took place this value should go down to 0. If it stays at a value >0 or if the value grows continuously an illegal loop is set up with the current flow. Such loops should be avoided in every case:

- when it is a loop that runs continuously it consumes computing power and slows down the complete application and the underlying system; on a possibly week target system this may result in an unusable application
- when the number of program flows grows dependent on user interaction or automatically without external input the application will consume all system resources after some time; such a problem is very easy to recognise when it happens fast but when it builds up slowly the application may become unusable only after a longer time – a typical reason for strange and unexplainable reboots that are necessary only after some days

So this information within the OpenDebugger is a very important one, you have to assure that this value goes down to 0 after some time or that it stays at constant, low values in case cyclic, event or timer triggered program flows are used.

As stated above there are some automatic mechanisms implemented within the application to avoid conditions that would slow down or stop the system in a way described above. First of all every program flow internally uses an unique identifier. When such a program flow reaches the input of a flow object where it fetched the output before, that will be detected by using this identifier and the flow will be terminated automatically after the input value is set. So such direct loops are suppressed automatically and a warning is shown within the debugger window when that happens. After that method will not work when two or more looped program flows access the same flow object interleaved this mechanism is helpful but not 100% effective. This means the number of active program flows still have to be checked.

Second within the project settings a time out value can be defined. This time out value is the total time a program flow can run at maximum, if it could continue to carry data to an other flow element after that time has elapsed, it will be killed automatically. Here also a warning is shown within the debugger window. This mechanism is a bit more tricky than the preceding one: after the time a flow element is handled by the application may differ slightly every time and after a project may run slightly slower within the OpenDebugger, no exact forecast can be made, when and where a program flow will be terminated due to that time out. That means the related information within the debugger window only gives a rough idea that there could be a problem and where it may appear, it is not an exact information to deal with. So here it is the best way to modify the program flow definition in a way so that such time outs do not appear any longer.

5.4.1.2 Watching the Outputs of Flow Elements

Within the watch window flow elements can be listed to inspect their outputs. The window consists of a table with eight columns that are equal of the maximum of 8 regular outputs of an flow elements. The flow elements itself are listed as rows. To add a new column the last row (named as “Add new element”) has to be double-clicked. Now a windows opens where all flow elements of this projects are listed, here the flow element to be watched can be chosen.

Now this element is shown in a new row within the watch table. Outputs that are not supported by the selected flow element are displayed in grey, all other outputs are displayed using the colour that represents their data type (digital, numeric, character or binary).

When a flow element is added to the watch window newly, the fields of the outputs are empty. Only after the flow was started and only after one of the outputs changes its state during operation, the new value is displayed within the table:

- digital values show their state “HIGH” or “LOW”, clicking into the related cell changes this state
- numerical values are displayed as floating point numbers, they can be edited within the cell directly
- character values are shown as printable text, they can be edited within the cell directly
- binary values are displayed as a hexadecimal number that represents the address where the binary data are stored at in memory, this value is only informational and can't be changed

Changing the value of an output causes a new data flow using the new value. That flow is emitted during the next execution cycle.

To remove a flow element out of this watch table the related row has to be double-clicked.

5.5 Using the OpenPlayer

The OpenPlayer is the execution environment for an ControlRoom project. This application is intended to be used on the target system. The player is as small and as efficient as possible and therefore doesn't offers any special service, usage or debugging functionalities. It has to be executed with the project file name (and path) given as parameter. Here it is recommended to hand over the full, absolute pathname always, especially in cases where the OpenPlayer itself invokes OpenPlugger-instances that is necessary so that the OpenPlugger can find the project file. The OpenPlayer loads this project file, then initializes all plug-ins used for it and executes the flows within that project it immediately. The player itself does not offer any possibility to stop the application, so if that is required within the loaded project an “application end” condition has to be defined or the player application has to be killed using the functionalities of the underlying operation system (TaskManager, kill-command, ...).

Within the ControlRoom software package no specific mechanism is provided to automatically execute the OpenPlayer together with a project file. That has to be done manually once the project works properly and the system has to be set up for the end user. From the OpenPlayer's point of view this configuration is simple: the one and only parameter required by the OpenPlayer is the full path to the project file that has to be loaded and executed. So dependent on the operating systems capabilities a call that starts the OpenPlayer using that parameter has to be added at a position that is intended to be used for auto-starting applications after the system is booted. This of course should be done after all elements are loaded that are required to display an application to the end user.

For Windows operating systems it is recommended to execute the OpenPlayer exclusively instead of the standard Windows Explorer (please refer to MSDN for details about how to do that). This locks the operating system for the user completely and no start menu, task bar or desktop popup-menu is shown where a user could change anything harmful on the computers system. Access to the operating system can be given via the “Execute Command” plug-in, here an instance of the Windows Explorer can be started. This of course should be allowed only for higher level users and not for standard operators.

For Linux operating system the situation is similar, here a window manager should be chosen where no access to the operating system is given and where OpenPlayer can be started automatically and as one and only application. Operating system access can be given the same way, via the command execution plug-in that e.g. could open a terminal or something other useful.

5.5.1 OpenPlayer Command Line Options

The OpenPlayer can be executed from command line or from within a script using the following syntax and parameters:

```
OpenPlayer.exe [-b] [-x xpos] [-y ypos] <projectname>
```

or (for non-Windows operating systems):

```
OpenPlayer [-b] [-x xpos] [-y ypos] <projectname>
```

- `OpenPlayer` or `OpenPlayer.exe` is the applications name that has to be used always
- `-b` is an optional parameter that decorates the OpenPlayers window with a border so that the user can grab it and change its position by dragging it
- `-x` is an optional parameter that can be used to specify its X-position which is normally 0 (left side of the main screen); this parameter has to be followed by the custom X position in unit pixels

- `-y` is an optional parameter that can be used to specify its Y-position which is normally 0 (left side of the main screen); this parameter has to be followed by the custom Y position in unit pixels
- this value is mandatory and is the full path to the ControlRoom project file (.apcp-file) that has to be executed

5.6 Using the OpenHPlayer

The OpenHPlayer is a special execution environment for a ControlRoom project that can be used on the target system instead of the standard OpenPlayer. This specific player variant has to be executed with the project file name (and path) given as parameter too but it is not able to display a window or any HMI elements. When a project contains HMI elements they are ignored. Beside this the OpenHPlayer behaves exactly as the standard OpenPlayer, it starts the flows contained within a project automatically.

Within the ControlRoom software package no specific mechanism is provided to automatically execute the OpenHPlayer together with a project file. That has to be done manually once the project works properly and the system has to be set up for the end user. From the OpenHPlayer's point of view this configuration is simple: the one and only parameter required by the OpenHPlayer is the full path to the project file that has to be loaded and executed. So dependent on the operating systems capabilities a call that starts the OpenHPlayer using that parameter has to be added at a position that is intended to be used for auto-starting applications after the system is booted. This of course should be done after all elements are loaded that are required to display an application to the end user.

5.7 Using the Interlock Server

This part of the software package is some kind of server that runs in background completely. It doesn't offers a graphical user interface and can be accessed only indirectly. The "I" within the applications name “OpenIServer” points to one possible use cases of this server, it is an abbreviation for "interlock". But more than that, the OpenIServer can be used not only to implement software interlocks, it is also an interface to access a complete project from outside, to dock it to other applications, to implement own software that takes control over the project, that watches the states of some elements, modifies the state of other elements and takes over complete control to realise automatic sequences.

The OpenIServer can be accessed via following ways:

- from within the main application (OpenDebugger or OpenPlayer) where elements are configured to send their output states to that server and/or to let their inputs be influenced from data changes within the server
- from within an own application that makes use of liboapc and the related functions to access the data (please refer to the SDK and the contained manual for a full description of these functions)
- via a LUA or Instruction List script

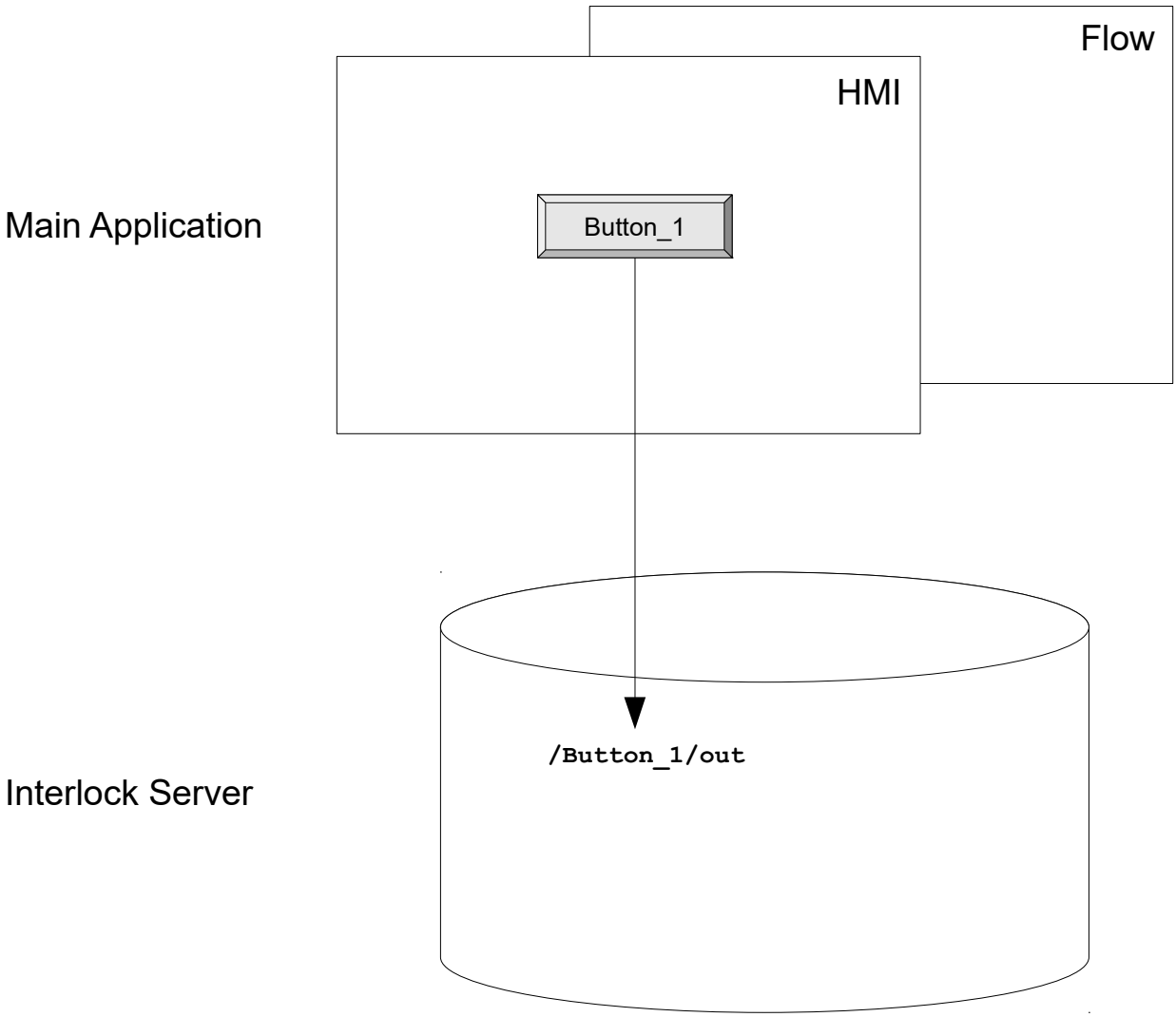
The task that is done by the OpenIServer is quite simple and can be summarised easily:

1. Values are stored using a unique identifier (a node name that is equal to the name of the user interface element it may correspond with, or any other, unique name)
2. Whenever new data are sent to the server using a new identifier these new data are stored and all connected clients are informed
3. Whenever new data are sent to the server using an already known identifier these new data overwrite the already stored ones and all connected client are informed about these changes

Beside of that following is true:

- every data set that consist of a identifier and some data is organized in the same way like an element within the main application: it can have up to 8 separate data values consisting of one of the data types DIGI, NUM, CHAR, BIN
- changing of the data type is not possible, once it was set to one of the allowed types it accepts only data of the same type

- an existing set of data can't be deleted



At a first sight this structure and the tasks the interlock server performs looks a bit useless. But this simple method of storing data gives a wide range of possibilities to control a complete project. Following some use cases are described, there of course more of them and combinations out of them are imaginable.

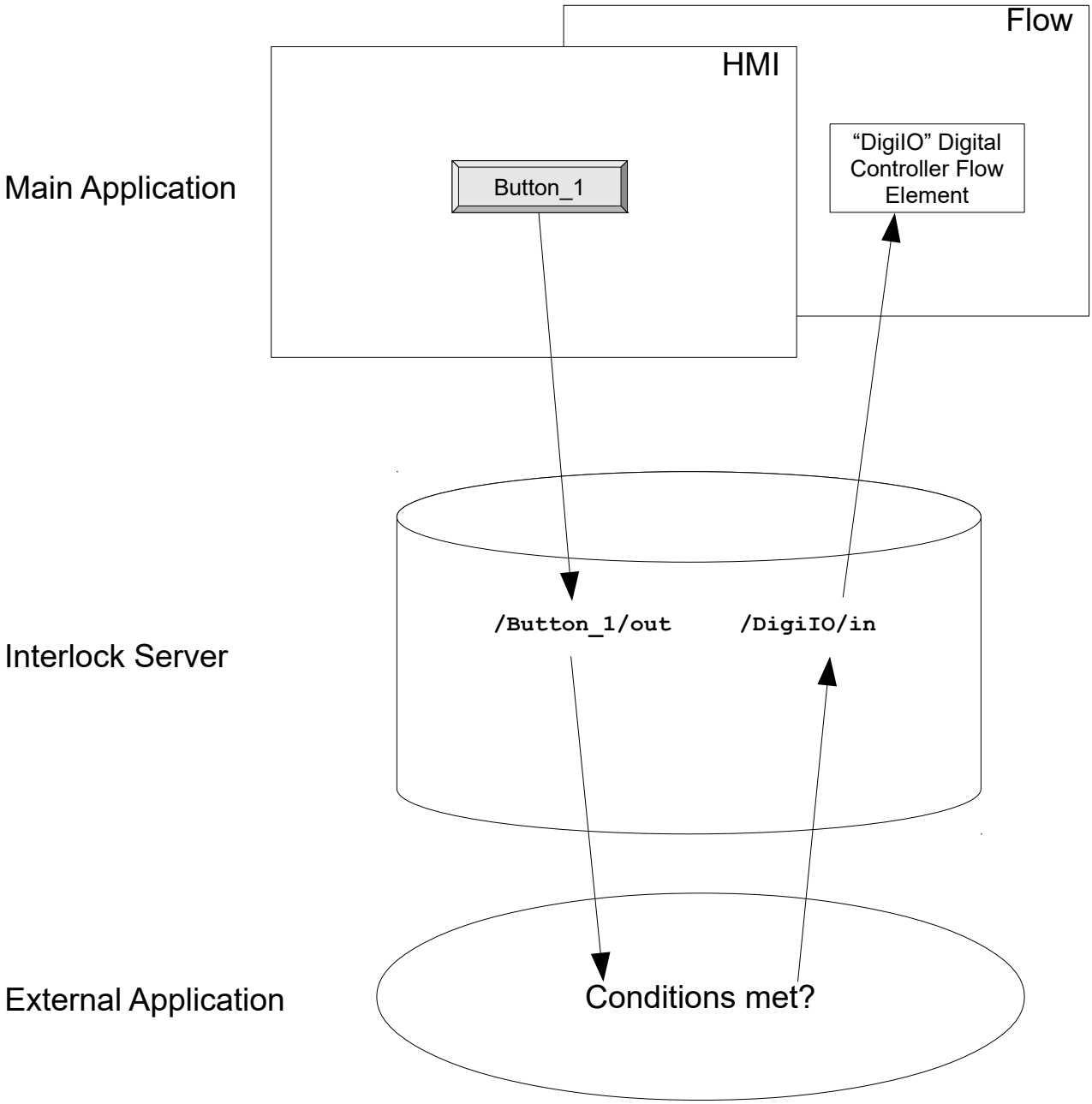
5.7.1 Reflect The Current State Of An Application

When all data elements of the main application are configured to send their output data to the interlock server and to receive input data from there as soon as they change within the scope of the server, the resulting configuration is nothing less than a complete backup of the current process data. When in this case for some reason the main application leaves unintentionally, it can be restarted and will be able to continue seamless at the point where it has left: on start-up it fetches all data from the interlock server and sets its internal values using the data received from there. So the last process state did not get lost.

PLEASE NOTE: whenever a value is received from the interlock server, it starts a new data flow. The data received from the interlock server are not ordered, there no prediction is possible which element is updated from the interlock server next. So the design of the data flow within the main application has to reflect this behaviour, elsewhere the project will not be able to continue with its last state.

5.7.2 Implement Software Interlocks

Many different application flows and logical connections can be done directly within the flow editor, these flows will run as soon as the project is executed. But there might be situations where this flow editor is not enough, may be because the logical combinations between elements are much to complicated or may be because the logic depends on external data sources that are not available within the scope of the ControlRoom program package.



In this case it is possible to configure the project in a way so that the state of all relevant elements are reflected within the interlock server. Next an external application can dock onto the server and communicate with it to watch and to influence these data. Now this external application can perform the tasks that no longer are implemented within a ControlRoom-internal flow.

As an example: There is a (very simple) project, that consists of a button which causes a motor to be turned on by setting a digital output. The state of the motor itself can be read from an digital input. Additionally there is an interlock condition: the motor is allowed to be turned on only in some cases (that might be a different

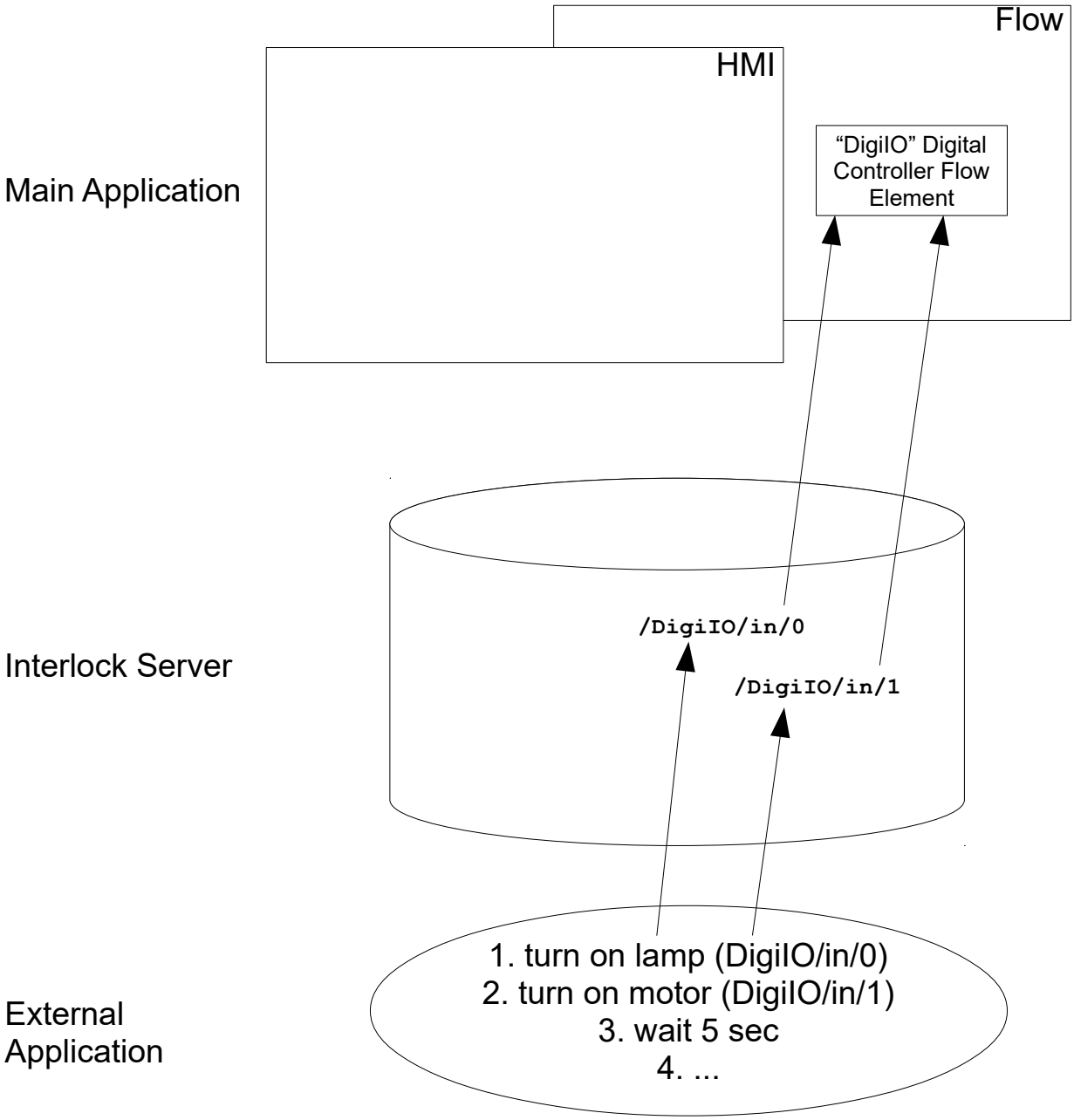
input that watches a door is closed or a database that has the information if this motor has to be used or not or something similar). This additional information is not available within the scope of the ControlRoom package, only the external application can access it. Here the set-up might be as follows: The button within the user interface is not connected to any other elements via the internal flows but it is configured to send its output values to the interlock server. As soon as the user presses this button, the changed output state is send to that server and the external application that has been connected to it gets the information about the pressed button. This external application and is now able to decide: is the interlock condition true so that the motor can be started or not? Only in case the motor is allowed to be used the application sends back a value to the interlock server. Now when the external application sets new data for an element that is linked to a flow element within the main application that itself sets the digital output which turns on the motor. As soon as the external application modifies this data node within the interlock server, the changes are sent to the OpenPlayer which itself sets the related flow element.

So comparing to a normal operation the way is just a bit longer: instead of having a direct flow connection from the button to the flow element that sets the output, the signal from the button is sent to the interlock server, is checked by the external application and the result is – conditionally – sent back by this application to the server which triggers the output. Now this external application is linked into the logic flow of data completely – and can be part of decisions and conditions that are necessary for a project.

5.7.3 Automatic Sequences

For some applications it might be necessary to implement automatic operations with complex sequences where different external outputs are set in a defined order to control external devices. Additionally it might be necessary to read some external input to check if some conditions are met. After reading of these inputs and the conditional usage of them is similar to what was described within the preceding section, the following example will be limited to a simple control that only sets some outputs. To describe the working principle imagine a project that has two elements which both are controlled via digital outputs: a warning lamp and a motor. Both have to be used in a defined sequence. First the warning lamp has to be turned on for at least 5 seconds, then the motor has to be started for some time. Afterwards the motor has to be turned of and in a last step the warning lamp has to be turned of too because the motor is no longer active, no user has to be warned about it.

Here the project within the main application may consist of exactly one flow element that controls two digital outputs: DIGI0 for the lamp and DIGI1 for the motor. Both are mapped to the interlock server so that the server is able to influence the state of the flow element. Next an external application is connected to the interlock server. This application implements the automatic sequence by setting the data within the interlock servers data space. First it writes a HIGH value to the data node within the server that corresponds to the flow element that is able to set the digital outputs. Here only DIGI0 is set to HIGH, DIGI1 is left unchanged so that only the lamp is turned on but not the motor. Next the application waits for 5 seconds before DIGI1 of the same element is set to HIGH. In both cases the interlock server informs all other connected clients about these changes, the OpenPlayer – which knows how to handle these data – takes them and sends them to the flow element which itself switches the digital outputs and therefore turns on the lamp and the motor. After some time the external application performs some similar steps: it sets DIGI1 to LOW to turn off the motor and afterwards sets DIGI0 of the correct data element to LOW to turn off the lamp.



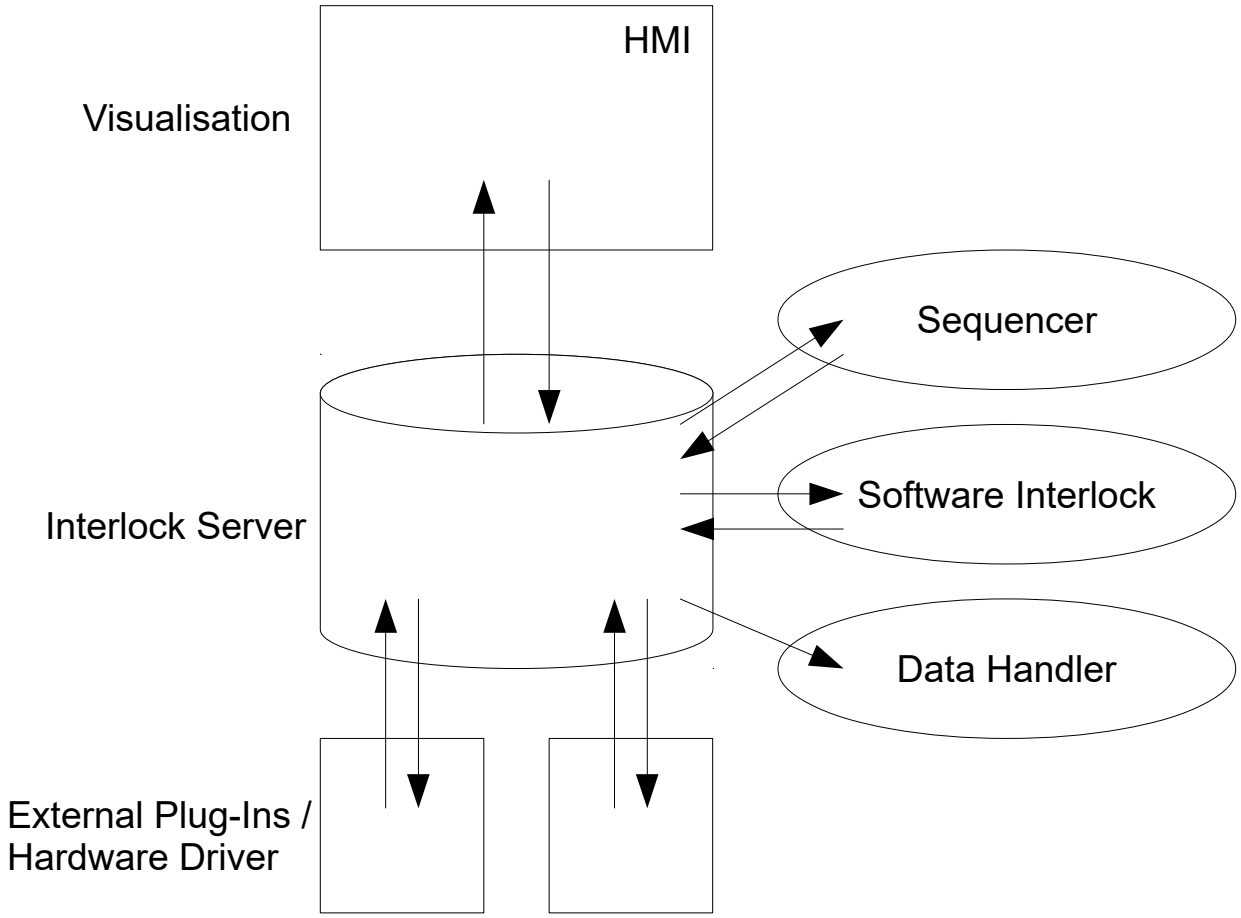
For this example the same is true as described within the preceding section: now the control no longer happens within a defined flow of the OpenDebugger/OpenPlayer but within an external application that takes control over the complete system via this way.

5.7.4 Conclusion

The examples given here show only some very specialized and very theoretical usage possibilities. Practically a combination of them will be used. Depending on the complexity of a project it is recommended to have several applications that do separate jobs on the same project. So a structure for such an environment could consist of:

- the main project running in OpenPlayer
- the Interlock Server running in background and holding the data of all user interface elements, flow elements and – what is possible just by sending data with a new, currently unused name to the server – helper data that are used to cache (additional) information

- an own application for automatic sequences that docks to the Interlock Server and runs automated processes
- and own application for software interlocks that checks if all security conditions are met that are necessary before an external device is used; in this case both, the application for automatic sequences and the main project from within the OpenPlayer never should access data elements directly that are located “behind” the interlock application, both should send events to the Interlock Server that have to be checked by the software interlock
- an own application for automatic processes that are no operational sequences and have to run all the time (here you can implement things like operating hours counters, as long as a specific device is turned on this application counts the time and writes the result into an other data element that itself can be displayed within the main projects HMI)
- an own application that stores important data from time to time to keep information that should not get lost (e.g. the operating hours counter values); at next application start-up these values of course have to be loaded so that they are still available



5.7.5 Accessing the Interlock Server

5.7.5.1 Access Via Own Applications

The Interlock Server and its data can be accessed out of own applications easily by using functions provided by liboapc. For details about how these functions can be used please refer to the OpenAPC SDK which is available for download from our webpage.

5.7.5.2 Access Via LUA Scripts

Lua is a lightweight, reflective, imperative and functional programming language, designed as a scripting language with extensible semantics as a primary goal. The name comes from the Portuguese word “lua” meaning "moon" and was developed at the Pontifical Catholic University of Rio de Janeiro, Brazil.

The ControlRoom software component comes with a special LUA interpreter “luaPLC” that already contains functions to access the data of the Interlock Server. The interpreter is a command line application that accepts the name of the LUA script file for execution. So by calling

```
luaPLC example.lua
```

the LUA interpreter is started, loads the file “`example.lua`” and executes it.

For a full description of the general functions and features of the LUA programming language please refer to http://www.lua.org/manual/.

The luaPLC interpreter additionally offers some specific functions which can be used for communication with the Interlock Server and for accessing its data, these functions have to be used in a similar way like the interface of liboapc (please refer to the SDK and the related manual for more details about the ControlRoom interface):

1. define a global callback function `oapc_ispace_recv_callback()` within your LUA-script
2. initialize the connection to the Interlock Server by calling `oapc_ispace_connect()`
3. access the data of the server using the callback and functions `oapc_ispace_set_data()`, `oapc_ispace_request_data()`, `oapc_ispace_request_all_data()`, `oapc_ispace_set_value()` and `oapc_ispace_get_value()`
4. Call `oapc_thread_sleep()` regularly to trigger the message handling and to let luaPLC call into `oapc_ispace_recv_callback()`
5. close the connection to the Interlock Server by calling `oapc_ispace_disconnect()`

Creation of a callback function within your LUA-script is mandatory, without such a callback no communication is possible with the server. This is true also in case no functions are used actively that send data to this function, in this case an empty implementation of it is necessary:

```
function oapc_ispace_recv_callback(nodeName, cmd, IO's, val0, val1, val2, val3,
val4, val5, val6, val7)
...
end
```

This callback function is called automatically whenever some data are changed within the database or when specific or all data are requested by executing `oapc_ispace_request_data()` or `oapc_ispace_request_all_data()`.

In all these cases the requested or new data are returned using the parameters of the callback function:

`nodeName` - the name of the data node that was changed in Interlock Server, this node name has format "/datablock/io" and does not contain the input number, this information is given in parameter `IO's` (please refer below)

`cmd` - specifies the reason for calling this function, when this value is equal to 256 (`OAPC_CMDERR_DOESNT_EXISTS`), it means a specific data block was requested by calling `oapc_ispace_request_data()` which doesn't exists. In all other cases the data could be retrieved successfully, the following parameters contain valid data and can be used

`IO's` - this numeric value contains a bit pattern that describes which of the following value-parameters

val0..val7 contains data of which type. This parameter contains OR-concatenated flags where every bit stands for one of the val-parameters. The position of this bit within one byte specifies the number of the val-parameter and the position of the byte where this bit is set defines the data type. So the bits from 0x00000001 to 0x00000080 specify digital (boolean) data types, 0x00000100 to 0x00008000 specify numeric data types and the range from 0x00010000 to 0x00800000 specifies character (text) data types. The range from 0x01000000 to 0x80000000 (binary data) is currently not available for LUA scripts.
As an example: when IO's is equal to 16385 (0x4001) that means val0 contains data of type boolean (0x01) and and val6 contains numeric data (0x4000), all other parameters val1, val2, val3, val4, val5 and val7 are set to "nil"

val0..val7 – these parameters may contain data depending on the state of the IO-flags

Please note: to let luaPLC call this function the main LUA script has to enable internal message handling. This has to be done by calling oapc_thread_sleep() regularly. When a script is running without these calls or when a script does not call it for a longer time, all received messages are held in a queue and do not arrive at this function. As soon as oapc_thread_sleep() is executed, all – probably old – messages are delivered at this function.

Following functions can be used to access the data of the Interlock Server:

oapc_ispace_connect(host,port)

This function tries to establish a connection to an Interlock Server. Here the string parameter host specifies the IP or host name to connect with, the numeric parameter port the port number the Interlock Server is available at. When host is set to an empty string "" and/or the port is set to 0 the default values are used.
This function returns 1 in case of success or a number not equal to 1 otherwise. All the following functions can be used only when this function returned 1, means when a connection could be established successfully.

oapc_ispace_set_data(nodeName,IO's,val0,val1,val2,val3,val4,val5,val6,val7)

This function can be used to send a complete set of data to the server. Its parameter can be used similar to the ones of the callback function oapc_ispace_recv_callback():

nodeName – the name of the data block where the data have to be set at in style "/datablock/io"

IO's – a set of bits that defines which of the following parameters contain values of which type; for the mapping of the bits please refer to description of oapc_ispace_recv_callback() above

val0..val7 – the values that have to be set for the data block at the same time

This function returns 1 in case of success or a number not equal to 1 otherwise.

oapc_ispace_request_data(nodeName)

Using this function the state of one single data block can be requested. The values of the data block are returned within the callback function. The only parameter of this function is the string nodeName which expects the name of the data block in format "/datablock/io". The same name is handed over to the callback function when the data are returned.

This function returns 1 in case of success or a number not equal to 1 otherwise. When a data block does not exist, this function will also return 1 (=success). The information about the unknown data block is given within the callback function, there the value of parameter cmd will be equal to 256 when the response for this data block is given.

oapc_ispace_request_all_data()

This function works similar to the preceding one but it request all data that are available at the Interlock Server at the moment. These data are returned at the callback function.

This function returns 1 in case the request could be sent successfully or a number not equal to 1 otherwise.

PLEASE NOTE: dependent on the amount of data the Interlock Server currently stores calling this function may cause a nameable system load and big traffic when all the data available are transferred.

oapc_ispace_set_value(nodeName,val)

Different to `oapc_ispace_set_data()` this function can be used to set exactly one value for a datablock. Here a string parameter `nodeName` is required that contains the name of the data block that has to be changed in style "/datablock/io/#" where # is a number in range 0..7 that specifies the exact position where the data have to be written. The second parameter, `val`, hands over the data that have to be send to this data block.

Here it is VERY important to hand over a value of the correct data type. This data type decides if the value can be written successfully or not. So for a digital IO of the data block a boolean value has to be given, for a numeric IO a numeric value has to be given and for a char IO a string value has to be given.
As an example: when a string value "42" is handed over for a numeric IO, it can't be set and the operation will fail. Here the correct value 42 or 42.0 has to be given.

This function returns 1 in case the request could be sent successfully or a number not equal to 1 otherwise. This return value informs only about the data transmission to the Interlock Server, it doesn't gives the information whether the data could be set correctly or not. In case the operation could be done successfully on server side too, the data block is noticed as a new/changed one within the callback function.

oapc_ispace_get_value(nodeName)

Using this function a singe value can be fetched from a data block without using the callback function. It fetches a data block where the name has to be set using parameter `nodeName` in style "/datablock/io/#".

The function itself returns two values result and value. Here result is equal to 1 when the operation could be done successfully. In this case the second returned value contains the data of the requested data block and its specific IO. The data type of this returned value depends on the type of the IO of this specific data block.

PLEASE NOTE: this function blocks until the data could be retrieved from the Interlock Server or until the connection to it failed. Depending on the network speed this may take some nameable time!

oapc_ispace_disconnect()

This function closes the connection to the Interlock Server. After calling `oapc_ispace_disconnect()` no more data blocks can be changed and the installed callback function `oapc_ispace_recv_callback()` is no longer notified when something changes within the Interlock Server.

oapc_util_thread_sleep(milliseconds)

This function is deprecated and will be removed in future software versions, so please use `oapc_thread_sleep()` instead.

oapc_thread_sleep(milliseconds)

This is a small helper function that implements a possibility to let the script sleep for a given time (in unit milliseconds easily) and to let LUA handle the internal messages during that time. This function has to be called regularly in order to receive data and to get changed data from the Interlock Server.

5.7.5.3 Access Via IL (Instruction Language) Scripts

Instruction List is one of the languages supported by the IEC 61131-3 standard. It is a low level language designed for programmable logic controllers (PLCs). The ControlRoom software component offers a special IL (Instruction List) interpreter, that extends the original IEC 61131 language by several functionalities that

are useful for accessing the data of the Interlock Server. Beside of that some of them make the use of IL easier.

The IL interpreter “ilPLC” is a command line utility that runs in background and does not offer any user interface. It has to be called with the name of the IL-scriptfile as parameter using the command line option “-f”:

```
ilPLC -f example.il
```

This example starts the IL interpreter with the IL script file “example.il”. Error messages and warnings are printed to the command line interface where it was started from. To get additional information about what is going on within a script, the interpreter can be executed with the additional parameter -v (“verbose”) which causes the IL-interpreter to print out much more information. When this option is enabled the current line number the interpreter is working with is displayed together with the command that is executed there.

Within the following sections the commands supported by the interpreter are described.

5.7.5.3.1 Commands according to IEC 61131-3

Here a short overview about the IEC 61131-3 conform commands is given. For a full and detailed description please refer to the original specification of the international standard.

LD – loads the operand into the accumulator, the operand can be a variable or a constant expression

LDN – loads the negated value of the operand into the accumulator, the operand can be a variable or a constant expression of an integer or boolean type

ST – stores the content of the accumulator into the operand variable

STN – stores the negated content of the accumulator into the operand variable, this command can be used only with integer or boolean data types for both accumulator and operand

S – sets the operand (data type BOOL) to TRUE when the content of the accumulator is TRUE or not equal to 0

R – sets the operand (data type BOOL) to FALSE when the content of the accumulator is FALSE or equal to 0

AND – bitwise AND of the accumulator and the operand, this command can be used only with integer or boolean data types for both accumulator and operand

ANDN – bitwise AND of the accumulator and the negated operand, this command can be used only with integer or boolean data types for both accumulator and operand

OR – bitwise OR of the accumulator and the operand, this command can be used only with integer or boolean data types for both accumulator and operand

ORN – bitwise OR of the accumulator and the negated operand, this command can be used only with integer or boolean data types for both accumulator and operand

XOR – bitwise exclusive OR of the accumulator and the operand, this command can be used only with integer or boolean data types for both accumulator and operand

XORN – bitwise exclusive OR of the accumulator and the negated operand, this command can be used only with integer or boolean data types for both accumulator and operand

NOT and **!** - bitwise negation of the accumulator's content, this command can be used only with integer or boolean data types for both accumulator and operand

ADD – addition of accumulator and operand, result is copied to the accumulator

SUB – subtraction of accumulator and operand, result is copied to the accumulator

MUL – multiplication of accumulator and operand, result is copied to the accumulator

DIV – division of accumulator and operand, the result is copied to the accumulator and the data type of the accumulators content is changed to `LREAL`

GT – check if accumulator is greater than operand, result (`BOOL`) is copied into the accumulator, this command can be used only with numerical and boolean data types for both accumulator and operand

GE – check if accumulator is greater than or equal to the operand, result (`BOOL`) is copied into the accumulator, this command can be used only with numerical and boolean data types for both accumulator and operand

EQ – check if accumulator is equal to the operand, result (`BOOL`) is copied into the accumulator, this command works using all data types including `STRING`

NE – check if accumulator is not equal to the operand, result (`BOOL`) is copied into the accumulator, this command works using all data types including `STRING`

LE – check if accumulator is less than or equal to the operand, result (`BOOL`) is copied into the accumulator, this command can be used only with numerical and boolean data types for both accumulator and operand

LT – check if accumulator is less than operand, result (`BOOL`) is copied into the accumulator, this command can be used only with numerical and boolean data types for both accumulator and operand

JMP – unconditional jump to the label specified as operand; this label has to exist in format "`labelname:`" elsewhere within the code but not within a different function block or outside of the scope of the current function block

JMPC – conditional jump to the label after a check if the accumulator is `TRUE`; this label has to exist in format "`labelname:`" elsewhere within the code but not within a different function block or outside of the scope of

the current function block

JMPCN – conditional jump to the label after a check if the accumulator is `FALSE`; this label has to exist in format "`labelname:`" elsewhere within the code but not within a different function block or outside of the scope of the current function block

5.7.5.3.2 Extended commands and statements

The following commands are ControlRoom-specific extensions of the IL language. In case they are used within an IL script, it may not run with other interpreters that do not understand these commands. Nevertheless they are very useful and make programming in IL much easier.

PRINT – prints the contents of the accumulator or of the (optional) operand to the console

EXIT – leave the instruction list immediately independent from the current position within or outside of a function block; this command terminates the execution of the instruction list and avoids that the main script is restarted automatically after its end was reached

STP – stop the execution of the current script context for the time given in accumulator (in milliseconds); the accuracy of this time depends on the underlying operating system

SIN – calculate sinus value of the accumulator's content; this operation changes the data type of the accumulator to `LREAL`

COS – calculate cosine value of the accumulator's content; this operation changes the data type of the accumulator to LREAL

POW – calculate the value of the accumulator raised to the power of the operand

LOG – calculate the logarithmic value of the accumulator

LN – calculate the logarithmic naturalis (e) value of the accumulator

SQRT – calculate the square root value of the accumulator; this operation changes the data type of the accumulator to `LREAL`

MOD – modulo of accumulator and operand, the result is copied to the accumulator; this operation can be performed with integer data types for both accumulator and operand only

USINT – change the current numeric data type of the accumulator to `USINT`, dependent on the current value this operation may cause a loss of data

BOOL – change the current numeric data type of the accumulator to `BOOL`, dependent on the current value this operation may cause a loss of data; here all values not equal to 0 are converted to TRUE, 0 is converted to FALSE

SINT – change the current numeric data type of the accumulator to `SINT`, dependent on the current value this operation may cause a loss of data

UINT – change the current numeric data type of the accumulator to `UINT`, dependent on the current value this operation may cause a loss of data

INT – change the current numeric data type of the accumulator to `INT`, dependent on the current value this operation may cause a loss of data

UDINT – change the current numeric data type of the accumulator to `UDINT`, dependent on the current value this operation may cause a loss of data

DINT – change the current numeric data type of the accumulator to `DINT`, dependent on the current value this operation may cause a loss of data

ULINT – change the current numeric data type of the accumulator to `ULINT`, dependent on the current value this operation may cause a loss of data

LINT – change the current numeric data type of the accumulator to `LINT`, dependent on the current value this operation may cause a loss of data

REAL – change the current numeric data type of the accumulator to `LREAL`, dependent on the current value this operation may cause a loss of data or accuracy

LREAL – change the current numeric data type of the accumulator to `LREAL`

CAL – call a function block those name is given as operand and return from this function block only when a command `RET`, `RETC` or `RETCN` is executed there.
Calling a function block lets the instruction list continue with that block until a return command is found there, afterwards the instruction list continues where the call to the function block appeared before. A function block has to be encapsulated with the statements `FUNCTION_BLOCK` and `END_FUNCTION` and is allowed to call other functions itself. This possibility has to be handled with care: when a function block calls itself in an endless iterative loop the application may stop or crash at some point when no more resources are available to handle this recursion.
When a function block is called some parameters can be handed over to it:

```
CAL print(text:="my text",colour:=255,thick:=var1)
```

Here in this example three parameters "`my text`", `255` and `var1` (which itself is a variable in the current execution context) are handed over, its values are stored within three new variables `text`, `colour` and `thick`. These three new variables exist only within the scope of the called function block, they can't be accessed outside of it. On the other hand every call to this function has to contain exactly these three variables, elsewhere the execution of the function block will fail when one of these variables is used, it will be undefined in this case. The new variables itself inherit the data type of the given values or variables.
Similar to these local variables every function block will use its own accumulator, means the contents of the current accumulator are not available within the function block and the contents of the accumulator within the

function block are not accessible outside of it

CALC – conditional call of function block similar to `CAL` if accumulator is `TRUE`

CALCN – conditional call of function block similar to `CAL` if accumulator is `FALSE`

RET – return from a function block and jump back to the calling position within the instruction list

RETC – conditional return from a function block and jump back to the calling position within the instruction list if accumulator is `TRUE`

RETCN – conditional return from a function block and jump back to the calling position within the instruction list if accumulator is `TRUE`

VAR_GLOBAL / **END_VAR** - these statements encapsulate a list of global variable definitions, this block is not allowed to exist within a function. The variable definitions have to be done in style

```
name : type
```

with one variable definition per line (as an example: "`var1 : UINT`" defines a variable of type `UINT` with name `var1`).

VAR_LOCAL / **END_VAR** – encapsulates a list of local variable definitions, this block is allowed to exist only within a function block and defines variables to be used only within this function block. Such local variables are not accessible from other, subsequent function blocks and are not accessible from the top level instruction list scope.

FUNCTION_BLOCK / **END_FUNCTION** – these two statements encapsulate a function block that can be accessed using one of the commands `CAL`, `CALC` or `CALCN` (please refer the description of `CAL` for some more details)

5.7.5.3.3 Reserved Function Names

Normally function blocks can be defined and used freely within a instruction list. But there exist some reserved function block names that behave different: they have to be defined under some special circumstances and they are not allowed to be called from the script itself. These function blocks are used for the connection to the Interlock Server and are called independent from the current position within the instruction list. Beside of that they are called in parallel: when an event occurs that causes such a function block to be executed the main program flow of the instruction list is not interrupted, both continue working. So when these special function blocks are used it is important to check which global variables are used and modified from within this function block.

The reserved function block names are `isX_cb_digi_data`, `isX_cb_num_data` and `isX_cb_char_data` (where `X` is the number of the Interlock Server connection the function belongs to). The purpose and usage of these function blocks is described within the following section.

5.7.5.3.4 Extended Commands for Interlock Server Access

The following commands can be used to access the data of a connected Interlock Server and to control other connected components via these data. These commands all contain an element "ISx" where "x" is a number that identifies a connection to an Interlock Server (currently only one connection is supported so x has to be 0 always).
The order of commands reflects the calling order of the related functions in liboapc (please refer to the SDK-manual for a detailed description of the usage of the liboapc-functions): first a connection to the server has to be established (using command "ISxC"), then the data can be used (with ISxRA, LDISx, STISx) and at the end the connection has to be closed (ISxD). As soon as a connection is opened all changes within the data space of the Interlock Server can be watched by the script. To do that three function blocks isX_cb_digi_data, isX_cb_num_data and isX_cb_char_data can be defined within the script to receive information about data changes within the interlock server.
Here the "X" has to be replaced by the number of the server connection these function blocks belong to (so currently their names are is0_cb_digi_data, is0_cb_num_data and is0_cb_char_data). These function blocks are called in parallel and independent from the position of the main script. They do not return to a position within that script after they have been finished. For a more detailed description of these function blocks please refer below.

ISxC – establishes a connection to a local Interlock Server; the success of this operation is returned in accumulator as BOOL, after this command was done successfully all changes of data within the Interlock Server are notified at the isX_cb_YYY_data (where X is the number of the server connection and YYY specifies the data type it handles) function blocks

ISxRA – this command requests all data that are currently managed by the interlock server, the result is sent to the local callback function blocks isX_cb_YYY_data (please refer below); this command has to be handled with care, it may cause a big load of data

LDISx – loads a value from the Interlock Server into the accumulator, the resulting accumulators data type depends on the IO of the loaded data block; as operand a string (or a variable of type STRING) has to be given that specifies the name of the data block, here a node name in format "/datablock/io/#" has to be given. Within that name "datablock" is the name of the block (e.g. "Toggle_Button_1"), "io" specifies if the input "in" or the output "out" of the block has to be read and "#" defines the number of the IO in range 0..7.
PLEASE NOTE: this command uses a network connection. In case of a broken or bad connection it may need several seconds until the command returns and the script continues. Beside of that the time this command needs until it finishes is not defined in general. So it is not recommended to use this command in productive environments but only for testing. For an alternative method of accessing data of the interlock server please refer to the description of the MAP_ISx / END_MAP statements below.

STISx – stores the value of the accumulator into the Interlock Server, afterwards the content of the accumulator is a BOOL value that specifies if this command could be executed successfully. As operand a string (or a variable of type STRING) has to b e given that specifies the name of the data block, here a node name in style "/datablock/io/#" has to be used. Within that name "datablock" is the name of the block (e.g. "Toggle_Button_1"), "io" specifies if the input "in" or the output "out" of the block has to be read and "#" defines the number of the IO in range 0..7. The data type of the accumulator needs to fit to the data type of the chosen data blocks IO, elsewhere this operation will fail. When an data block or IO is used that doesn't exists within the Interlock Server, a new data block and/or IO is created by this operation.
PLEASE NOTE: when a value is stored that is mapped to a global variable using the MAP_ISx / END_MAP statement, this modification is NOT notified to this variable, only the contents of the data block within the Interlock Server are changed. Because of that it is not recommended to use this function within a script when mapped variables are used too (please refer to the description of the MAP_ISx / END_MAP statements below).

ISxD – closes the connection to the Interlock Server, afterwards no more "ISx"-commands can be used and

no data are received at the isX_cb_YYY_data callback functions

isX_cb_digi_data – this is a callback function which has to be defined within a FUNCTION_BLOCK / END_FUNCTION block. As soon as a connection to Interlock Server is established that belongs to the number "X" this function block is called automatically whenever digital data within the servers data space change or when all data are retrieved from the server using command "ISxRA". When this function block is called following local variables are created and can be used here:

- nodeName (STRING) - the name of the data block the following data belong to
- IO's (UDINT) - a bit mask that specifies which digital IO's have changed, here the lower 8 bits (bits 0..7) are used
- digi0 .. digi7 (BOOL) - variables containing the data of the data block, here only these variables contain valid data that are specified by the bit mask of variable "IO's"

isX_cb_num_data – this is also a callback function that has to be defined explicitly within an IL script. As soon as a Interlock Server connection is established that belongs to the number "X" this function block is called automatically whenever numeric data within the servers data space change or when all data are retrieved from the server using command "ISxRA". When this function block is called following local variables are created and can be used here:

- nodeName (STRING) - the name of the data block the following data belong to
- IO's (UDINT) - a bit-mask that specifies which digital IO's have changed, here the bits 8..15 are used
- num0 .. num7 (LREAL) - variables containing the data of the data block, here only these variables contain valid data that are specified by the bit mask of variable "IO's"

isX_cb_char_data – a reserved function block name that can be used as callback function: as soon as a Interlock Server connection is established that belongs to the number "X" this function block is called automatically whenever text data within the servers data space change or when all data are retrieved from the server using command "ISxRA". When this function block is called following local variables are created and can be used within the scope of this function block:

- nodeName (STRING) - the name of the data block the following data belong to
- IO's (UDINT) - a bit mask that specifies which digital IO's have changed, here the bits 16..24 are used
- char0 .. char7 (STRING) - variables containing the data of the data block, here only these variables contain valid data that are specified by the bit mask of variable "IO's"

MAP_ISx / **END_MAP** – when data of the Interlock Server are accessed via "ISxRA" / "LDISx" / "STISx" the result can be accessed only via an indirect way or the communication may slow down the execution of the script for a non-predictable time. Because of that it is also possible to map IO's of data blocks directly and asynchronously into the scope of an instruction list script. These mapped values are assigned to a variable, means whenever the data within the server change the value of the variable changes. And whenever the user changes the value of such a variable these modification is sent to the Interlock Server automatically. Such a map can be defined in a similar way like variables are defined, but additionally the name of the data block has to be specified:

```
variable : type : /datablock/io/#
```

Here "variable" is the name of the defined variable, "type" is the data type of it and "datablock" is the name of the block (e.g. "Toggle_Button_1"), "io" specifies if the input "in" or the output "out" of the block

has to be read and "#" defines the number of the IO in range 0..7. So as an example a definition

```
level : INT : /Level_Control/out/3
```

would assign a variable "level" which is of type INT to the data block named "Level_Control". From this data block the output number three (equal to the fourth output) is mapped to this variable.
PLEASE NOTE: the data type defined within the instruction list and the data type of the data blocks IO within the interlock server have to fit. If they do not, the variable is not set. So for a digital input or output a variable of type BOOL has to be defined, for a character I/O a variable of type STRING has to be set as well as for a numeric I/O one of the number data types from USINT to LREAL need to be used.

5.7.5.3.5 Supported Data Types

The following data type identifiers can be used within VAR_GLOBAL, VAR_LOCAL and MAP_ISx statements to set a type for a variable that is defined there. The type specifies the allowed range and the purpose of this variable.
Please note: there are also IL commands available with the same name which can be used to cast the contents of the accumulator to a different data type. Here the usage context decides how a mnemonic is used: when it is called within the executable part of a script, it is an active statement that causes a modification to the accumulator. Only when they are used within VAR_ or MAP_ blocks they define the data type of a variable and do not modify the accumulator.

USINT – whole numbered unsigned data type, range 0..255 (8 Bit)

SINT – whole numbered signed data type, range -127..128 (8 Bit)

UINT – whole numbered unsigned data type, range 0..65536 (16 Bit)

INT – whole numbered signed data type, range -32767..32768 (16 Bit)

UDINT – whole numbered, 32 Bit unsigned data type

DINT – whole numbered, 32 Bit signed data type

ULINT – whole numbered, 64 Bit unsigned data type

LINT – whole numbered, 64 Bit signed data type

REAL – 32 Bit floating point data type (signed)

LREAL – 64 Bit floating point data type (signed)

BOOL – boolean data type that accepts only two states TRUE (!=0) and FALSE (=0)

STRING – a text data type that can contain an ASCII-text of variable length

5.7.6 Usage Examples

There are usage examples available within the OpenAPC SDK (software development script). So to get some working examples of scripts and programs that access the Interlock Server, please download the SDK from our website.

5.7.7 Interlock Server Operation Modes

The Interlock Server can run in different modes. The mode is set within the settings of the project and needs to be chosen depending on the desired functionality. Following the available operation modes, their purpose and usage is described.

Please note: An Interlock Server will not be stopped automatically when the connected applications terminate, in this case it simply flushes all data and stays active. Now when a mode is used where the OpenPlayer or the OpenPlugger would execute the Interlock Server automatically and it is still running, no further action is taken. That means, a changed operation mode will not be applied to the running instance of the OpenIServer, thus it will not switch over to the possibly different operation mode.
In such cases it is recommended to kill the OpenIServer first before running a project with a different Interlock Server mode.

5.7.7.1 Single Local Mode

This mode is the most simple one, here the OpenPlayer or OpenDebugger starts exactly one instance of the OpenIServer automatically. This instance runs on the same host, external applications have to connect to that host only, no further actions are necessary.

5.7.7.2 Mirrored Local Mode

Using this mode two instances of the OpenIServer are started by OpenPlayer or OpenDebugger. One of both becomes active immediately and its server socket can be accessed by applications. The other one mirrors all data from the active instance but is not accessible from outside.

As soon as the first instance dies for some reason the second Interlock Server activates its server socket, replaces the instance that has gone and starts a new Interlock Server that now will mirror all the data.

External applications that make use of the Interlock Server will lose the connection to it at this point, they now simply have to reconnect to the same IP again. Standard applications like the debugger, the player or the OpenPlugger will perform this reconnect-operation automatically.

5.7.7.3 Single Remote Mode

This server mode is similar to the single local mode but with one difference: the Interlock Server will not run on the same host but somewhere else in network, thus the OpenDebugger or OpenPlayer are not able to start the OpenIServer and the possibly required OpenPlugger instances automatically. So a start up procedure is necessary that executes the components of the whole system in following order:

1. The Interlock Server has to be started on the host that is configured within the project parameters
2. All OpenPlugger instances have to be started using the project file that contains the information about the Interlock Servers host
3. External applications that implement software interlocks have to be started
4. The OpenPlayer has to be started using the same project file (please refer pt. 2)
5. External applications that implement automatic sequences have to be started

5.7.7.4 Redundant Remote Mode

This is the most secure and also the most complex operation mode. It extends the mirrored local mode, here again two instances of the Interlock Server exist where one mirrors the data of the active one but they are executed on different hosts so that a complete hardware failure can be managed without any interruption. This mode also requires a manual start-up procedure with a fixed order to ensure proper operation:

1. The first Interlock Server has to be started on one of the the hosts that is configured within the project parameters and with command line option “-r <ip>” where “ip” is the IP of the other, not yet started server; this interlock server will become the active one automatically
2. The second Interlock Server has to be started on the other host that is configured within the project parameters and with command line option “-r <ip>” where “ip” is the IP of the other, already accessible server; this interlock server will connect to the active one and start mirroring of its data

3. All OpenPlugger instances have to be started using the project file that contains the information about the Interlock Servers host; they automatically try both IP's and connect to the active Interlock Server
4. External applications that implement software interlocks have to be started, they have to check both IP's actively and use the one where the Interlock Server is accessible
5. The OpenPlayer has to be started using the same project file (please refer pt. 3); it automatically tries both IP's and connects to the active Interlock Server
6. External applications that implement automatic sequences have to be started, they have to check both IP's actively and use the one where the active Interlock Server is accessible

Now that the whole system is up and running a problem might happen with one of the both hosts where the Interlock Servers are running.

As soon as the first, active instance of the OpenIServer dies for some reason, the second Interlock Server activates its server socket.

External applications that make use of the Interlock Server will lose the connection to it at this point, they now have to reconnect to the other IP to continue operation with the other Interlock Server that already contains the data of the first one. Standard applications like the debugger, the player or the OpenPlugger will perform this reconnect-operation automatically. Whenever the host with the formerly active Interlock Server is back, the OpenIServer has to be restarted again with the same command line option “-r <ip>”, it connects to the now active Interlock Server and starts mirroring of its data.

In this scenario the state of the two hosts that keep the Interlock Server has to be watched externally an – in case one of them dies – restoring of it hast to be triggered externally too.

5.8 Using the OpenPlugger

The OpenPlugger is an application that is important for system architectures where the player/debugger does not contain all functionality and logic but where the access to devices is externalized.

Here the OpenPlugger performs following tasks:

- load exactly one plug-in out of the ones specified in the given project file
- communicate with the plug-in (initialize, send and receive data,...)
- communicate with the Interlock Server (send data received from the Interlock Server to the plug-in and vice versa)

So an OpenPlugger can be used only with projects that make use of the Interlock Server. In case a local Interlock Server is configured within a project, the OpenPlayer (or the OpenDebugger) will start all required OpenPlugger instances automatically. But in case a remote Interlock Server is used that needs to be done manually after the server and before the player was started.

In such cases the OpenPlugger can be executed using the following command line options:

- **-p <project_file>** - specifies the path to the project file; this project file has to be exactly the same like it is used for all other OpenPlugger instances and for the OpenPlayer/OpenDebugger. Beside of that the Interlock Servers IP that is configured within that project file needs to be accessible from the host where the OpenPlugger is started from in order to establish a communication channel with the Interlock Server
- **-n <plug_in_name>** - the unique name of the plug-in that has to be handled by this instance of the OpenPlugger, every plug-in within a project that was configured for external use has to be handled by exactly one OpenPlugger
- **-i <id>** - the unique number that was assigned to a plug-in within the Plugged Devices list, every plug-in within a project that was configured for external use has to be handled by exactly one OpenPlugger; in case this option is used no plug-in name (option -u) needs to be given
- **-V** – be verbose, print out additional information on the command line that may help to find problems

- **-h** – display a short help that lists and describes all available command line options

Here the option **-p** which specifies the project file and one of the option **-n** or **-i** are mandatory, they are necessary to identify the plug-in that has to be used.

When a plug-in runs within an OpenPlugger instance it gets all data from and sends all its own data to the Interlock Server automatically. From there no default communication channel to the player is defined. Here during set-up of the project or during set-up of the complete system the communication path has to be created explicitly. This can be done via the Interlock Server Connection Flow Element (not recommended) or via separate scripts or programs that dock onto the Interlock Server, receive data from one side (player or external plug-in) and forward them to the other one (external plug-in or player). Such scripts or programs can be used at this position to implement software interlocks: they can check if an operation requested by the OpenPlayer is really allowed and safe in the current state of the application and then allow or reject these operation by forwarding the request to the plug-in or by dropping it. This would extend the use case described in section “Implement Software Interlocks” above, here not only the data flow would be separated but also the applications where the data are handled into. That adds an other level of security.

5.9 Using the User Management Functions

The ControlRoom software offers an integrated user management that gives the possibility to change the layout of an HMI depending on the user that is currently operating it. When this user management is enabled within a project, a user has to log into the system first to get its personal view to the HMI. Depending on the privileges this user has (which may reflect. e. g. the experience of this user) he will be able to access functionalities of the project or not.

This user management requires several conditions that require each other and intertwine with each other:

- user privileges have to be defined (e. g. the privilege to start a process, the privilege to change parameters, the privilege to shut down the system, ...)
- users have to be defined and privileges (as created in previous step) need to be granted to them
- HMI elements have to be configured if they are enabled and usable, disabled and visible or invisible for the different privileges

Now when a user logs in, all these data are taken and it is checked for every HMI element how it has to appear for this specific user. Here the user management within the ControlRoom software behaves different than within other visualisation solutions. The user privileges have a priority here (which is similar to their order within the user privilege panel) and a HMI element can support different visibility states for different privileges. Now when a user has more than one privileges which would overlap within the HMI definition, this priority comes into account: the visibility definition of the privilege with the higher priority is taken. This gives the possibility not only to change the visibility of HMI elements depending on one privilege but also depending on combinations of privileges a user might have.

Following it is described which mandatory and optional steps have to be done in order to set up the user management for a ControlRoom application.

5.9.1 Defining User Privileges

First of all the user management has to be enabled and a set of user privileges together with their names have to be defined. This can be done within the global user privilege panel in menu “File” -> “User Privileges”. This panel contains an “Enable”-check-box that has to be set as very first. Only if this option is enabled, all the different user management functionalities, the related HMI element and flow objects are available.

Next a bunch of user privileges has to be set that are relevant for a project. This is done by setting a short, descriptive name for every of them. These names can be used in later steps to identify such a privilege non-ambiguous. By default the application offers some possible privileges that can be used or modified. Right beside the input field for the privilege names two arrows for up and down can be found. If these arrows are pressed, a privilege definition is moved up or down in the list of available privileges. The higher a privilege is stored within this list the higher its priority is. This is important later when the privileges are assigned to visibility states within an HMI element.

This order of privilege names also can be changed at a later time, when moving its position the related flags within the HMI elements and the user definitions that are using it change their position too.

There is one predefined privilege “Supervision” on top of the list which cannot be removed or edited. This privilege is used for full access to the complete system and is available in every case.

5.9.2 Defining User Data

Next below the menu item for the user privilege panel an other one can be found where the users, their names, data and granted privileges can be set with. This user data dialogue opens when menu “File” -> “Users” is selected. It shows the list of existing users on the left hand side. On the right hand side the formerly configured user privilege names can be found.

Whenever a user is selected within the list, its data are shown on the right hand side of the dialogue: the privileges that are granted for the user, the login-name of the user, the full name, a freely usable comment and others. These values can be changed and have to be set to the user by pressing the “Apply” button. Please note: Such changes have to be applied in order to set them to a user before an other one is selected or the dialogue is left, elsewhere the changes are lost.
Within the privileges there again the option “Supervision” can be found. When this privilege is selected, all other check boxes are disabled because it overwrites all other definitions.

When a new user is generated using the button “Create user”, the log in name of this user has to be given. Afterwards the list of users is extended by this new user. As default password the login name is used. This default password should be changed using the button “Set password”.

Predefined there always exists a user “Supervisor” which has “Supervision” privileges. It is recommended not to delete this user or to have at least one other user with “Supervision” privileges.

All data that are set here are stored within the project file as some kind of project default values. There exists a possibility to modify the user data within a running project. The values changed here are not stored within the project but within a special file that is stored in application directory within a file that has the same name like the project file plus extension “.rtdat” or “.rtbak”. When these files do not exist or have been deleted the default user data specified within the OpenEditor and stored within the project file are used. Thus this is a possibility to get access to a system that is locked and where the login information are lost. On the other hand it is necessary to protect general operating system access so that nobody is able to delete these files accidentally or illegally.

5.9.3 Specifying Visibility States for HMI-Objects

When the user management is enabled and privileges are defined as a next step it is possible to set the visibility for the HMI elements. There within the configuration dialogue a special panel “User Privileges” exists. Within this panel all existing privileges are shown ordered by priority (as defined within the “User Privileges” settings dialogue out of menu “File”). Here it can be defined which visibility and access state this HMI object will have dependent on the logged in user and the privileges that have been granted to this user.

It can be chosen from the options “enabled” (the element is visible and usable), “disabled” (the element is visible but not usable) and “invisible” (the element is invisible and therefore not usable). As a fourth option “ignore” can be chosen. This option is necessary because within a ControlRoom project more than one privilege can be assigned to an user element. Because of this functionality – which does not exist in most other HMI solutions – the assignment of visibility states to privileges is a bit more complex but offers much more flexibility. So it is possible to have different visibility states not only dependent on one privilege only but also on combinations of privileges. To use this feature it is necessary to understand how this feature works: When a user logs in, the list of privileges that are granted to this user is sent to every HMI element. Now the list of privileges set within the HMI element is checked from top to bottom (according to their priorities) and compared with the list of privileges of the current user. When the user has a privilege granted that was set for this HMI element too, the related visibility “enabled”, “disabled” or “invisible” is set to the HMI element and comparing user and HMI privileges is stopped. When the HMI element is set to “ignore” for this privilege comparison continues with the next privilege in the lists. So the complete list is checked until a non-”ignore” option is found for this HMI element. If no matching privilege could be found the visibility state is used that is set for the option “None of them”.

To clarify that following examples are given that base on a (shortened) list of privileges that may be set to a HMI element:

Privilege	enabled	disabled	invisible	ignore
Supervision	**X**			
Manager Users				**X**
Load Data		**X**		
Start Process	**X**			
...				
None of them		**X**		

1. A user logs in that has the privileges “Manage Users” and “Start Process” granted. Resulting from that log in process the complete list is checked. The first match can be found at row “Manage Users”. Here “ignore” is set so it is continued with the list. The next match is in row “Start Process” which is set to “enabled”. Now the HMI element is enabled and the operation is finished.
2. Alternatively an other user logs in with privilege “Manage Users” only. Here the visibility is set to “ignore” so that the other privileges are checked. After none of the other HMI element privileges fit to the users privileges the operation stops in the last row “None of them” and disables the HMI element.
3. Now a user logs in with the privileges “Load Data” and “Start Process”. The first match between user privileges and HMI element privilege data can be found in row “”Load Data”. Here the HMI element is disabled and the operation is finished.
4. No user is logged in (which is default at program start-up) or the last user has logged out. In this case no privileges are given and the definition of row “None of them” is used, the HMI element is disabled.

Example number 3 shows the difference to simple 1:1 privilege assignments: There is a difference between users that have the privilege “Load Data” and “Start Process” and users that have the privilege “Start Process” only. Both of them use the privilege “Start Process” and this privilege is defined for the HMI element but they still get different visibility states for this HMI element.

Resulting from that its a bit different to implement the most simple possibility where exactly one user privilege is used for the HMI element to enable it and to disable it for all others. To get this assignment all visibility states have to be set to “ignore”, except the one where it has to be enabled. It has to be set to state “enabled” and “None of them” defines the “disabled” state for all other possible user privileges.

5.9.4 Logging In Users During Runtime

After all these definitions have been done as a last step a possibility has to be set up to log in a user during runtime. There exists a special flow element that accepts a users login name and a password as input and – if both fit to existing user data – switches the state of all HMI elements automatically.

There are no additional operations necessary, the user management and HMI element enabling/disabling is done fully automatically by this flow element. For a full description of this element, its inputs and outputs please refer to section “Flow Elements” above.

5.9.5 Changing User Data During Runtime

The user data that are set within the OpenEditor are default values that are stored within the main project file. To offer a possibility to manage and change users during runtime within the OpenPlayer a special HMI element “User Management Panel” can be added. This panel looks and works exactly like the user management panel that can be found in OpenEditors “File” menu.

PLEASE NOTE: When such a HMI panel was added to a project, it has to be configured to be enabled only for users with a privilege “Manage Users” in order to avoid that everybody is able to change user data and grant itself privileges.

Whenever user data are changed during runtime within this panel the modified data are stored within an own data file. This data file is located in application directory (the exact location of this directory depends on the used operating system). It has the same name like the project file plus an extension “.rtdat” or “.rtbak”.

PLEASE NOTE: user data changed within a HMI running in OpenPlayer are never stored within the original project file but within these special data files. So when these files get lost, when they are deleted or when the project is moved to an other computer and the files are not copied with it, the application goes back to the default user data that have been done within the OpenEditor and that are stored within the project file.

6 BeamConstruct

6.1 Security

This application is designed to control laser equipment which may effect a person's health or may otherwise cause damage. Prior to installation and operation compliance with all relevant safety regulations including additional hardware-controlled safety measures has to be secured.

Beside of that some laser equipment can be damaged in case it is controlled with wrong signals. Thus it is highly recommended to check the output generated by this plug in using e.g. an oscilloscope to avoid problems caused by wrong configurations. This should be done prior to putting a system into operation for the first time and whenever a software update was installed or whenever some of the relevant parameters have been modified.

Here “software update” not only means updates of the OpenAPC or BeamConstruct software but also modifications with dependent software, like components or drivers for scanner controller cards.

6.2 Overview

BeamConstruct is a CAD application specialised for creation and processing of laser marking data. These data can be used together with a laser scanner system to perform the material processing. Beside of that BeamConstruct is able to store the generated processing data in different formats that can be used by ControlRoom plug-ins directly. Thus the generated laser marking data can be used within a process control environment directly and without the need to deploy the editing software “BeamConstruct” on the target system. This application is not only able to generate these data but – optionally – also to test it and to start laser marking and related motion processes directly out of the application.

6.3 Position within the system

BeamConstruct is not a vital part of the OpenAPC runtime environment. More than this OpenAPC not necessarily requires BeamConstruct main application for proper operation also in case a ControlRoom project is used for processing BeamConstruct-generated data. This application exists beside the ControlRoom environment and is an additional tool that primarily belongs to construction process, not to production.

So one possible usage cycle is as follows:

1. Processing data are created within BeamConstruct, that can be done apart the production place, e.g. in an construction office.
2. The constructed data are saved or exported.
3. The saved or exported data are transferred to the production where an ControlRoom project is running.
4. This ControlRoom projects loads the BeamConstruct-data and uses it as binary control data internally for processing materials in the desired way.

Optionally it is possible to perform some additional fine-tuning between step 2 and 3 with the generated project files using an other (or the same) installation of BeamConstruct at the production with the complete external hardware. Here the capability of BeamConstruct to mark directly out of the application can be used.

An other, more simple usage scenario would be as follows and can be used preferentially in experimental and testing environments (since here no real machine-HMI is involved):

1. Processing data are created within BeamConstruct, that can be done close to the laser marking equipment.
2. The constructed data are marked out of BeamConstruct directly, here the laser marking software has full control over scanner, laser and any additional hardware equipment.

6.4 Quick Start into BeamConstruct

This section describes how to use BeamConstruct in direct operation mode where laser marking is done out of the application and without the use of a separate ControlRoom HMI/process control project. It gives a short overview about basic set-up and usage of BeamConstruct and describes with some short steps how to get first results easily. So this section is a shortcut to start into this software very fast but it ignores most of the useful features. These features are described in detail in following sections of the users manual, so it is recommended to use this quick start only to understand the very basic working principles of the software while going deeper into it afterwards.

1. **SECURITY CHECK**: The following steps describe how to set up BeamConstruct and how to control laser equipment out of it. Thus all laser safety rules and regulations need to be respected, all required technical security mechanisms need to be available and active prior to starting with this software.
2. **Main configuration**: First the scanner controller card that has to be used needs to be selected and configured. To do that, select Menu “Project” menu item “Project settings...” and go to panel “Hardware” of the now opening settings dialogue. Within the selection-list named with “Scanner Card” choose the scanner controller card you want to use. After selecting it, press the button “Configure” and set up the scanner controller card according to its specific needs (choose things like firmware file, correction table, working area size, number of controlled axes or what ever your card requires for proper operation) and according to your local hardware configuration.
3. **Save Configuration**: Leave all these settings dialogues by pressing “OK” once all parameters have been entered correctly and select menu “Project” menu item “Save as default configuration”. Now whenever you start BeamConstruct and as long as you do not load a project file with a different hardware set-up, these settings are used by the application.
4. **Create basic geometry**: Select the blue triangle-symbol within the tool bar and draw that triangle within the drawing area right below the tool bar. There you have to left-click your mouse three times at different positions to create the triangle, every of these mouse-clicks specifies an other corner of the triangle.
5. **Modify existing geometry interactively**: Once you have finished creation of the triangle this new element is selected and highlighted by a double-lined blue box surrounding it. This box can be used to modify the geometry. While picking and dragging one of the coloured squares that are contained in this surrounding, the triangles size can be changed (the grey squares) as well as its rotation (via the red square). The cyan-coloured squares can be used to slant the geometry. When the triangle is dragged by clicking the selection frame between these coloured squares, its position can be changed.
   PLEASE NOTE: When modification of the geometry has finished it still has to be located completely within the working area that is symbolised by the grey rectangle. This working area is the maximum range your scanner can work within, so all geometry that is located outside of that range will be cropped.
6. **Modifying existing geometry manually**: The method described previously is a very quick way to change the geometry but it is also a very inaccurate one. Thus there exists an other possibility: As long as an element is selected, there is a configuration panel usable at the windows left hand side. The first configuration panel is always an element-specific one, in this example it contains several parameters and values that influence the creation of the triangles base geometry. On the same side of the main window there also exists an other tab-pane “Geometry” where the generic geometric data of an element can be changed by entering the desired numeric values. Using this way things like size, position and rotation can be modified in a very exact way. The related values are taken over and applied to the geometry as soon as the data input is confirmed by pressing “Return” or by leaving the input field.
   Several of these data depend on each other and will change automatically whenever one of them is modified. So in case you change the scale factor of your geometry, the position and size values of it will be changed too as soon as you hit your “Return”-key within the scale value input field.
7. **Apply laser parameters**: Required laser parameters can be applied to existing geometry via a number of predefined and freely definable pens. Such an (existing) pen can be set via the tab-pane

"Element" and the selection list named "Pen". Whenever a new pen is selected for an existing geometry it is applied to it automatically, which means the visual representation of that geometry within the drawing area changes its colour and the next laser marking cycle would use the parameters of this pen when this specific geometry is processed.
The different laser and scanner parameters that are assigned to such a pen can be modified by selecting menu "Project" menu item "Pen settings...". There a dialogue opens where you can select the pen that has to be modified (by using the selection list on the upper side of the dialogue). The tab-panes below this list give access to all relevant parameters that influence mark and jump speed, laser power and frequency, scanner delays and others more. When you change some of these parameters and leave the dialogue by pressing "OK" all the geometries that are assigned to this (now changed) pen will use the modified parameters during next laser marking process.

8. **Adding a hatch pattern**: Currently only the outline of the triangle is shown and only this outline would be processed during laser marking. To process the inner side of the triangle too, a fill pattern has to be added by using the Additional Geometry element "Hatch". It is symbolised by a purple-coloured tool bar icon that consists of several horizontal lines. When this icon is clicked as long as a Primary Geometry element (like the triangle) is selected, it is added to this element automatically.
Now the appearance of the geometry and the appearance of the application changes:
- the first panel on the main windows left hand side switches to show all the "Hatch" parameters
- the Element Tree on the main windows right hand side now shows the new hierarchical structure of the element with the "Hatch" as sub-element of the primary "Triangle" geometry
- the triangle within the drawing area is filled with some hatch lines
This hatch pattern now can be changed by modifying the values within the property-panel "Hatch" on the main windows left hand side.
PLEASE NOTE: after clicking the Hatch-button within the tool bar, the new geometry of the now added hatch/fill pattern is selected instead of the triangle! That means all modifications to the selected geometry (as described previously) now would apply to the hatch geometry only but NOT to its parents triangle! So in order to manipulate the whole element the parent of the hatch has to be selected – which is the base element generated in previous step.
9. **Save data**: To avoid the currently generated geometry gets lost, the project has to be saved now. To do that, select menu "Project" menu item "Save as..." and save these data at a suitable position using the .BEAMP format.
PLEASE NOTE: this project contains all the hardware definitions that have been done during configuration of the software in first steps. Whenever you change this configuration and then load this project again, the hardware set-up of the project is used instead of the changed one of the application. To avoid that after loading such a project, the menu "Project" menu item "Load default configuration" can be chosen to replace the projects hardware set-up by the local, desired one.
10. **SECURITY CHECK**: Next the scanner controller card together with a possibly connected laser will be accessed for the first time. That means it is opened and initialised and all connected equipment may start working now. Thus it is very important to ensure all security regulations are met and nobody can be injured and no damage can be caused also in case laser output or other motion starts spontaneously and unexpectedly!
11. **Prepare for laser marking**: Now the created geometry has to be sent to the connected laser and scanner system for processing it on some material. To do that the menu "Process" menu item "Mark" has to be selected. It opens the marking dialogue and tries to access the connected hardware. In case it is not possible to access the scanner controller card, an error message is shown and none of the buttons of the mark dialogue are usable (except the "Cancel" button to leave the dialogue). In this case you have to go back to the project settings as described in step 2 and to correct the set-up of your scanner controller card.
12. **SECURITY CHECK**: Next the laser and scanner will be accessed. Since there are some (laser) controllers available that are VERY sensitive to wrong laser signals, it is recommended first to check the output your scanner controller card produces. This can be done e.g. by using an oscilloscope instead of the real hardware. The target equipment should be connected only in case all signals are checked, correct and acceptable by the hardware. More than this it is recommended to repeat this step whenever the something within the complete set-up changed; changes may be caused e.g. by modified parameters, by software or driver updates.
13. **Start marking**: Now marking can be started by pressing the big yellow button (the one with the laser warning sign) in the middle of the marking dialogue. This starts sending of all laser and scanner data

to the scanner controller card so that it is able to output motion and laser control data synchronously. Such a marking operation can be stopped by pressing the big red button (with the STOP-symbol) at the dialogues right hand side.
PLEASE NOTE: this stop-button is not a replacement for a real emergency stop, it will try to stop the currently running laser process only via software which – in worst case – may fail for an example due to an abnormal problem or in case of a hardware failure. Thus there must be independent and working emergency stop equipment available in every case!

6.5 User interface

6.5.1 The Project menu

The menu “Project” gives access to different global, project- or application-specific operations:

- New project – create a new project, this deletes all possibly existing data and sets BeamConstruct to default configuration values
- Open project – open an existing .BEAMP BeamConstruct project which contains all required data like hardware settings, geometries, control elements, pen settings,…; when using this function, all data of a previously used project will get lost
- Insert project – load a .BEAMP project without deleting the current one: the hardware settings of the current project are left unchanged and the elements and pen settings of the new project are appended to the new one
- Load hardware settings – loads hardware settings out of a .BEAMH file while leaving possibly existing elements (control and geometry) and pen settings unchanged
- Load payload data – loads geometries, control elements and pen settings out of a .BEAMV file while leaving only the hardware settings unchanged
- Import – load vector data out of a file that is not a BeamConstruct project file (.BEAMP or .BEAMV), the newly loaded geometries are added to the current project
- Save project – save the currently loaded project using the same name, this operation overwrites an existing project without further notice
- Save project as – save the currently loaded project while giving it a (new) name by using a file dialogue
- Save with options – save the currently loaded project with the possibility to specify what this project may contain exactly
- Save hardware settings – save the current hardware settings into a .BEAMH file
- Save payload data – save geometries, control elements and pen settings of the current project into a .BEAMV file
- Project settings – open the global settings dialogue where all application and hardware related configurations can be done; for details please refer to section “6.6 Project Configuration”
- Pen settings – open the dialogue where all material and output related parameters can be configured; for details please refer to section “6.8 Pen Settings”
- Save as default configuration – save the current hardware and pen settings as default so that these parameters are loaded as default on next start-up of BeamConstruct or when a new project is created by selecting menu “New project”
- Load default configuration – when a process file was loaded, it possibly can contain an own hardware configuration which does not fit to the current device. Using this menu item the local default configuration can be loaded overriding the parameters of the loaded project

6.5.2 The Help menu

This menu gives access to different informational and service functions:

- About – displays information about software version, copyright and vendors homepage
- License – displays information about the current license, when your application pops up with an error message “Insufficient license” at some functions, please check here if your really have the license which is required
- Reset license – this function can be used to get rid of a software license which was retrieved online. After this function has been called, BeamConstruct is no longer licensed and runs in demo mode

- Check dongle – this is a service function which typically is needed in case of some problems with the license and the used dongle. It displays some information which may be useful for HALaser Systems' support, they will guide you to this function when it is necessary to use it.
- Generate bugreport – this function can be used in case of problems with the software to gather different data which are helpful to analyse these problems.
  Please note: this function collects different information which may include sensitive, personal or other confidential data. When you do not agree to share these data, do not use this function. Otherwise you agree that we get, store, analyse and read these data for an undefined time (but typically only as long as it is necessary to check your specific problem).
  In case of a problem with the software, please use this function to generate a bug report. When you experienced a crash, where the software has died completely and therefore this function is no longer accessible, please restart BeamConstruct, load the last project you used and where you experienced this crash and then call this function.
  This function creates a ZIP-file where you can specify where to store it. This ZIP-file contains the bugreport-data and has to be sent to the support HALaser Systems in order to check what causes the problems. HALaser Systems always provides a public key for encryption of e-mails and data, so you also can send us such a bugreport fully encrypted.
- Credits – displays credits regarding to people and companies which contributed things to this software

6.6 Project Configuration

When starting with a new project it has to be configured first so that the editing environment of BeamConstruct reflects the real processing conditions within production.

This can be done in global settings, the related dialogue can be found in menu “Project” sub-menu “Project settings...”. There a dialogue opens which offers several configuration options within different tab panes. The following parameters can be set and modified within these panes:

In tab-pane “User interface”:

- Visual grid size – this is a helper grid that is used and visible within the working area as an array of of points with a distance in X and Y direction as specified here
- Snap to Grid – when this check box is selected, all geometry that is drawn or re-positioned within the working area automatically snaps to a position that is defined by the visual grid; this option gives the possibility to position elements very exact
- Background Colour – this option changes the visual representation of the editor and does not have any influence on the generated data, it specifies the colour of the background within BeamConstruct's drawing area
- Working area Border – the working areas (as defined in scanner controller settings for every scanhead) are shown as rectangle within the drawing area of the application, here the colour of the bounds of related rectangles can be changed; similar to the previous option this one also does not influence the generated data
- Editing area border – beside the working area, which specifies the range a single scanhead can cover, an editing area exists which is a visual help showing the user where drawing is allowed for the current configuration. This area exists only once for a configuration, it can be equal or smaller than the working area of a single scanhead (in single head environments), it can include several scanheads working areas (in multihead environments) or it can be larger than the available scanheads working area (e.g. when it has to be used together with the functionality of Active Split Group).Here the colour of the bounds of this area can be changed; similar to the previous option this one also does not influence the generated data
- 3D mesh colour – this colour is used in 3D view only and specifies how an imported 3D model is drawn
- 3D support mesh colour – this colour is used in 3D view only and specifies how the support structures for a sliced 3D model are drawn
- Editing area upper left – specifies the upper left corner of the global editing area within the coordinate system of the application
- Editing area size – specifies the size of the global editing area within the coordinate system of the application

In tab-pane “Scanner” it is possible to configure the scanner controller(s). In case these components are set up properly according to the used hardware environment, it is possible to start marking and motion operations directly from within BeamConstruct:

- Scanner Controller Selection – this combo box has to be used in multihead mode where BeamConstruct controls more than one scanner card at the same time. The selection with this combo box influences the settings of all following parameters. Please note: in multihead and singlehead mode the first scanner controller always has to be set, it is not possible to start e.g. with settings for the second scanner controller and leave the first one empty.
- Scanner Controller – here a scanner card can be chosen out of a list of available scanner controllers; this list makes use of the related plug-ins that belong to the ControlRoom environment. As soon as a scanner card is selected for usage within BeamConstruct, it can be configured by pressing the button “Configure”. Although this configuration is done for the scanner controller card all the parameters like the working area size and position are used to set up BeamConstruct too.
  For a description of the available scanner controller plug-ins, their usage and parameters, please refer to section “5.3.3.7.2.13 External Laser Flow Objects” above
- Initialise on startup – when this option is set, the scanner controller card and – when available and

configured – image capture devices, motion-, laser- and process-controllers are initialised on startup of the software. Normally this is done when the Mark-dialogue is opened for the first time, but when this option is enabled, it happens when BeamConstruct is executed. In general it is not recommended to use this option since it causes possibly unnecessary accesses to the hardware. Only in very rare cases where some external equipment has to be initialised as early as possible (e.g. when some hardware needs to be heated up), this function should be used

- Home position – when this option is set, the scanner is moved back to the XY-coordinates given right beside this checkbox whenever a marking cycle was finished; using this function the head always ends up in a defined position no matter what the last marked vector really was.

- Pilot Laser – this section is related to the configuration of a pilot laser that does not perform marking operations but points to the position where the main laser would influence the working piece (as some kind of preview-laser). This pilot laser can be used out of BeamConstruct's marking dialogue directly (please refer below for more details); the related parameters are enabled only in case a scanner controller card was chosen and configured properly
- Output Port – here a port can be chosen that has to be used to turn the pilot laser on and off
- Output Number – here the bit number at the previously chosen output port has to be selected which shall be used to toggle the pilot laser;
  PLEASE NOTE: some scanner controller cards use some of their output bits for specific, reserved purposes, these bits should not be used for the pilot laser to avoid any unwanted and undefined behaviour. As a second precondition it is necessary to connect the pilot lasers on/off input with the output that was selected here.
- Timeout – here a timeout (in unit minutes) can be specified which turns off a pilot laser after this time. Using this function it can be ensured a user does not (accidentally) let a pilot laser run over hours or independently
- Invert Output Logic – inverts the output value for the signal that should turn the pilot laser on, this option needs to be set for some specific pilot lasers to invert their behaviour
- Always on – when this option is set, the pilot laser is kept on all the time, means it is not modulated and not turned off during jumps. Only before next real marking operation it is ensured the pilot laser is turned off since some laser types can't turn on the main laser as long as the pilot is still on.
- Correction Factor / Correction Offset – pilot lasers normally have a different colour and therefore a different wavelength than the main laser. In case some optics are used within the scan head, this may result in variances in the marked positions due to a different light refraction factor. Using these parameters such variances can be compensated. The correction factor is a factor that modifies the size of the pilot lasers output in X and Y direction while the correction offset modifies its absolute position

- Ready For Marking Signal – here a signal can be defined that is emitted on a digital output of the scanner card as soon as the software is ready for marking
- Output Port – here a port can be chosen that has to be used for the “Ready For Marking” signal
- Output Number – here the bit number at the previously chosen output port has to be selected which shall be used to toggle the “Ready For Marking” signal;
  PLEASE NOTE: some scanner controller cards use some of their output bits for specific, reserved purposes, these bits should not be used for this signal in order to avoid any unwanted and undefined behaviour.
- Invert Output Logic – inverts the output value for the ready-signal

- Mark dialogue opened signal – here a signal can be defined that is emitted on a digital output of the scanner card as long as the marking dialogue is open
- Output Port – here a port can be chosen that has to be used for the “mark dialogue open” signal
- Output Number – here the bit number at the previously chosen output port has to be selected which shall be used to toggle the “marking dialogue open”” signal;
  PLEASE NOTE: some scanner controller cards use some of their output bits for specific, reserved

purposes, these bits should not be used for this signal in order to avoid any unwanted and undefined behaviour.

- Follow-up time – here a time range (in unit seconds) can be specified the “mark dialogue open” signals stays turned on after this dialogue window was closed. When it is reopened again before this time elapses, the output stays active without interruption
- Invert Output Logic – inverts the output value for the active-signal

- Marking Active Signal – here a signal can be defined that is emitted on a digital output of the scanner card as long as marking is in process; this signal is NOT identical to the laser gate which turns laser on and off during such a marking process
- Output Port – here a port can be chosen that has to be used for the “Marking Active” signal
- Output Number – here the bit number at the previously chosen output port has to be selected which shall be used to toggle the “Marking Active” signal;
  PLEASE NOTE: some scanner controller cards use some of their output bits for specific, reserved purposes, these bits should not be used for this signal in order to avoid any unwanted and undefined behaviour.
- Invert Output Logic – inverts the output value for the active-signal

- Allow mark start via external signal – some scanner cards provide a special input for a “mark start” signal; when this option is set and the mark dialogue is opened, marking process can't be started not only by pressing the mark-button but also by setting the start-input of the used scanner card;

When more than one scanner/head was configured, it is possible to do some multihead-related configurations in tab-pane “Multihead”. When only one scanner controller/scanhead exists, these settings do not have any influence on marking operations:

- Preferred head usage – within this matrix it is possible to define which head should be used for what kind of operations. So as long this is really possible for a head and related geometries, the head then will be used for this operation. Here following options can be chosen for a head, search order for suitable heads during marking is from left to right:
  - Automatic (default) – the geometries are assigned to this head, which is suited best to process them. So in case of partial overlapping working fields the automatic operation tries to spread the geometries over the heads in a way so that as long as possible marking operations are done while switching as seldom as possible from one head to an other
  - Contour – the head should try to mark only contours. Only when a contour is located in an area of an other scanhead and outside of the working field of the selected scanhead, or when an other head in “Automatic” mode fits better, this option is ignored and an other scanhead will mark contours
  - Hatch – the head should try to mark only hatches. Only when a hatch is located in an area of an other scanhead and outside of the working field of the selected one, or when an other head in “Automatic” mode fits better, this option is ignored and this other scanhead will mark the related hatch

In tab panel Motion Axes it is possible to configure up to two different motion controllers with up to four independent axes each. Both possible motion controllers are identified via names OUT2 and OUT3, which are identically to the outputs of the related BeamConstruct2Control plug-in. So when a BeamConstruct project is executed out of a ControlRoom environment using this plug-in, the hardware settings given here are ignored and the motion data are emitted at these plug-ins outputs. When the motion controllers have to be driven by BeamConstruct directly (or by the BeamConstruct-HMI-plug-in) here a valid configuration is required for at least one of them:

- Motion Controller OUT2 or OUT3 – this combo box can be used to select a motion controller that would be accessed whenever a motion control information is assigned to output OUT2 or OUT3 (please refer to the description of the Primary Element “Motion” and other motion-related functionalities below). This selection list makes use of the motion plug-ins of the ControlRoom environment, so for a description of the related plug-ins parameters please refer above. The parameters that are set for such a motion controller are NOT handed over to BeamConstruct fully, so when some of the axes of a motion controller are not enabled here within its settings, they are still be

usable and configurable within the related BeamConstruct elements. This is necessary to keep all options and possibilities for a user that wants to export a BeamConstruct project and use it within a ControlRoom project where more/other axes may be available.
For a description of the available motion controller plug-ins, their usage and parameters, please refer to section “5.3.3.7.2.15 External Motion Flow Objects” above

- Reference axis on start-up – here these axes can be specified that have to be referenced on application start-up or on first opening of devices automatically
- Bind axis to slices Z-position – this function is related to marking of sliced 3D data. This option should be set for all these axes, which perform a Z-movement which has to have the same height as the slice thickness of a 3D slice group which is processed. Then the height of the slice is set as motion distance automatically – no matter, what value the related element is configured really. So when a 3D model is sliced with different parameters, the motion settings of a globally configured process project (an external project which is executed between slices automatically) do not need to be changed manually. Only the sign of the motion distance of the motion element is kept, means when a negative movement is specified, the negative movement direction will be performed – but using the slice height.
  This option can be set for all axes which in any way are related to a motion operation which depends on the height of a slice.
  Following preconditions have to be met in order to let the motion distance of an axis automatically be set to the current height of a slice:
  - the option “Bind axis to slices Z-position” needs to be set
  - within the related motion element the axis motion needs to be set to relative
  - a positive or negative relative motion distance needs to be set
  - the motion element is part of a sliced 3D group (as child of the related slice) or it is part of a project which is processed between slices (please refer to parameter “Process between slices” below)
- Axis alias and orientation – here for the default axes A, B, C and D an own name can be set that is shown e.g. within mark dialogue to identify the axis; additionally for planar axes it can be specified in which direction movement is done, the setting made here influences what kind of arrows are shown for axis movements in mark dialogue
- Key controlled – configure single axes for control via keyboard operation. When this option is enabled, these axes can be moved out of mark-dialogue or “Motion”-panel by pressing ALT-key plus one of the cursor keys up/down/left/right. Then these axes are moved by the step distance and the speed as entered in related input fields.
  Here it is possible to configure up to two axes for this operation mode. When the related checkbox is set for a longitudinal or vertical axis, it can be controlled by they key-combination ALT plus cursor up/down. When the checkbox is set for a horizontal axis, it can be moved step-wise by the key-combination ALT plus cursor up/down in motion panel or mark dialogue.
- Don't show warning when referencing axes – when some axes are configured to perform referencing/homing operation on start-up, this may result in an unexpected, surprising and therefore possibly dangerous movement. Because of this prior to referencing operations an information is shown the user has to confirm in order to start the referencing operation. When this option is set, this information is NOT shown, referencing of axes starts immediately and WITHOUT ANY WARNING! Thus this option has to be used carefully and only in case the whole machine is secure and covered completely.
- Process between slices – here a path to an existing project can be given that is used when 3D meshes are sliced; this project can contain a processing sequence that is performed between every slice

The tab-pane “Other hardware” contains the possibility to configure other external and devices which may be required for operating a machine which is controlled by BeamConstruct:

- Image capture – this gives the possibility to choose a source for a live background video which is displayed in drawing area behind all vector and bitmap data. This live image can be used to adjust the current marking project to the working piece that is shown by a connected camera.
  To have a background image just as pattern for a laser project, it is recommended not to use such a live image but to import a still image and use it as light-table background (via menu item “Project” → “Import”, please refer to description of import functionality below).
  For a description of the available image and video capture plug-ins, their usage and parameters,

please refer to section “5.3.3.7.2.10 External IO Operation Flow Objects” above.

- Position / Size – using these input fields the position and displayed size of the captured video can be specified, it expects the same coordinate system and orientation like it is used for definition of working area.
  Beside of entering values for position and size of the live background image manually, it is also possible to right-click into the drawing area, select option “Modify live background” and then drag and scale the background image like any normal geometric element in order to place and stretch/shrink it as desired
- Display image on manual activation only – when this option is set, the image is not shown all the time, it has to be enabled manually and can be disabled afterwards to continue without a live background image; this can be done via vision menu or via the image button in mark dialogue
- Stop image capture on motion operations – this option may be useful on systems with limited computing power, when it is set, capturing of image data is stopped as long as an external axis is driven
- Stop image capture during marking operations – this option may be useful on systems with limited computing power, when it is set, capturing of image data is stopped as long as marking is in progress

- Scanhead Selection – this combo box is dedicated to multihead configuration too, different to the other combo box which can be found in “Scanner” tab-pane, it is assigned to a scan head. This differentiation is made because one scanner controller / one plug-in defined above may access several scan heads independently, therefore head-specific parameters have to be set separately. The selection made with this box belongs to the following two parameters:
- Additional laser control – here a laser controller plug-in can be selected that communicates with a laser independent from the scanner card. As soon as a laser controller is selected for usage within BeamConstruct, it can be configured by pressing the button “Configure”. This function is useful in case a laser is used that requires control via a different communication channel that does not exist on a scanner controller card. This can be e.g. an Ethernet connection or a serial interface which is used to send control commands to the laser.
  For a description of the available scanner controller plug-ins, their usage and parameters, please refer to section “5.3.3.7.2.13 External Laser Flow Objects” above.
- Z-Shifter – here a plug-in can be selected that communicates with a special motion axis to be used for shifting the Z-position, expanding the beam or shifting the lasers focus independent from the scanner card. As soon as such as Z-Shifter is selected for usage within BeamConstruct, it can be configured by pressing the button “Configure”.
  For a description of the available scanner controller plug-ins, their usage and parameters, please refer to section “5.3.3.7.2.15 External Motion Flow Objects” above.

- Process Controller – here a plug-in can be selected that is able to control additional process parameters (beside the laser marking related ones). Possible parameters are pressure, temperature, gas flow and others where the current value is read from an external device. Depending on the possibilities and settings of the used plug-in, other devices can be controlled or switched and/or the currently running marking process can be halted or continued. Also depending on the used plug-in, the pen settings dialogue may be extended by an additional tab-pane as soon as such a process control plug-in is enabled. This additional tab-pane then contains parameters which are handed over to this plug-in during runtime and dependent on which pen is currently selected.
  As soon as such as Process Control plug-in is selected for usage within BeamConstruct, it can be configured by pressing the button “Configure”. There all parameters can be set according to the capabilities of this plug-in.
  For a description of the available image and video capture plug-ins, their usage and parameters, please refer to section “5.3.3.7.2.10 External IO Operation Flow Objects” above.

Tab-pane "Misc" contains options and configuration possibilities that do not belong to any of the previous ones logically:

- Enable IOSelect-Mode – this checkbox can be used to activate the IOSelect functionality (please refer below) and to enable the settings that belong to it:

- Numbered Projects directory – here path to a directory has to be specified that contains the project data to be loaded depending on the current input state
- Card 1 Input Port – the digital input port that has to be used for external selection of next project; this selection is important in case a card provides more than one digital input port; as specified by the name of this parameter, only the first scanner card can be used to read the IOSelect input data from
- Auto-Start Marking – when this option is set, marking of IO-selected project starts automatically after mark-dialogue was opened. Attention: this function may be dangerous and should be handled with care! The start of a marking operation directly after opening the mark-dialogue and without further actions may be unexpected and surprising for the user!

- Auto-optimise vectors on marking – BeamConstruct is able to handle a very high accuracy internally – which is in very most cases much higher than what the controller hardware and scanhead are able to produce on output. When this option is set, the vector data are analysed before they are sent to the scanner controller(s) and all data that are beyond the resolution of the used hardware are removed. Depending on the data to be marked this can save nameable amount of marking time and data.
- Auto-slice 3D-meshes on change – when a 3D-model is imported to be sliced into layers (please refer to section 6.11.2) there are several operations that require reslicing of these data. Depending on the size of the 3D model and the thickness of slices this can be a time-consuming operation and therefore has to be triggered by the user manually. This behaviour can be changed, when this option is set, reslicing of the 3D mesh is done automatically after every operation that requires it. This option should be used only in case the used 3D models are simple enough or the computer hardware is powerful enough, elsewhere it can result in long lasting delays during editing of 3D models and sliced data.

- Show Tab-Panes in Mark-Dialogue – this area consists of a set of checkboxes that correspond to the different tab-panes shown in Mark-Dialogue. Here it is possible to uncheck some of them in order to hide the related tab-panes. This function can be used to not to show unused functions to the end user

- Custom Dot Style – the following parameters are responsible for defining a style for a customised dot layout that is used by several geometric elements; this drop-down box selects the type of geometry, here polygon (includes a circle) or spiral can be chosen to be used as replacement for simple dots
- Edges/Segments – specifies the number of the edges or segments the custom geometry has to consist of
- Outer Radius – sets the size of the custom geometry
- Inner Radius – this parameter is used only in case of a spiral, it specifies the inner size of it
- Slope – here the same is true, this parameter applies to a spiral only and can be used to set the steepness the spiral grows with
- Draw CCW – here the drawing direction of the custom geometry can be inverted
- Inside To Outside – when this option is set for a spiral, it is drawn starting from its inner point

- The sub-section “Limits” gives the possibility to configure allowed ranges and behaviour of some pen parameters, here the minimum and maximum value that can be specified within pen settings dialogue can be set. Beside of that the behaviour of the related slider can be changed, when check box “Logarithmic” is set, the sliders no longer act linear. Here allowed range and behaviour of following values can be configured:
  - Pen Frequency
  - Jump Speed
  - Mark Speed

  Option “Allow unrestricted laser on/off delays” disables some logic checks in pen settings. Normally the delays applied to a pen have to follow some logic rules which are automatically checked when the user modifies their values. When this option is set, these logic checks are disabled and illegal

laser delay values can be set for a pen. Since this may result in some very strange and undefined behaviour during marking, this function should be used only in some very rare cases where it is clear what happens when such illegal laser parameters are used!

- Parts Counter / Enable – with these two parameters some processing-related limits can be set: this option enables the parts counter; a function which counts how many parts already have been processed and informs the user when a limit was reached; in this case the user has to confirm this limit and needs to reset the counter to continue marking; confirmation and resetting can be done in mark dialogue
- Default part number – here a number of default parts can be specified that is used for the parts counter, it can be overwritten by the user in mark dialogue and requires the parts counter option to be enabled
- Power down laser / Enable – enables a function which gives the possibility to power down a laser when it is unused for a given time
- Power down idle timeout – here a value in seconds can be set which specifies after what idle time a connected laser has to be turned off; this requires the power-down option to be enabled

All parameters from this dialogue are stored within local default settings and within every .BEAMP project file. Thus it may happen a project file is loaded that contains different hardware settings and therefore is not compatible to the local system. To avoid such troubles two functions can be found in menu “Project” that handle the local default data:

- Save as default configuration – saves the current set up as default, whenever the application is started or a new project is generated, this configuration is used automatically; this not only includes the settings done within the project settings but also the number of pens and pen settings
- Load default configuration – loads a saved default configuration, overwrites the current project settings and appends the loaded pen definitions to the currently used ones

6.7 Machine Configuration

There is a set of machine-dependent parameters which are not stored within each project file or in default project configuration, but locally on each machine where BeamConstruct is installed at. These parameters can be configured via related dialogue in menu “Project” sub-menu “Machine settings...” which offers several configuration options within different tab panes. The following parameters can be set and modified within these panes:

In tab-pane “User interface”:

- Expand / Move entity tree to tab panel – this option saves some space in X-direction; the entity tree that is shown on main windows right hand side normally will be moved to the left side and becomes a normal tabbed panel there; changes to this parameter do require restarting of the software in order to take effect
- Security / No warning message at start-up – in general it is not recommended to use this option, it makes sense in some very rare cases only: when this checkbox is set, the warning message about the possible dangers of this application is no longer shown. This means, prior to turning of this warning, other ways of informing the user have to be established in order to avoid unwanted and possible dangerous usage of this option.
  Changes to this parameter do not require saving of the current configuration as default one, the related information is stored automatically and always only locally, it does not belong to a project file which can be given away.
- No license progress bar on start-up – on application start BeamConstruct checks for a valid license. Depending on the speed of the used computer this can take some seconds. During this time a progress bar is shown. When this option is set, this progress bar is no longer used but license checking is still done – so with this option the delay at application start up is still there but without a visual feedback.
- Switch back to old OpenGL 1.1 – the 3D view within the application makes use of modern OpenGL for fast drawing also of complex geometries and 3D models. On some graphics hardware and in case of poor quality of drivers of such graphics card this can cause issues – starting from wrong or corrupt displaying of models and ending up in a crash of the software as soon as the 3D view is opened. In such cases it is recommended to turn on this option to switch back to the slower OpenGL version 1.1
- Language – by default this parameter is set to “Automatic”, means the application chooses the language the underlying operating system is configured for. When other translations are available, they are listed in the related combobox and can be chosen to be used instead. Then on next start of the application, this language is used for the user interface.
- Menu item shortcuts – the area below of these parameters contains an editor that gives the possibility to edit the shortcuts of BeamConstructs menu items. On lower left hand side the menu item to be edited can be selected in a tree view. Next right beside that tree the new shortcut has to be typed in field “New Shortcut” and then can be added using button “Add”. The new shortcut is shown in the list right beside the menu tree view and can be removed from there using button “Remove”. After the settings dialogue is left with “OK” these new menu shortcuts are applied to the menus.
  These settings are locally only and do not become part of a project.

The “Smart Factory” tab-pane offers functionalities which are related to machine integration into automated production lines:

- Machine name – here a name can be set which identifies a software installation/a machine uniquely. By default an auto-generated name is set here which can be replaced by a human-readable value. This name is used for all Smart Factory related functions, e.g. within all Smart Interface state messages to identify the machine it comes from and within the relevant Hermes messages that require an unique identification too.
- Enable Smart Interface – when this checkbox is set, the Smart Interface is enabled and the software can be integrated into a Smart Factory / Industry 4.0 environment. For details regarding this interface please refer below
- Enable MQTT connection – when this checkbox is set, state messages are sent to a suitable broker

using the MQTT protocol

- Broker – here IP and port number of the broker to publish the MQTT state messages to has to be set
- Topic basename – using this field a topic for the MQTT messages can be configured. It will consist of a base-part of the topic which always is expanded by the machine name configured above
- Enable Hermes automatic control – when this checkbox is set, support for machine integration according to the Hermes-standard is enabled (for details please refer to https://www.the-hermes-standard.info). This enables several configuration parameters which can be used to specify connection to other machines within the same line. Here it depends on the availability of previous and/or next machine if BeamConstruct acts as end of line machine, begin of line machine or something in between. If the next or previous machine does not connect according to the Hermes specification, BeamConstruct automatically switches to end/begin of line mode.
- Previous machine – here IP and port number of the machine which transfers working pieces to BeamConstruct can be configured
- Next machine – here the port number of the machine BeamConstruct transfers working pieces to can be configured. Since the next machine connects with BeamConstruct actively, here no IP has to be specified. Instead of this the local IP must be known at the next machine.
- Conveyor axis – using the following comboboxes it can be specified which motion controller and which of its axes has to be used as conveyor to feed the working pieces. Here also the movement direction of the conveyor can be set.
- TransportFinished distance – when the conveyor axis moves to transport a working piece to the next machine, a TransportFinished command has be sent to the next one, as soon as this transport is complete. Using this field the motion distance can be specified which has to be used to move a working piece out of the current machine. This parameter is used only when BeamConstruct does not act as end-of-line machine.
- Enable Hermes remote configuration – according to the Hermes standard there is a possibility to send configuration data from a remote host and to overwrite local information as they may have been specified here. When this checkbox is set, overwriting of the local configuration is allowed and possible. When it is not set, the Hermes-configuration of this machine can be changed only by direct manual interaction.

In tab-pane “Users” access to functions can be limited depending on logged-in users, their roles and privileges. By checking the box “Enable user management”, this functionality is activated and some predefined settings are provided. For all existing, preconfigured users the password is similar to the username and should be changed prior to operating it on the target system.
The users-tab-pane lists the available users on the left hand side. By selecting one of them, the related privileges and settings are shown on the right hand side of the pane. Once a user is selected, its privileges can be changed, the login and full name as well as the state and the comment can be modified. To activate such changes after setting them, the button “Apply” has to be pressed, elsewhere the changes of a users parameters are not activated.

The check-boxes on upper right side of this tab-pane define what a user can do after logging into the software successfully:

- Supervision – this is a special privilege that applies only to user “Supervisor”, as long as this user is logged in, all functionalities are enabled, no restrictions exist
- Manager Users – a user with this privilege is allowed to modify the users and its privileges within this tab-pane; this should be combined with the “Modify Settings” privilege in order to give the related user access to the configuration dialogue
- Exit Application – this privilege gives a user the possibility to leave the application
- Modify Settings – a user with this privilege is able to enter the project configuration dialogue and to change the global parameters and settings of an installed software
- Edit Geometry – when this privilege is set for a user, this user will be able to modify loaded geometry and to edit the data of the currently loaded project
- Load Project – this gives a user the possibility to load an other project, this privilege is similar to “load recipe” like it may be named in other software

- Save Project – a user with this privilege is allowed to save the current project to disk; this privilege should be combined with an other privilege that gives the possibility to change the project settings, to edit the loaded project or to load/insert an other project, elsewhere it would not make sense for this user to save an unchanged project only
- Control Marking – when this privilege is set, a user is allowed to start and stop the marking process

These user data are stored in the default application data directory of the used operating system within a sub-folder “beamhome”. The exact position depends on the used operating system. The data are neither encrypted nor protected, so deletion of the directory and its contents would disable the user management completely. Thus it has to be ensured by using the operating systems functionalities that no user is able to access this folder without permission (e.g. by disallowing users to exit the application and by locking the whole system as long as BeamConstruct is running). This is done by intention to offer the possibility to restore a system just by deleting the “beamhome” directory when some relevant passwords got lost.

In case the user management was activated, on next software start-up all functions are disabled until a valid user logs in by using the correct password. Logging into the software can be done by pressing the user-symbol in toolbar: 

6.8 Pen Settings

Every element that is created within a BeamConstruct project has a pen assigned to its geometry (or for scanner bitmaps to its bitmap data). This pen contains all information the related element will be marked with: scanner speeds, delays, laser power and frequency and others more. These parameters can be set within the pen settings dialogue that opens when the menu “Project” menu item “Pen settings...” is selected.

On top of that dialogue the head can be selected this pen applies to. This is important for multihead environments where more than one scanner controller/more than one independent working area is used. There will always exist the same number of pens for all heads but the same pen number can have different parameters for different heads.

Below of this a pen can be chosen for editing, the same combo box can be used to create a new pen, in this case the option “Append new pen...” has to be selected there. After selecting an existing or a new pen all its parameters can be changed within the lower part of the dialogue. It is separated into some sections that contain different categories of pen parameters.

All tab-panes that hold these categories have one common functionality for selected parameters: by pressing the button “To all pens” a parameters value can be applied to all pens that currently exists. So in case a specific pen parameter has to be changed globally, with this function it is not necessary to edit it manually within every single pen.

PLEASE NOTE: there is no feedback from a possibly connected laser or scanner equipment to this dialogue, so the application can't know if a value is suitable for the connected equipment or not. Although there are some logical and range checks that try to avoid heavy mistakes it is up to the user to keep all parameters within valid and useful ranges.

6.8.1 General Pen Settings

This panel contains some basic pen parameters. On top of it an input field can be found that can be used to rename the current pen, there its name can be changed and the new name can be applied by pressing the return key.

Beside of that the following parameters can be edited here:

- Power – the output power in range 0..100% the laser has to be driven with
- Frequency – the frequency in unit Hz the laser has to be driven with; the allowed frequency range can be configured within project configuration dialogue
- Mark Speed – the speed of the scanner that is used during marking (motion while laser is turned on); the allowed speed range can be configured within project configuration dialogue
- Jump Speed – the scanners speed that is used during jumps (motion while laser is turned off); the allowed speed range can be configured within project configuration dialogue

- Spot Size – the diameter of the lasers spot on the surface, this value should be as exact as possible because it is used for different calculations
- Draw Off-Movements – this option is related to the visual representation of a project within BeamConstruct only; when this option is set not only these parts of the geometry are shown within the drawing area where the laser is turned on but also the jumps where the laser is turned off; this option can be helpful to optimise a project and to re-arrange elements to perform as less jumps as possible
- Use as default Pen – the pen where this option is set for is used as default pen for all newly created geometries. More than this, whenever a new pen is created the parameters of this pen will be set to the values specified for the default pen. Within a project only one pen can be default, so whenever this option is selected for a pen, it is disabled, resetting this option can be done only by choosing an other pen as default one.
- Use for Pilot-Laser – the parameters of this pen are used to control the pilot-laser. This means the speed and delay settings are used whenever the pilot laser is started. Since this laser can be turned on and off only, all values like power, frequency and some delays are disabled, they are not relevant for the pilot laser. Within a project only one pen can be used for the pilot laser, so whenever this option is selected for a pen, it is disabled. Resetting this option can be done only by choosing an other pen for the pilot laser.
- Pen On-Colour – this colour is used within the drawing area for all geometries
- Pen Off-Colour – this colour is used when “Draw Off-Movements” are enabled to show the laser jumps within the drawing area
- Use for coloured scanner-bitmaps – when this checkbox is set the current pen is used for coloured scanner bitmaps; this option should be enabled only
  1) when the current pen parameters cause a coloured result on current material
  2) when the colour of this result is specified using button “Colour after marking”; PLEASE NOTE: setting the resulting colour requires a calibrated monitor in order to have exact results

6.8.1.1 Pen Parameter Wizard

It is sometimes difficult to find parameter-combinations that fit to a special material or application. Thus the pen dialogue offers a special wizard that creates a matrix out of different values where the user can choose a suitable result from. To open this wizard, the button “Parameter Wizard” on upper left corner of the pen settings windows has to be pressed.
There a window opens where a 4x4 to 10x10 sized matrix of small marking areas is symbolised by a matrix of buttons. On upper part of the window the parameters to be used have to be selected and their range to be checked can be entered. These values are spread over the matrix in order to find best parameters. Next the yellow “Mark Start” button right beside the button matrix has to be pressed in order to start output to the material. Alternatively it is possible to use the button “As project” in order to not to mark out of the pen parameter wizard directly but to use the generated pen parameter matrix as current project which then is editable like every normal BeamConstruct marking project.

ATTENTION: by pressing the mark-button a laser operation is started! This may operate with up to 100% power and therefore is potentially dangerous! Prior to opening the pen parameter wizard and prior to starting output out of this wizard all security measures have to be ensured!

As soon as marking has finished, the area with the best marking result has to be chosen from marked material and the related button in pen parameter wizard has to be pressed.
To make it easier to find the correct position there is one circle and one triangle used for marking, these symbols identify orientation and direction of the matrix uniquely. So to find the position of the best result it is possible to first count from triangle to circle until the correct column is found and then count the row number. When the same is done in pen parameters button matrix of the wizard dialogue, the exact position is found and this button can be pressed. Such a button-press opens a popup-menu where one of the following options can be chosen for further processing:

- Refine this result – some values close to the chosen position are used to create a new matrix for a next, more accurate calibration step
- Go back from this result – this option performs the opposite operation of the previous one, it creates

a new parameter matrix with a larger range of values and should be used when a parameter range out of a refined result or when the initial parameter range does not give expected marking results

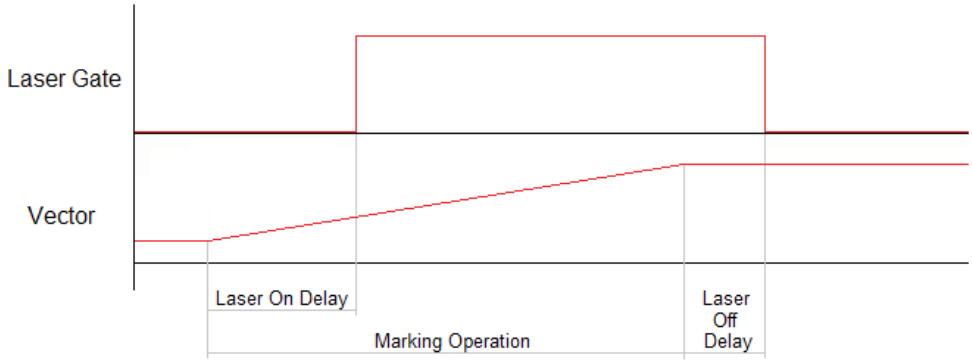
- Use parameters for pen – when this item is chosen the parameter wizard will be closed and the found parameters that belong to the selected field are set to the current pen.

For the other marking parameters apart the selected ones the pen parameter wizard always used values of the currently selected pen, only the selected ones are varied and modified by this function.

6.8.2 Scanner Pen Settings

This tab-pane contains all scanner-related pen parameters:

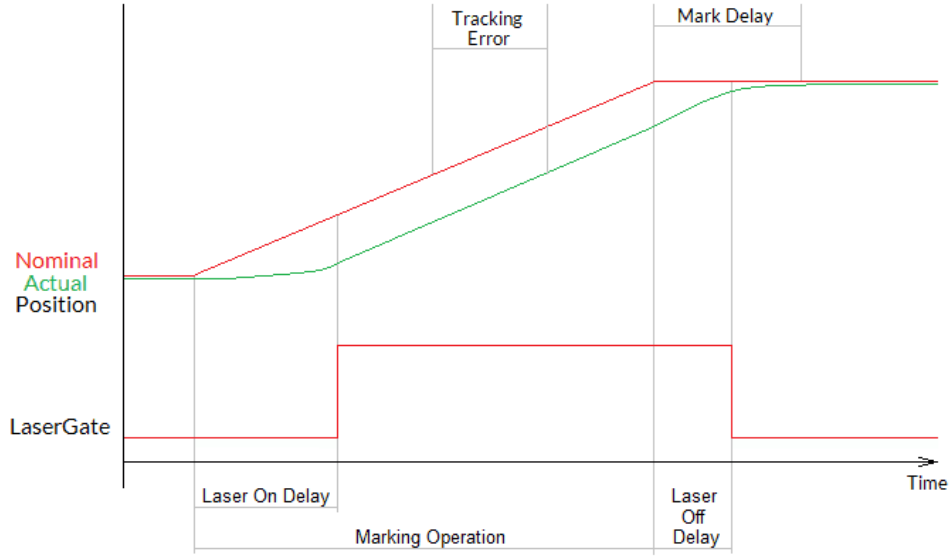
- Laser On Delay – the time that is set between start of a mark movement and the moment the laser is switched on; so this parameter specifies if and if yes how much the laser should be turned on delayed in case a marking operation is started
- Laser Off Delay – the time that is set between the end of a mark operation and the moment the laser is turned off; so this parameter specifies how much turning the laser off has to be delayed in case a marking operation ends



- Dot-length – this is a convenience function which shows and calculates the number of pulses a dot consists of with the current settings. The value given here depends on the frequency and the current laser on and laser off delays.
  So when the value in this field is changed, the corresponding laser off delay is changed automatically. On the other hand, when frequency, laser on delay or laser off delay are changed, the number of pulses shown in this field are changed according to the modified values.
- Jump Delay – time the scanner waits when a jump was executed; this delay applies at the end of a jump to let the scanner finish its movement

- Mark Delay – time the scanner waits when a marking operation was executed; this delay applies at

the end of a marked line to let the scanner finish its movement

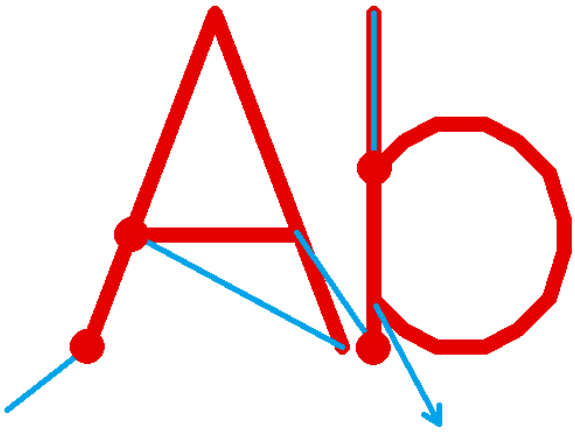


- Polygon Delay – the time the scanner has to wait for every corner within a polygon when the laser is turned on and stays turned on
- Use variable polygon delay – this option belongs to parameter Polygon Delay. When it is set, there is no static delay used in every corner of a polygon, but a variable one which depends on the current angle between two lines of the polygon. The sharper the angle is, the longer the used polygon delay becomes. This option is available only when a scanner controller card is configured which supports this feature.
- Auto-generate delays – the delay values described above are essential to get high quality marking results also in case of an as high as possible marking speed. Since the evaluation of the correct values is a difficult and time-consuming process, BeamConstruct offers a possibility to set some default values which can be used as starting point for optimising these delays.
  After pressing this button, a dialogue opens, where the used scan head can be selected out of a list of predefined types. Then some base delay values are generated for this scan head.
  In case a used scanhead is not listed there, option “Other” has to be chosen, which requires to enter one single value for delay calculation: Every scan head vendor should provide a value named “time lag”, “tracking error” (or something similar) within its technical documentation. This value describes the specific inertia of this scan head and needs to be entered here.
  When this dialogue is left by pressing “OK”, BeamConstruct calculates some basic delay parameters that fit to the scan heads mechanical properties and often can be used directly. Alternatively these delays can be used for further optimisations for these applications, where a more strong and exact set-up of the scan head is required.
  For more details about the purpose of all these delays and their influence please refer to your scan heads or scanner controller manufacturers manual.
- Set Defaults for Bitmap-Marking – when marking scanner bitmaps instead of vector data, some very specific pen parameters have to be used that follow other rules. In such cases it is recommended to use this button, it changes the current pen parameters including the scanner delays to values that are suitable for marking scanner bitmap elements.
- Wobble Frequency / Wobble Amplitude – when wobble is enabled the linear movement of the scanner during marking is distorted by an additional frequency that lets the scanner oscillate around the predefined position. This functionality can be used e.g. to warm up larger areas of a material, for welding or similar things. To enable wobble both parameters amplitude and frequency have to be greater than 0. The wobble frequency specifies how fast to oscillate and the wobble amplitude how much to vary over the original position. Wobble has to be supported by the used scanner controller card in order to have any effect during marking.

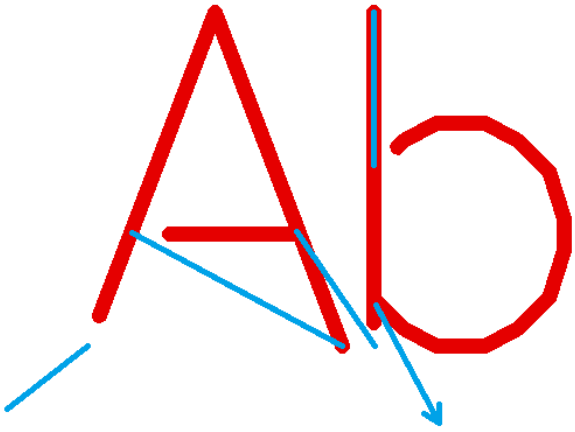
The given delay values have a direct influence on the marking result and need to be adjusted carefully. Wrong delay values can result in inaccurate geometries, burn-in points at some specific positions and/or unnecessary long marking times. Out of the visible result it can be evaluated which delays are to small or too

big by checking following rules:

Parameter	**Value too big**	**Value too small**	**Remark**
Laser On Delay	First part of a mark vector (following a jump vector) is not marked	Burn-in point at the beginning of a mark vector that follows a jump vector	See pictures below
Laser Off Delay	Burn-in point at the end of a mark vector and before the next jump vector	Last part of a mark vector (before a jump vector) is not marked	See pictures below
Jump Delay	Mark time is longer than necessary	Geometry is damaged at the end of jump (transient oscillation)	
Mark Delay	Mark time is longer than necessary	The end of a vector is turned towards the direction of the following jump vector	See picture below
In-Polygon Delay	Burn-in points at the corners	The corners of the polygon are not sharp but rounded and at wrong position	

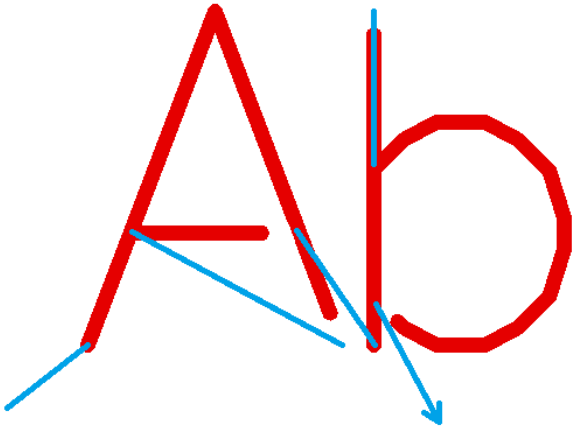


Laser-on delay is too short, laser is switched on too early while mirrors are still slow or accelerating, thus it fires on the same position for a too long time which results in burn-in points

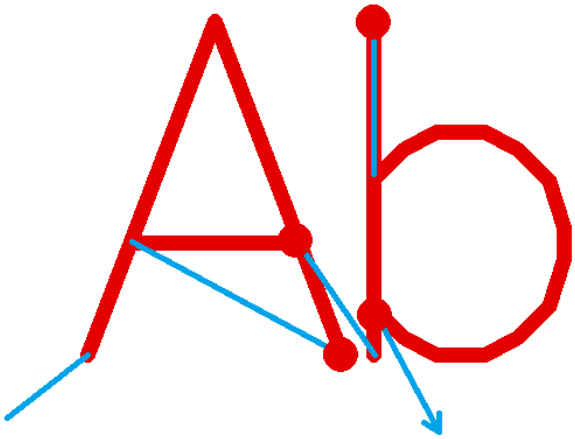


Laser-on delay is too long, laser is turned on too late while scanhead already started marking of the next

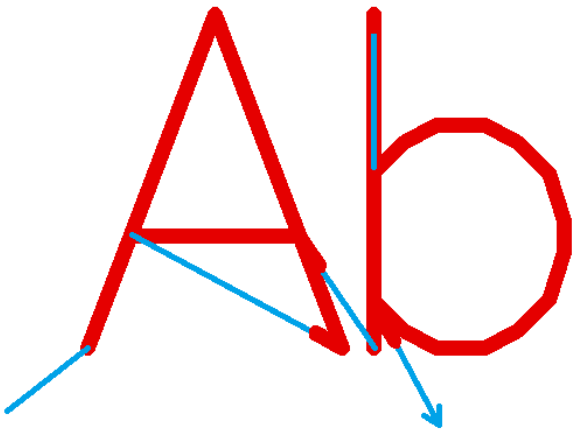
vector, thus parts of the mark geometry are missing at the beginning



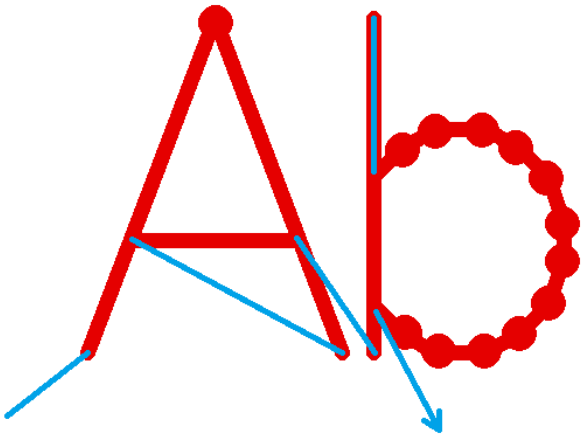
Laser-off delay is too short, laser is turned off before the end off the current mark vector is reached, thus parts of the mark geometry are missing at the end



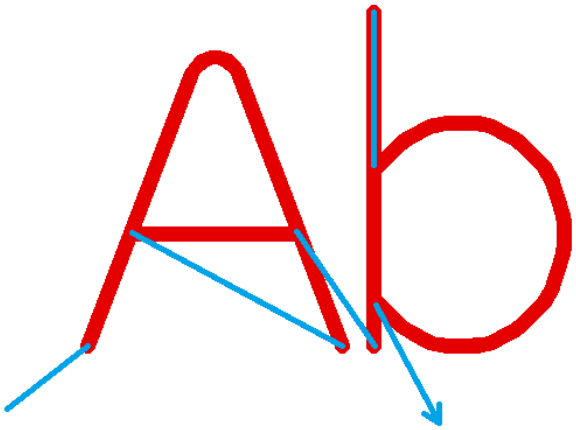
Laser-off delay is too long, laser is turned off too late while end point of the vector is already reached and mirrors of the scanhead slow down, this results in burn-in points at the end of the mark vectors



Mark delay too short, motion of next jump vector is started too early and before end position of previous mark vector was reached, thus the laser is still on when motion for the following jump vector is performed



In-polygon delay too long, scanner stops for a too long time at the position of a corner point within a polygon, thus burn-in points appear at these positions.



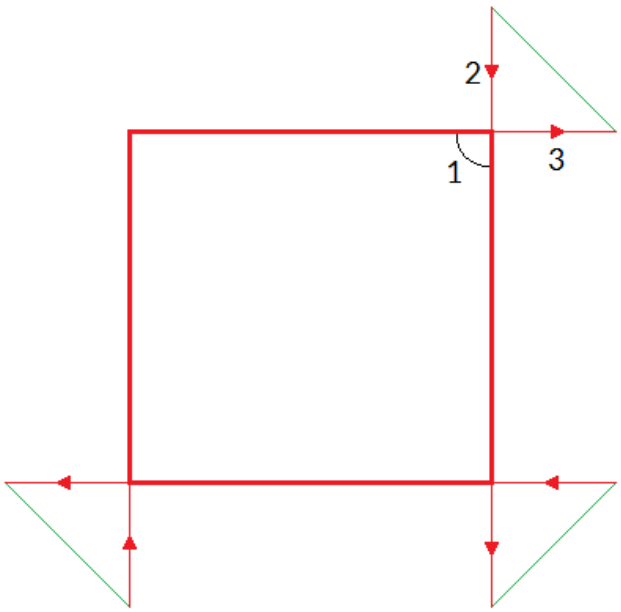
In-polygon delay too short, mirrors are not able to arrive at the positions of the corner points within a polygon before the next mark vector starts, thus the whole geometry is rounded, the vectors are pulled into direction of the next one

6.8.2.1 Sky Writing

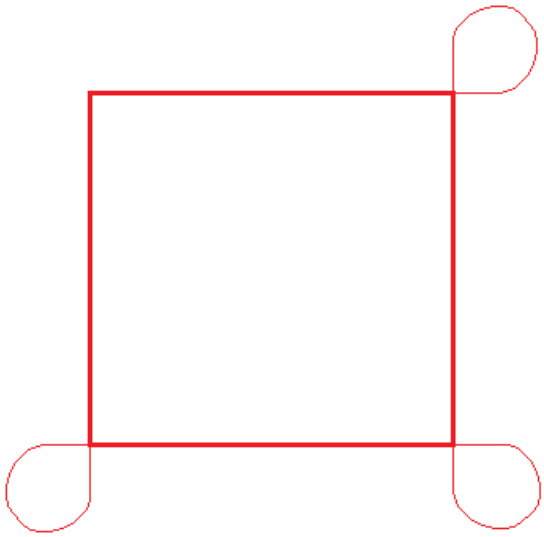
As described in previous section, it is necessary to apply delays in order to have clear and sharp corners within polygons. Depending on the used hardware and its capabilities these delays can become quite long, slowing down the total marking speed. To avoid this, the “sky writing” mode is available (on selected scanner controller hardware which supports this feature or has required capabilities). When this mode is active, the configured polygon delays are no longer used which waste some marking time. On the other hand the vectors to be marked become longer which itself again needs some time. So it depends on the used hardware and its specific capabilities if sky writing really saves time during marking.

Sky writing avoids delays and rounded, blurred corners by starting vectors earlier, ending them later and turn on/off the laser at exactly the position where the polygons line begins/ends:

In this picture the thick red lines symbolise the marks where laser is turned on. The thin red lines show the movements where vector lines are extended by a given length but where laser is already turned off. For a better understanding small arrows in the jump lines show the marking direction. The thin green lines show the jumps, only at these positions the chosen jump speed is used, all other movements are done with mark speed.



Here each line of the rectangle starts earlier, turns on the laser at the point where the polygon line has to begin, turns it off at the point where it has to end and then continues a bit in the same direction. Next a jump is done to the position where the next extended vector line has to start. Now when the scanhead would distort vectors because there is no more in-polygon delay active, this distortion would happen outside the rectangle and at a position where the laser is turned off. Thus the distortion would no longer be visible in the marking result when the lengths given for the polygon are long enough:



In this image the real scanner movements are shown. Here the movements at the corners are rounded and distorted due to the high speed, but these distortions do not influence the marking result since they happen when the laser is already turned off. In this specific example the lengths the lines are extended by could be shortened because there is still a longer straight line before and after every line.

Sky-writing requires following parameters to be configured:

- activation threshold – marked with a “1” in picture above: this is a value that specifies at which angle between two lines within a polygon the sky writing mode needs to be activated, if an angle is equal or smaller (means sharper) than this value, sky writing is used to keep the marking result clear, for bigger (means flatter) angles the normal in-polygon delay (as described above) is used
- fade-in distance – marked with a “2” in picture above: this value specifies the length a line has to be extended with at the beginning, means this is the size which is added before laser is turned on. This value should be as small as possible in order to not to waste marking time. At the borders of the current working area it may be possible this value can't be applied fully because the starting point would be located outside of the valid range of the scanhead.
- fade-out distance – marked with a “3” in picture above: this value specifies the length a line has to be

extended with at the end, means this is the size which is added after the laser was turned off. This value should be as small as possible in order to not to waste marking time. At the borders of the current working area it may be possible this value can't be applied fully because the end point would be located outside of the valid range of the scanhead.

6.8.3 Bitmap Pen Settings

Within the panel “Bitmap” some scanner-bitmap-specific parameters can be set. They are used only, when this pen is assigned to a bitmap but has no effect when it is used with vector data.

Please also note the button “Set Defaults for Bitmap-Marking“ which is described in section “6.8.2 Scanner Pen Settings” above.

This panel provides the following parameters:

- Bitmap greyscale mapping – in some cases a linear 0..100% power range does not result in similar, linear grey tones. This happens when the laser does not react linear on provided power values and/or when the material does show a non-linear behaviour on marking. In such cases this option can be enabled to define an own greyscale mapping which allows to adjust such deviations.
  On upper part of the window a nominal (linear) greyscale range is shown. Below of these the mapping values can be entered which have to be assigned to the nominal, input grey values of the bitmap to be marked. Below of these sliders and input fields the resulting greyscale map is shown as it would be sent to the laser in terms of power values. When the given values are correct, this non-linear, lower greymap then would result in linear marking results.
  The given values are in range 0..255 which corresponds to the 8 bit greyscale values of a scanner bitmap.
  This greyscale map is used only with greyscale bitmaps where a pen is assigned to in which this option is enabled.
  To adjust a bitmaps greyscales with this map, the test image from https://halaser.eu/scannerbitmap_test.png can be used – beside different patters which can be used to evaluate the quality of a greyscale marking it contains seven grey values in lower right corner which directly correspond to the seven grey parameters which can be set with this option. So as soon as these seven fields have the correct brightness after marking, the given greyscale map is correct.

6.8.4 Laser Pen Parameters

This panel contains several parameters that belong to the used laser and the parameters/signals that are required to operate it. So depending on the laser type that was configured for the current project these options are enabled that contain useful parameters for proper laser operation. For a detailed description of these parameters and their meaning please refer to your laser vendors manual.

PLEASE NOTE: some laser types need to be operated using relatively tight parameter ranges. When they are controlled with wrong parameters the results may be an undefined behaviour or the laser hardware may be damaged in worst case. So when setting up new parameters or when changing them or after a software update was done, it is highly recommended to (re-)check these laser parameters and the output signals of the scanner controller in order to avoid problems with the laser. Here “software update” not only includes updates of the OpenAPC package or BeamConstruct. Also in case the software or drivers that for an example belong to the scanner controller card are modified, it is recommended to perform these checks.

6.8.5 Using Pens for Coloured Bitmap Marking

On some materials variations of power, speed and frequency cause marking results in different colours. This behaviour can be used within BeamConstruct to easily mark coloured bitmaps. Following the steps are described to get such images. First of all it is important to know that the process of marking colours is not easy to handle. Environmental conditions may have a strong influence the result so that pen parameters that cause a specific colour one day may result in a different colour the next day. So the whole process should be done in a well-defined environment. Next the material condition can have an influence too, its state of oxidation, grease or dirt on the surface may affect the resulting colour. So in case of troubles a defined

cleaning process may be necessary, possibly by using a laser- or plasma cleaning process.

To get coloured bitmaps following steps are necessary:

- check image and the colours it contains
- find as much as possible pen parameters that result in different colours when they are used for marking on your target material; they should include as much of the colours that are contained in image to be marked
- calibrate your monitor so that the colours displayed there are as close as possible to the real colours
- walk through all pens where you found parameters for coloured marking results, enable option “Use for coloured scanner-bitmaps” and click the colour chooser “Colour after marking”, there select a colour that is as close as possible to the marking result of this pen (that's the step where the calibrated monitor is required)
- close pen dialogue
- import your coloured image using menu Project → Import
- select option “Use as coloured scanner bitmap” in image import dialogue; when this option is not enabled, you don't have enough pens defined for usage in coloured bitmaps
- on top left corner of import dialogue a preview of the image can be seen that makes use of the colours defined by the pens; now the image can be manipulated using the sliders for Brightness, Gamma, Contrast, Correction Red, Correction Green and Correction Blue to improve the result

Now after clicking OK the image is imported. It is processed in a way that maps the existing coloured pens to closest colours in image. During marking, such an image is processed in several cycles, one for every colour/pen that is contained.

6.9 Construction Of Geometry

Comparing to other CAD applications that offer basic primitives with huge amount of options and parameters the concept of BeamConstruct is different: It offers several basic primitives like lines, circles and others that contain only as less parameters as necessary. To get a wide range of effects and complex geometries, these basic data can be combined with other elements that add new data to them or that modify and manipulate them. Thus several simple and easy to understand elements can be put together to complex geometries.

BeamConstruct offers following element types that can be used for these combinations:

- Primary Geometry – such an element is always the starting point and offers a basic, primitive geometric element (like a rectangle, triangle, text or others like this); this element can be combined with several other elements like:
- Additional Geometry – this elements uses a previously defined geometry and adds some more geometry to it
- Post-processing Tool – this element modifies existing geometry
- Input Element – here an indirect influence to existing geometries is possible, this kind of element gets data from an external source and puts it into Primary Geometry elements that can use these data for their own geometry (like the text or barcode element)

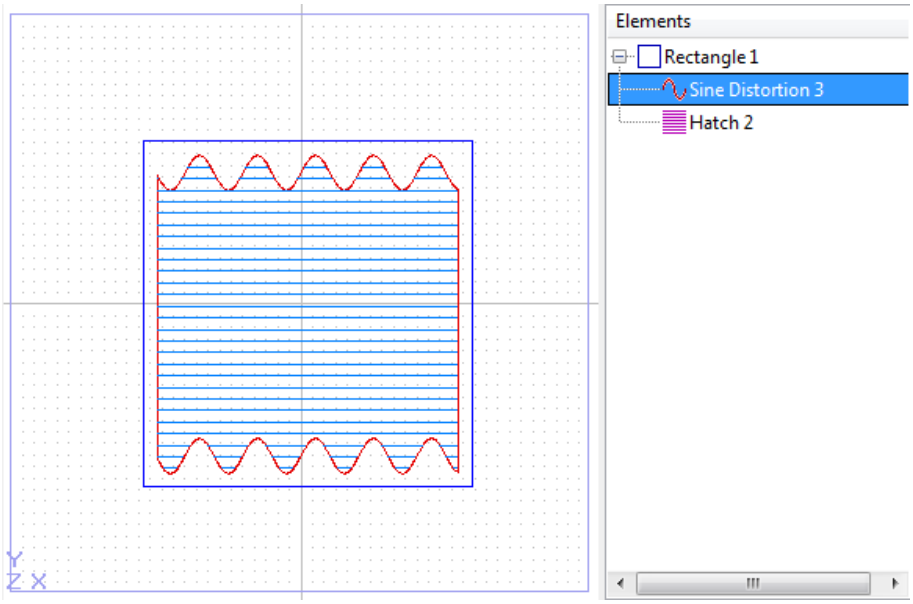
Following rules apply to these elements:

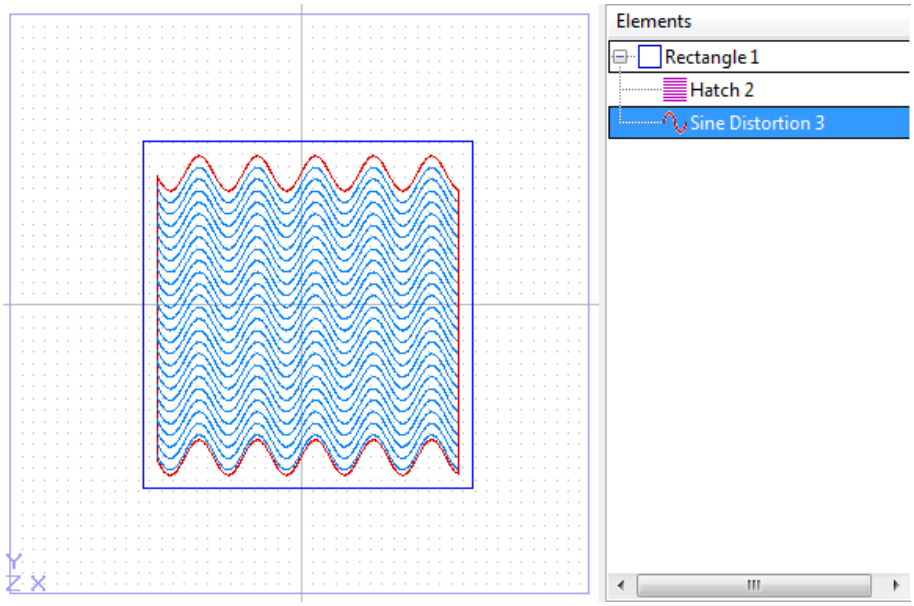
- a Primary Geometry element can have several child elements but none of type “Primary Geometry”
- Additional Geometry and Post-Processing Tools can exist several times as child of a Primary Geometry element, but they influence only the state of the geometry like it exists before their own position
- an Input Element can exist exactly once for a Primary Geometry element and only for such elements that really are able to handle input data)

The second rule can be made explained better with an example: as Primary Geometry a rectangle is drawn.

It is modified with a “Sine Distortion” Post-processing Tool which changes the borders of the rectangle to have a sine waveform. Next an additional element “Hatch” is used which adds lines to the geometry that fill the rectangle

In this case the sine distortion is applied to the rectangle only (because it is located before the sine-element) but not to the hatching lines (because they are added after the sine distortion takes place. So the hatching lines stay straight:



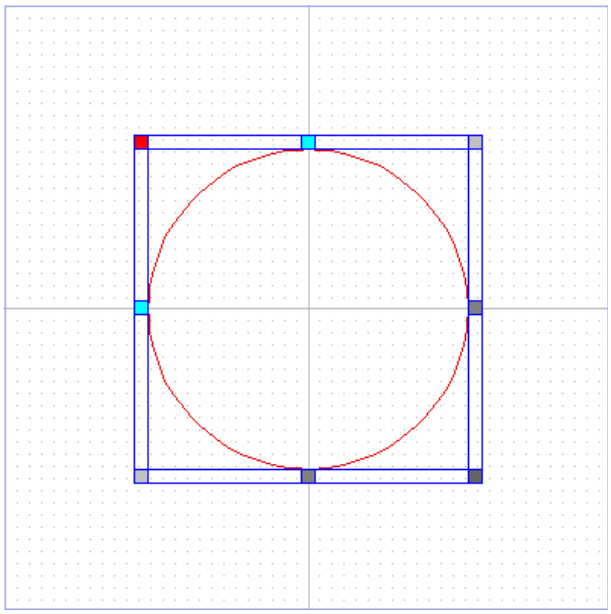


The situation becomes different when the two sub-elements change their position. When the first child element below of the primary element is the hatcher the resulting hatch lines will be distorted by the following Post-processing Element too. They are no longer straight lines but have the same sine shape like the rectangles border because the sine distortion now applies to the hatcher too: that's caused by the fact that the sine distortion Post-Processing Tool stays after the hatcher:

PLEASE NOTE: for the example explained above the line styles for rectangle and hatch have to be switched to “Connected Lines”.

6.9.1 The Drawing Area

Within BeamConstruct's main window the drawing area for geometry is located in the middle. There new primary elements can be created or existing geometry can be modified. Whenever an element is selected, it is highlighted by a double-lined, blue border. This border can be used to manipulate the element. When pressing the left mouse button within this border and outside the coloured squares it contains, it can be moved and repositioned. Left-clicking it within one of the squares gives access to other manipulation functions:



- the dark-grey rectangle at the lower right corner provides a resizing-function where the aspect ratio between both visible directions is kept intact
- the medium grey rectangles at the lower or right border of the rectangle provide resizing functionality in one direction only
- the light-grey rectangles in upper right and lower left corner allow to rescale the element in every direction and without keeping the current aspect ration between the two visible directions
- the cyan rectangles at the left and upper border can be used to slant the geometry in one of the visible directions
- the red rectangle in upper left corner can be used to rotate it

How and by which value the elements are changed can be seen within the panels on the main windows left hand side as described in following section.

PLEASE NOTE: The manipulation possibilities described above apply to the geometry that is currently selected. When a primary element is selected the changes apply to its geometry, possibly existing additional or Post-processing Tools are re-calculated according to these changes. Whenever additional geometry within a Primary Element is selected and manipulated, this has no feedback to its parent Primary Element, in this case the Additional Geometry might be changed without any relation to the parents geometry and therefore in a way that is not desired.

The part of the current project that is visible within the drawing area can be changed too: when no element is selected the whole area can be moved by holding down the left mouse button and dragging it into the direction where the complete view has to be moved to. Beside of that there are zoom-buttons available within the tool-bar:

- Zoom In – zooms into the drawing area so that more details become visible
- Zoom Out – show a bigger area but less geometry details
- Zoom To Working Area – changes zoom and position of the current view so that the full working area is visible in the maximum size that is available in the drawing area
- Zoom To Selected – changes zoom and position of the current view so that the currently selected element is visible in the maximum size that is available in the drawing area

When one single element is selected that contains some geometry, a special point editing mode can be enabled to modify the single points this element consists of. This mode can be enabled by turning on the tool-bar button right beside the zoom-buttons. Once this button is enabled the appearance of the current element changes, now all editable points are surrounded by a rectangle. These points now can be dragged to other positions. This can be done for every point separately. To drag a group of points together please refer the description of the Geometry Tab Panes table “Coordinates” below.

6.9.1.1 Select Elements

To select existing elements for further editing there exist different possibilities:

- left-click the geometry of a single element in drawing view → the element is selected, when it is part of a group, the whole group is selected
- left-click the name of a single element in element tree → the element is selected
- left-click the geometry of several elements in drawing view while Shift- or Ctrl-key is held down → the elements are selected, when one of them it is part of a group, the whole group is selected
- left-click the name of several elements in element tree while the Ctrl-key is held down → the clicked elements are selected, clicking an element again un-selects it
- left-click the name of two elements in element tree while the Shift-key is held down → all elements which are in range from first to the second clicked element are selected
- draw a frame around several elements in drawing view while the Alt-key is held down → all elements which are touched by the drawn frame are selected, when one of them is part of a group, the single element is selected but not the whole group

6.9.1.2 2D and 3D Editing

Within most scanner controller parameters (please refer to the description of the project settings above) it is possible to choose between a 2D operation mode (two-axis mode) and a 3D operation mode (three axes controlled by the scanner controller). This option depends on the capabilities of the used scanner controller and influences the behaviour of BeamConstruct's drawing area. When a two-axis mode is selected, only the two axes X and Y can be modified within BeamConstruct and the drawing area offers manipulation possibilities for X (horizontal direction) and Y (vertical direction) only.

As soon as 3D mode is enabled, the drawing area is extended by some semi-transparent buttons that are shown directly within the drawing areas upper left corner. These buttons can be used to switch the current view and the axes that are visible within this view:

- top view – the default view for a laser marking system from top where the laser source is located above the working piece, here X/Y are shown in horizontal/vertical direction of the drawing area
- front view – view from front where X/Z are shown in horizontal/vertical direction of the drawing area
- right view – view from right side where Y/Z are shown in horizontal/vertical direction of the drawing area
- 3D view – here all three dimensions of the current project are visible at the same time within a three-dimensional, foreshortened view; comparing to the other views here no elements can be created and edited but it is possible to rotate the view by moving the mouse while holding down the right mouse button
- split view – when this view is enabled, all four views described above are shown at the same time within a split view. This mode requires more hardware resources but makes editing easier because all three views (top, front, right) can be edited simultaneously.

6.9.2 The Property Tab-Panes

The main window of BeamConstruct contains some special tab-panes on its left hand side. These panes give access to different parameters and values, they can be used to influence geometries in a very exact

way.

6.9.2.1 The tab-pane Of The Currently Selected Element

The very left tab-pane is a dynamic one, it changes its contents depending on the element that is currently selected. As long as no element is selected this panel is empty, no parameters can be changed here. These element-specific values that become visible as soon as something is selected within the drawing area or within the element tree on the windows right hand side are described in detail below together with the different elements.

In case more than one element or in case a simple group element is selected this tab-pane is empty too.

6.9.2.2 The Geometry Tab-Pane

This panel can be used only when an element is selected that contains some geometry. Thus it does not offer any functionality as long as no element or as long as a geometry-less Post-Processing Tools is selected.

Within this panel geometry-related values are displayed and can be changed. When 3D mode is enabled, the number of available and usable options and parameters depends on the currently selected view, in 2D mode only these values and options can be changed and used that belong to a flat, two-dimensional project that only extend in X and Y direction:

- Mirror X, Mirror Y and Mirror Z buttons – when pressing one of these buttons the currently selected geometry is mirrored in X, Y or Z direction; pressing the same button twice results in no changes (element is mirrored and then mirrored back to original state in this case)
- Centre X, Centre Y and Centre Z buttons – these buttons reposition the selected geometry within the current working area, they centre it in X, Y or Z direction
- Scale – these input fields show the current scaling factor comparing to the original state of the geometry for X, Y and Z direction, a value of 1.0 means the geometry is not scaled, values smaller than 1.0 mean it is shrunk in this direction, vales greater than 1.0 mean it is bigger comparing to the original size
- Top Left – the upper left coordinates of the selected element, whenever they are changed the element is repositioned without changing its size
- Centre – the centre coordinates of the selected element, whenever they are changed the element is repositioned without changing its size
- Bottom Right – the lower right coordinates of the selected element, whenever they are changed the element is repositioned without changing its size
- Size – the total width, height and depth of the element, when it is changed the geometry is scaled in X-, Y- or Z-direction, the related values for scaling
- Rotation – the angle in unit degrees the element is rotated with, here the rotation can be changed which may result in changes of other parameters depending on the shape of the used geometry
- Coordinates – this list contains the coordinates of all points of the currently selected geometry, here they can be changed to re-position a single point of it; selecting a point within this list also affects the highlighted element within the drawing area: there the selected point is surrounded by a separate, blue box which can be used to drag this point, thus the geometry can be edited using the mouse. Within this table of coordinates a single point or groups of points can be selected. This can be done by left-clicking into the row of the point that has to be selected. Additional points can be chosen by selecting additional rows of points while the shift or control key is held down. In case more elements are selected they can be modified within the drawing area by dragging them to other positions, here all highlighted points are moved together.
  Comparing to this a point editing mode is available that gives the possibility to highlight and edit all points of an element separately. For more details about this mode please refer above.

For the parameters described here the same is true as described for the manipulation functionalities within the drawing panel: When the changes are applied to a primary element, they influence the subsequent

elements too. When an additional element is selected there is no feedback to its parent primary element, the changes may override the parents geometry.

6.9.2.3 The Element Tab-Pane

Comparing to the Geometry-panel this one contains more global parameters of the selected element. Depending on the type of selected element only some of the parameters are accessible:

- Type – this is an informational field and therefore read-only, it displays the type of the element
- Name – here the name of the element is shown and can be changed; this name is auto-generated when the element is created and shown within this panel and within the element-tree on the main windows right hand side; for some operations out of e.g. BeamServer or own applications it may be necessary to have unique names here.
- Layer – BeamConstruct supports several drawing and displaying layers that can be used to separate parts of the whole project from each other and to make it easier to manage complex projects. These layers can be hidden and shown depending on which has to be visible. Using the combo-box provided here the selected element can be assigned to one of the available layers. For more information please refer the section about the layer tab-pane below.
- Pen – a pen is a definition about how the related geometry has to be processed on the target system; here a pen can be assigned to the current geometry so that is is processed using these specific pen parameters; please refer above for a description of the pen parameters
- Process Geometry – this check-box decides if the related geometry has to be processed or not, when it is not checked the related element acts as some kind of help-drawing only, it is not used by the target system for processing the working piece
- Trace data – when this option is set, the name and when available unique data of this element are used for traceability purposes. This means, when a marking cycle was performed successfully, the related Smart Interface message contains information about this element too. These information consist of the name of the element. When this option is enabled for an input element, beside the name also the possibly unique input data are added to the trace data. For more information please refer to section “6.21 Smart Interface“
- Count X/Y/Z and Distance X/Y/Z – these four parameter fields belong to each other and can be used to create arrays of geometry without the need to duplicate them; here it can be defined how often they have to be repeated in X, Y and Z direction and which distance has to be set between them
- Number of vectors – this is an informational element and can't be changed directly, it displays the number of vector lines that are used to create the selected element
- Repeat marks – using this parameter it can be specified how often marking of a single element has to be repeated during a single marking cycle
- Skip marks – when this is set to a value greater than 0, the related element is no longer marked on every process cycle but skipped for the given number of times; after this number has elapsed, it is marked once and skipped afterwards again
- Skip offset – this parameter belongs to "Skip marks" and specifies an offset at what mark/skip counting has to start; so with both parameters elements can be created that are marked interleaved:

  Element 1 with skip marks = 1 and skip offset = 0
  Element 2 with skip marks = 1 and skip offset = 1
  → results in two elements that are marked alternating, each exclusively

  Element 1 with skip marks = 2 and skip offset = 0
  Element 2 with skip marks = 2 and skip offset = 1
  Element 3 with skip marks = 2 and skip offset = 2
  → results in three elements that are marked alternating, each exclusively
- Total line length – this field contains an information too, here the total length of the geometry is shown; this value takes the “on-”-lines into account only, “off”-lines (movements with the tool turned of to go to the beginning of the next “on”-line) are ignored here
- Process time – here the time is displayed that is needed to process the currently selected element;

this value is calculated out of the length of “on”-lines, the length of “off”-lines between them and the speed and delay information of the pen that is assigned to this element

6.9.2.4 The Layer Tab-Pane

Every element can be assigned to exactly one layer. These layers are in no way related to the processing of the data within the target system, but they are a useful utility that makes it easier to handle complex projects.

Within this tab panel two things can be done: layers can be renamed according to their meaning and they can be turned on and off.

When the box in front of a layer is checked, all elements that are assigned to this layer are drawn within the drawing area. Layers that are not checked are not visible within the drawing area, they are hidden and therefore can't be selected or edited.

When a marking process is started out of BeamConstruct the layer visibility states do not have an influence on the marking result. Means also these layers that are currently hidden will be marked.

6.9.2.5 The Motion Tab-Pane

In tab panel Motion it is possible to drive configured motion axes manually and independent from any motion elements in current marking project. Please note: when there are some marking elements used directly within the project, it has to be taken into account that the user may have changed the position of the axes via these manual functions. Thus at least the first motion element of a project should perform a movement to a defined, absolute position or should reference its position.

Within the motion tab-pane following functions are available:

- OUT2 / OUT3 – specifies which of the two possible motion controllers has to be used
- Speed – specifies the speed a manual motion operation can be performed with;
  PLEASE NOTE: when the speed value given here is bigger than the maximum speed of the related axis, this value is clipped internally, means the axis moves with it's maximum speed which may be slower than the nominal speed entered here
- movement value – the value that can be entered in middle between the arrows specifies the relative movement distance or the absolute motion position to move to (depending on the arrow button that was pressed)
- Double arrows – these buttons cause a movement as long as the button is pressed
- Single arrows – these buttons cause a single movement via the distance or angle specified with the number input field in the middle, here the time the button is pressed does not influence the motion
- Arrow to dot – when this button is pressed an absolute movement is performed, the number input field in the middle between the buttons specifies the target position to move to, the time the button is pressed does not influence the motion
- Ref – start referencing of the related axis

For planar movement axes the images of the buttons can be changed within the motion axis configuration panel of the general project settings. Depending on the real movement direction (left/right, up/down, in/out) a different symbol can be shown here in order to make it easier for the operator to decide which direction a motion will use. The same is true for the axis names, they can be configured via the axis alias value in general project settings.

6.9.3 The Element-Tree

On the main windows right hand side a tree is shown that represents the current projects structure. It consists of a list of primary elements and their (optional) child elements. During processing on the target system the elements are processed in the same order as they are listed there from top to bottom (including their children).

Every primary element may contain several Additional Geometry elements and several Post-Processing Tools. They are listed as child nodes of their Primary Geometry parents within this tree.

All these primary and child nodes can be edited within this tree via a context menu. This menu can be opened by selecting a node within the tree and clicking the right mouse button. Then following functions can be chosen:

- Group Elements – while holding down the Shift- or Ctrl-key it is possible to select a range of elements within the entity tree; in case more than one Primary Geometry element is selected it is possible to put them into a group using this function
- UnGroup Elements – this item can be used when a group element is selected, it resolves the group, pulls all element out of it into the hierarchical level where the group existed before
- Delete Element(s) – deletes an element completely; in case this is applied to a primary element all its sub-elements are removed too
- Duplicate – the currently selected Primary Geometry element is copied and inserted at a position that is slightly shifted according to the original element
- Move To Top – the selected element is moved to the top position within its same hierarchically level (means a child of a primary element will stay a child of it but it will become the first one)
- Move Up – the selected element is moved one position up but also within the same hierarchically level
- Move Down – the selected element is moved one position down within the same hierarchically level
- Move To Bottom – the selected element is moved to the bottom position within its same hierarchically level (means a child of a primary element will stay a child of it but it will become the last one)
- Import 3D Mesh – this option is active only when a single Sliced 3D Mesh Group element is selected, it gives the possibility to import an additional 3D Mesh into this Sliced 3D Mesh Group; after successful import this Sliced 3D Mesh Group keeps two more 3D Meshes and creates slices that consist of all of their 3D data, the 3D Meshes which can be positioned within the working area independent from each other
- Auto-Arrange – this option is active only when a single Sliced 3D Mesh Group element, when a group or when several primary elements are selected, it gives the possibility to arrange several 3D Meshes or child elements within a group; for a detailed description of the auto-arrange function please refer below
- Merge 3D Meshes – when several 3D slice groups are selected, this option can be chosen to merge their 3D meshes into one element. After this operation has been finished, all the selected 3D slice groups are combined into one containing all the 3D meshes as one single object

6.9.3.1 Auto-Arrangement of elements in groups

For a description of auto-arrange function of 3D Meshes in Sliced 3D Mesh Groups please refer to the related section below.

A group can hold several sub-elements as children. These elements can be positioned with the standard functions in tab-pane "Geometry" in order to place them at suitable positions in working area. Alternatively it is possible to let BeamConstruct arrange these elements automatically. To do this, the group has to be selected and right-clicked in element tree, then menu item "Auto-Arrange" has to be selected. Now a dialogue opens which provides options to arrange the elements in X- and Y-direction and to align them at a specific Z-height. This is done via the configuration possibilities within the dialogue:

- Arrange in X and Y – when enabled, the sub-elements are arranged on working area using the following parameters
- Distance to borders – specifies the minimum distance the child-elements of the group need to have to the border in both, X and Y direction of the working area
- Distance between elements – specifies the minimum distance the children of the group need to have between each other

- Arrange in Z – when enabled, the elements are aligned at a specific height using the following options and parameters
- Align geometries at top or bottom – here it can be selected if all the elements need to be aligned at their upper or lower position
- Arrange in Z – this is an option which can be enabled when elements have to be top or bottom aligned at a specific, absolute height; when it is set, the height has to be specified in unit mm; in case this option is not set but arrangement in Z is enabled, all sub-elements of the group are aligned at the highest/lowest top/bottom position of that element, that is located at the highest/lowest position (according to its outlines top/bottom border)

6.9.4 Primary Geometry Elements

To create a new primary element following steps have to be performed:

1. select the related button within the tool-bar, all primary elements use blue tool-bar symbols
2. click into the working area to position the element; depending on the geometry that is created by the related function it can be drawn now with some additional clicks into this area
3. now the new element is selected and its parameters are shown within the first, element-specific tab-pane on the main windows left hand side

When the parameters of such an element are changed within its tab-pane, all following elements of the same type use exactly these parameters as default values.

Following the available elements, the steps to create them within the working area and their parameters are described.

6.9.4.1 Dot

This is the most simple element that is possible. When it is selected for drawing within the tool-bar, the first click into the drawing area creates a single point at the left-clicked position.

This element does not offer any parameters, so there are no input fields available within the element-specific tab panel as soon as such a dot is selected.

Within the options the dot can be configured:

- Style – specifies the style of the created point, here Dot is a simple one that consists of a single coordinate and has no size or Custom for a dot as configured within the global parameters (please refer to section 6.8.1 above)

6.9.4.2 Line

When drawing a line two left-clicks are necessary within the drawing area: the first click specifies the starting position of the line, the second one the end position. It is not necessary to keep the mouse pressed between these two clicks.

Drawing functionality can be extended by pressing one of the keys "Shift" or "Ctrl" after specifying the starting point of the line (first click): when Shift is held down, a horizontal line is drawn, and vertical deviations of current mouse position are ignored. When Control is held down, a vertical line is drawn, and horizontal deviations of current mouse position are ignored.

After drawing or selecting such a line following parameters are available to manipulate them:

- Line Style – the style the line is created with, here following options exist:
  Continuous: the geometry consists of exactly one starting and one end point where one single line is drawn between
  Connected Lines: the geometry consists of several short lines including additional points between the start and end point, all these lines are connected
  Dashed: the geometry consists of several short lines including additional points between the start

and end point, every second line is drawn that results in a dashed line style
Dotted: no lines are drawn, the geometry consists of a set of not connected dots
Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)

- Distance – whenever a non-continuous line style is selected, this distance value specifies the length of the additional points that form the selected line style
- Expand By Dot – expands the line according to the chosen distance-value by one dot (or one line segment)
- Remove Last Dot – removes the last dot or line segment

6.9.4.3 Triangle

To draw a triangle all three corners have to be specified separately by single left mouse clicks for each. First the starting point is defined, with the next click an other corner of the triangle is specified and the last one completes the triangle by setting the third corner of it.

Drawing functionality can be extended by pressing one of the keys "Shift" or "Ctrl" after specifying the first corner point of the triangle (first click): when Shift is held down, a horizontal line is drawn next, and vertical deviations of current mouse position are ignored. When Control is held down, a vertical line is drawn, and horizontal deviations of current mouse position are ignored.

Within the triangles property tab-pane following options are available:

- Line Style – the style the triangles lines are created with, here following options exist:
  Continuous: the geometry consists of exactly tree points that are connected by three completed lines
  Connected Lines: the geometry consists of several short lines connecting the three corners including additional points, all these lines are connected and therefore form a closed shape
  Dashed: the geometry consists of several short lines including additional points between the three corners, every second line is drawn that results in a dashed line style
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)
- Distance – whenever a non-continuous line style is selected this distance value specifies the length of the additional points that form the selected line style

6.9.4.4 Rectangle

When drawing a rectangle the upper left and the lower right corner of it are specified by two separate mouse clicks. The resulting rectangle will always be parallel to the coordinate axes.

Drawing functionality can be extended by pressing the "Shift"-key after specifying the first corner point of the rectangle (first click): when this key is held down, a square is created instead of a rectangle. Horizontal distance from first point specifies the side length of the square.

Within the rectangles property tab-pane following options are available:

- Line Style – the style the rectangles lines are created with, here following options exist:
  Continuous: the geometry consists of exactly four points that are connected by four completed lines which form a rectangle
  Connected Lines: the geometry consists of several short lines connecting the four corners including additional points, all these lines are connected and therefore form a closed, rectangular shape
  Dashed: the geometry consists of several short lines including additional points between the four corners, every second line is drawn so that it results in a dashed line style
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)
- Distance – whenever a non-continuous line style is selected this distance value specifies the length of the additional points that form the selected line style

- Nominal width – this is the width of the rectangle (in X-direction) as done by this primary element; so this value does NOT reflect possible scale factors done in “Geometry”-tab-pane, the real geometry width is shown only there
- Nominal height – this is the height of the rectangle (in Y-direction) as done by this primary element; this value does NOT reflect possible scale factors done in “Geometry”-tab-pane, the real geometry height (after all transformation operations have been applied) is shown only there
- Nominal depth – this is the depth of the rectangle (in Z-direction) as done by this primary element; this value does NOT reflect possible scale factors done in “Geometry”-tab-pane, the real geometry depth (after all transformation operations have been applied) is shown only there
- Corner radius – when this value is greater than 0 and when the rectangle is large enough, it is created with rounded corners
- Segments – when the rectangle is created with rounded corners, this value specifies of how much segments all these corners consist and therefore how smooth they are

6.9.4.5 Circle

This primary element can be used to create circles and arcs. Starting point is always a circle that is drawn in two steps: the first left mouse click into the drawing area specifies the centre point of it, the second one specifies the radius (via its distance from the first point).

Next the circle can be modified further:

- Line Style – the style the circle is created with, here following options exist:
  Continuous: the geometry consists of several short, fully connected lines that form a closed circle; for this mode the parameter “Segments” specifies how much of these short lines are used to create this circle, the more lines are used the more smooth the circle becomes
  Connected Lines: the geometry consists of several short lines that form the circle, comparing to the previous option here the parameter “Distance” influences the length of these lines and therefore the resolution; here the shorter the lines are the more segments this circle consists of and the more smooth it becomes
  Dashed (Arcs): the circle consists of dashes where the dashes are not straight lines but follow the shape of the circle and therefore are round
  Dashed (Straights): the circle consists of dashes where the dashes do not follow the shape of the circle but are straight lines
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)
- Distance – this parameter decides how long the resulting lines of the chosen line style have to be, it is not valid for the continuous line style where the length would be always equal to the circumference
- Radius – the radius of the circle
- Segments – here it can be chosen how much segments the circle has to consist of, this parameter influences the accuracy of the resulting circle; for the line style “Dashed (Arcs)” it influences the smoothness of the resulting arcs while the “Distance”-parameter influences their length
- Start Angle and End Angle – these two parameters can be used to change the circles starting and end angle, it can be used to draw arcs (in case the resulting total angle is smaller than 360 degrees) or to let a circle be drawn several times (in case the resulting total angle is a multiple of 360 degrees)
- Grow: this parameter is valid in 3D mode only; it defines how much the circle shall grow in the third direction per every rotation of it. Here “third direction” does not necessarily mean Z-direction, this depends on the view the circle is drawn in. So in case the right side view is enabled the circle extends in Y and Z direction, resulting from that the third direction where the grow parameter applies to is the X direction.
- Layered Sphere: in 3D mode this option can be used to create a sphere that consists of circles of different diameters that form a sphere; to use this option the Grow value has to be set which specifies the distance between the circles that form the sphere
- Draw CCW – normally the circle has a clockwise orientation, means it is drawn to right direction;

when this check box is set, the drawing direction is reversed

6.9.4.6 Star

This element is a special variant of a circle, it consists of two radii where every second point the element consists of, belongs to an other radius. Resulting from that stars can be drawn and modified by its parameters. Such a star is drawn in two steps: the first left mouse click into the drawing area specifies the centre point of it, the second one specifies the first radius. While drawing the star the second radius is set automatically to value smaller than the first radius.

After the star is drawn it still can be modified interactively. Here every second point (the ones that belong to the second radius) is marked by a small, blue cross. This cross can be dragged with the mouse in order to change the second radius independent from the first one and in order to apply an offset to the angle of these points.

Next the star can be modified via its options:

- Line Style – the style the circle is created with, here following options exist:
  Continuous: the geometry consists of single, continuous lines
  Connected Lines: the geometry consists of several short lines including additional points, all these lines are connected
  Dashed: the geometry consists of several short lines including additional points, every second line is drawn that results in a dashed line style
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)
- Distance – this parameter decides how long the resulting lines of the chosen line style have to be, it is not valid for the continuous line style
- Radius 1 – the first radius that is valid for all odd-numbered points
- Radius 2 – the second radius that is valid for all even-numbered points
- Spikes – here it can be chosen how much spikes (or rays) the star shall have,
- Angle offset – here an offset can be specified for the points that are assigned to the second radius in order to create non-regular spikes
- Draw CCW – normally the star has a clockwise orientation, means it is drawn to right direction; when this check box is set, the drawing direction is reversed

6.9.4.7 Spiral

This primary element can be used to create spirals. Starting point is always a spiral that is drawn in two steps: the first left mouse click into the drawing area specifies the centre point of it, the second one specifies the outer radius.

Next the spiral can be modified via its options:

- Line Style – the style the spiral is created with, here following options exist:
  Continuous: the geometry consists of several short, fully connected lines that form a continuous spiral; for this mode the parameter “Segments” specifies how much of these short lines are used to create this spiral, the more lines are used the more smooth the resulting spiral becomes
  Connected Lines: the geometry consists of several short lines that form the spiral, comparing to the previous option here the parameter “Distance” influences the length of these lines and therefore the resolution; here the shorter the lines are the more segments this spiral consists of and the more smooth it becomes
  Dashed (Arcs): the spiral consists of dashes where the dashes are not straight lines but follow the shape of the spiral and therefore are round
  Dashed (Straights): the spiral consists of dashes where the dashes do not follow the shape of the spiral but are straight lines
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots

Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)

- Distance – this parameter decides how long the resulting lines of the chosen line style have to be, it is not valid for the continuous line style where the length would be always equal to the circumference
- Inner Radius – the inner, minimum radius of the spiral
- Outer Radius – the outer, maximum radius of the spiral
- Slope – this value defines the steepness of the spiral, so this value influences how strong it rises per rotation
- Segments – here it can be chosen how much segments the outer part of the spiral has to consist of, this parameter influences the accuracy of the result; for the line style “Dashed (Arcs)” it influences the smoothness of the resulting arcs while the “Distance”-parameter influences their length
- Smoothing Factor – normally a spiral is generated out of short lines where the length is calculated out of the number of segments and the current radius. This leads to an adaptive line length with as less as much data and an as fast as possible, resulting process time. But in some cases this may lead to spirals that are not very smooth but angular at their inner radius. Here the number of segments could be increased (what would lead to a higher amount of data used because of smoother corners also at curves where it would not be possible) or the smoothing factor can be set to values greater 1.0: this changes the calculation of a spiral in a way where the inner line length becomes shorter and therefore more smooth while not using more data than necessary. So the more smoothing factor is taken for the calculation the smaller the radius becomes.
- Start Angle – this parameters can be used to change the spirals starting angle
- Grow: this parameter is valid in 3D mode only; it defines how much the spiral shall grow in the third direction per every segment. Here “third direction” does not necessarily mean Z-direction, this depends on the view the spiral is drawn in. So in case the front side view is enabled the spiral extends in X and Z direction, resulting from that the third direction where the grow parameter applies to is the Y direction.
- Draw CCW – normally a spiral has a clockwise orientation, means it is drawn from its starting point to right direction; when this check box is set, the drawing direction is reversed
- Inside To Outside – this option influences the growing direction, when the check box is set, drawing of the spiral starts at the inner side and ends outside; in case it is not set it starts at the outer side and ends at the inner

6.9.4.8 Polygon

A polygon consists of an undefined number of edges (but requires at least three). Drawing it can be done by left clicking at the position where a corner between the connecting lines has to be positioned. This can be done several times, the process of drawing this polygon can be finished by right-clicking the mouse. Then the last corner and the first position where the user left-clicked are connected by a line.

Drawing functionality can be extended by pressing one of the keys "Shift" or "Ctrl" after specifying the starting point of the polygon (first click): when Shift is held down for one of the following points, a horizontal line is drawn next, and vertical deviations of current mouse position are ignored. When Control is held down, a vertical line is drawn, and horizontal deviations of current mouse position are ignored.

Following options are available to modify the parameters of the polygon:

- Line Style – the style the polygons lines is created with, here following options exist:
  Continuous: the geometry consists of lines that use exactly one starting and one end point where one single line is drawn between
  Connected Lines: the geometry consists of several short lines including additional points, all these lines are connected
  Dashed: the geometry consists of several short lines including additional points, every second line is drawn that results in a dashed line style
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)

- Distance – whenever a non-continuous line style is selected, this distance value specifies the length of the additional points that form the selected line style
- Polygon Style – here it can be chosen if the last polygon point and the first one are connected by a line (style “Closed”) or not (style “Open”)

6.9.4.9 Bezier Curve

Creation and parameters of this element are very similar to the polygon described above but it does not generate direct connections between the specified base points. Using this element a smooth, round Bezier curve is generated. This curve uses the given base points as creation hint but does not necessarily cross them.

Such a Bezier curve consists of an undefined number of base points but requires at least three of them. Specifying the position of these base points can be done by left clicking within the drawing area. This can be done several times, the process of drawing the related curve has to be finished by right-clicking the mouse.

After that the positions of the base points still can be modified interactively. As long as this element is selected, the real position of the base points are marked by blue crosses. These crosses can be dragged with the mouse. Changing the position of the base points also changes the appearance of the resulting Bezier curve.

Following parameters are available to modify the parameters of the Bezier curve:

- Line Style – the style the polygons lines is created with, here following options exist:
  Continuous: the geometry consists of continuous lines that consist of a number of sub-segments which are specified by the parameter “Resolution”
  Connected Lines: the geometry consists of several short lines of a defined length including additional points, all these lines are connected
  Dashed: the geometry consists of several short lines of a defined length including additional points, every second line is drawn that results in a dashed line style
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)
- Distance – whenever a non-continuous line style is selected, this distance value specifies the length of the additional points that form the selected line style
- Bezier Style – here it can be chosen if the last polygon point and the first one are connected by a line (style “Closed”) or not (style “Open”)
- Resolution: this parameter specifies the smoothness of the Bezier line, the bigger it is the more sub-lines are used to create such a curve, the more smooth it becomes and the more resources are used for it

6.9.4.10 Text

This primary geometry can be used to draw some text using an existing TrueType or special laser font. This element accepts static texts or dynamic values that can be set via an Input Element. When such a project is moved from one computer to an other or from one operating system to an other it has to be made sure a used font is available on the other system too (this applies to TrueType fonts only since the fast laser fonts are part of BeamConstruct). Otherwise the application tries to find a similar font which might be only somehow similar, but exact positions, sizes and outlines may be got lost in such a case.

Drawing such a text is quite easy: only one single click into the drawing area is necessary to specify the position of the text, there a new geometry is set using a default text. This text can be changed together with some other parameters:

- Line Style – the style the lines the text consists of is created with, here following options exist:
  Continuous: the geometry consists of closed, connected lines with as less as possible points
  Connected Lines: the geometry consists of several short lines including additional points, all these lines are connected
  Dashed: the geometry consists of several short lines including additional points, every second line is

drawn that results in a dashed line style
Dotted: no lines are drawn, the geometry consists of a set of not connected dots
Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)

- Distance – whenever a non-continuous line style is selected, this distance value specifies the length of the additional points that form the selected line style
- Text – here a multi-line text can be specified that has to be drawn; a modified text is applied as soon as the text input area is left, either by clicking an other element or by pressing the Tab-key
- Unicode – instead of characters here space-separated Unicode values can be entered either as decimal numbers or as hexadecimal numbers with a leading “0x”. On pressing “Return” the entered numbers are converted to text and shown as their character-representation in input area “Text”
- Font Type – here the type of font that has to be used for this text element can be chosen, following options are available to select from:
  TrueType: the standard fonts that are installed on the system can be used. When this option is selected, the combo box directly below of this one is enabled to select a font out of it; all these fonts are in no way optimised for output via a laser scanner system, their intended target is publishing, therefore their lines and paths are optimised for quality which may result in a slow output
  Laser Vector: this special type of fonts is optimised for laser marking operations. They consist of as less as possible lines and are designed for single-lined outputs like they are done with lasers. Using these fonts marking of texts is much faster than with the TrueType fonts. When this font type is selected the font list below of this combo box is modified to show the available laser fonts to choose from.
- Font – this list of available fonts can be used to select a specific font and style for the text. Depending on the font type chosen above, here a specific font can be selected. In TrueType mode the number and types of available fonts depends on the operation system and which ones are installed there. In Laser Vector mode the internal laser fonts are listed here, these fonts are delivered with BeamConstruct and do not depend on the fonts installed on the system.
- Alignment – the text is always drawn from a fixed position that is specified by the left mouse click when setting the text initially; using this list it can be chosen how the text has to be aligned to this position (left, centred or right)
- Orientation – this list specifies the direction the single characters of the text have to be drawn
- Spacing – this value can be used to modify the distance between the characters; here a value of 100% is the distance that is specified by the used font, smaller values set the characters closer to each other, larger ones stretch it to a bigger distance
- Kerning – this value can be used to influence the kerning distance for fonts that support it, here 100% is equal to the kerning value that it specified by the font, bigger values set the characters with a kerning value closer to each other, values smaller than 100% set a bigger distance between them; this parameter is different to the “Spacing”-value, it only influences character-combinations where kerning is defined for
- Arc Length – when creating geometry out of the available font information depending on the style of the chosen font different round parts have to be created (like the two arcs that are needed for creating a letter “B”). This parameter specifies the segment length for these round parts, the smaller it is the more accurate the result is but the more data are used to hold it.
- Nominal height – here the nominal height of a line of text can be specified in unit mm; this does not only include all possible characters and their heights but also the distance between two lines of the chosen TrueType font. So the height specified here in most cases will be always bigger than the vector height of every possible characters of the chosen font.
  Beside of that this is the nominal value on creation of the text, means no additional scaling operations which are done with this element or with optional parent groups elements are included in this value.
- Set monospaced – when this option is checked, equal distance is set between all characters independent from their shape and independent from the distance information of the used font
- Lock size – this option can be used to force a text element to always keep a specific size. When this option is enabled, the current geometric size of the element is stored. Whenever the text changes –

either by manual modification, by remote access via a command interface or because the related input element changes the data – the element is scaled in a way so that the size is left unchanged. Depending on how much the real length of the texts changes, this means the characters itself are shrunk or stretched

To create radial text this element can be combined with the Curve Post-Processing Tool (please refer below)

6.9.4.11 Barcode

Using this function several types of 1D and 2D barcodes can be created. This element accepts static data for creating a barcode or dynamic values that can be set via an Input Element. The size of the created barcode can be defined freely, when creating this element a rectangle is drawn exactly in the same way like it is done for the Rectangle Primary Element. This rectangle describes the outline of the barcode. For some barcode types that use a fixed aspect ratio between width and height the predefined outline is modified automatically.

The complete layout and structure of the barcode is defined via the parameters that can be set within the elements tab-pane:

- Data – the text or numeric data that are encoded within the barcode, here it depends on the chosen barcode type if and which characters, numbers or combinations of them are allowed and will result in a valid barcode. This means it is not possible to enter any kind of data for any type of barcode! For most barcode types special formatting rules need to be respected, elsewhere no barcode is created and the primary element stays empty.
- Type – here the type of barcode can be chosen, depending on this type several of the following options are available and can be used for further definitions that are necessary for this type
- Mode – this is a barcode-type-specific option, here an encoding mode can be chosen for some of the barcodes
- Error Correction – this is a barcode-type-specific option, here an error-correction level can be set
- Size – this is a barcode-type-specific option, here the resulting size can be specified (which not influences the dimension of the barcode but the encoded data size)
- Bar Width – this parameter influences 1D barcodes only, here the size of the resulting bar lines can be modified; this value might be changed depending on the method of material processing in order to get a readable barcode
- Segments – this parameter is used only for the Maxicode barcode, it specifies the number of segments and therefore the smoothness of its circles
- Fade In Size – this parameter is used for inverted barcodes, it defines the size of the additional frame that is drawn around the barcode to get an inverted representation
- Force to Square – this option applies to DataMatrix barcodes only, in case it is set, the resulting barcode is always a square and never a rectangle
- Merge Cells – several barcode types may consist of cells (rectangles or squares) that are located directly beside each other; this option merges such cells wherever that is possible without destroying the barcode information, it can be used to save vector data
- Draw Cells as Dots – when this option is enabled, the cells some barcode types consist of are replaced by plain dots
- Draw Cells as Custom Dots – when this option is enabled, the cells some barcode types consist of are replaced by the custom dot geometry as defined within the settings (please refer to section 6.8.1 above)
- Invert – barcodes created by this element consist only of the outlines, together with the Hatching Additional Geometry element these outlines can be filled. Depending on the used material processing method these filled areas may result in lighter or darker areas. In case the result is not readable by barcode scanners the light and dark areas can be exchanged by this option, here an additional border is drawn around the barcode so that the hatching function generates an inverted fill pattern
- Invert with embedded normal – this option combines the normal mode and mode “Invert” into one

element. When this mode is enabled, the visual representation of the barcode is switched to “inverted” with the possibility to define fade in size for the additional bounds created around it. This inverted barcode will make use of the pen assigned to the element as usual.
Additional (but not directly visible) a non-inverted, normal barcode is created too making use of exactly the same input data like the inverted one. This non-inverted barcode is located at exactly the same position overlapping its inverted pendant perfectly. When this mode is enabled, the pen to be used for the embedded, non-inverted (=normal) barcode can be selected using the pen combobox which is shown right below the checkbox of this mode.
This special mode can be used e.g. in cases where both, inverted and normal barcode needs to be marked with different pen parameters in order to have a good contrast on resulting material. With this mode it is not necessary to manually overlap two separate barcodes and it is guaranteed they both have the correct position and the same input data.

6.9.4.12 Delay

This element does not have a visual representation, it does not accept any additional data or post-processing but it can be used to influence the material processing. The elements of a project are processed in the order they appear within the Element Tree. At the position where this element is added, the full process is delayed for the given time. Thus this element can be used whenever a delay is necessary e.g. to let material become dry, let material become colder or whatever.

This element expects exactly one parameter:

- Delay – the delay (in unit seconds) the full process has to stop

PLEASE NOTE: the accuracy of the delay given here depends on the used hardware. When the scanner controller supports delays, it is handled by it directly and is normally very exact. When the scanner controller does not support this, the delay is done within BeamConstruct and much less accurate since timer resolution of the host operating system is quite low and transmission times to and from scanner controller card are unpredictable.

6.9.4.13 Motion

This element does not have a visual representation, it does not accept any additional data or post-processing elements but it can be used to influence the material processing in general. The elements of a project are processed in the order they appear within the entity tree. At the position where this element is placed additional motion axes (apart from the axes of the used scanner) can be driven. These axes have to be connected with the two additional CTRL-outputs of the “BeamConstruct to Control”-plug-in within an ControlRoom project that makes use of this BEAMP-file.

The motion can be influenced by the parameters that have to be set for this element. First of all up to three axes can be controlled by the element using following parameters:

- Enable – specifies if an axis has to be changed by the following parameters or if it has to be left untouched, axes that are not enabled stay at their current position
- Position – the new position to move the axis to; this field is highlighted in yellow when the current settings of the axis may lead to a situation where the position value entered here is not used but overwritten during marking. This is possible e.g. in cases where an axis is bound to the height of a slice while marking 3D slice groups. For additional information please refer to axis settings as described in section “6.6 Project Configuration”.
- Speed – the speed for the movement to the given position
- Relative – defines a non-absolute movement, means the axis does not drive to the given position but changes its current position by the given distance
- Stop Axis – the motion of an axis is stopped immediately, this option is available only for motion controllers that support this function and ignores the given position and speed values
- Move axes to home-position – the axis is moved to its home / reference position, this option is available only for motion controllers that support this function and ignores the given position and speed values

All other parameters influence the whole motion element:

- Pre-Motion Delay – this delay is done before the motion is triggered
- Use Control Output – the “BeamConstruct to Control”-plug-in provides two CTRL-outputs where motion control data can be emitted, this option gives the possibility to define which of both outputs OUT2 or OUT3 have to emit the motion information defined for this element

6.9.4.14 Custom Output

Most scanner controller cards provide several analogue and/or digital outputs. As soon as a scanner controller card is configured within the project settings this element can be used to send values to these outputs. This Primary Element does not contain any geometry so clicking the related tool-bar button immediately adds it to the current project.

The exact layout of the related property panel highly depends on the capabilities of the used scanner controller card. There options are shown depending on the outputs the used card provides.

So it is possible that following types of outputs can be enabled for modification and set within this element:

- Laserport – a numerical value according to the valid range of the (LP8) laserport (e.g. 0..255) can be entered, this value is sent to the card during processing in case the related check box is enabled
- Digital – several buttons are shown that influence the related output ports bits and can be toggled into different states: “X” means this bit is left unchanged, “1” sets a bit to HIGH and “0” clears it; here the first button is the highest bit while the last button is always bit 0
- Analogue – a numerical value according to the resolution of the analogue output (e.g. 0..1023) can be entered, this value is sent to the card during processing in case the related check box is enabled
- Pulse Output – when this option is not set the output(s) simply is/are set according to the settings that are done above. In case this option is enabled and a time in unit milliseconds is given, the output it switched to the defined state only for this time, after that it is set back to its previous state. So this option can be used to emit only pulsed signals instead of a permanent output value.

PLEASE NOTE: The scanner controller card has to be set up and configured BEFORE this element is used for the first time. This is necessary because the capabilities of the scanner controller card need to be known for this element so that it can show the correct output options that belong to the card. When a completely different scanner controller card is selected for a project, it is necessary to save this project, close and restart the application to force a re-initialisation, load the project again and then check if all Custom Output elements still use the correct ports and values.

This element is symbolised by this toolbar icon: 

6.9.4.15 Custom Input

Most scanner controller cards provide several digital inputs. As soon as a scanner controller card is configured within the project settings, this element can be used to read values from these inputs and to halt processing of current project until a predefined state is found there. This Primary Element does not contain any geometry so clicking the related tool-bar button immediately adds it to the current project.

The exact layout of the related property panel highly depends on the capabilities of the used scanner controller card. There options are shown depending on the outputs the used card provides.

So it is possible that following types of inputs can be enabled and used within this element:

- Digital – several buttons are shown that read the related input ports bits and can be toggled into different states: “X” means this bit is ignored, “1” expects a HIGH value at this input and “0” a LOW; here the first button is the highest bit while the last button is always bit 0
- Serial interface – when supported by the used scanner controller card, data can be read from the serial interface periodically by this element. A comparison is done with the data read from this serial interface, here it can be selected if this has to be done with a full string (complete match) or with a

substring (partial match).

- Display message during wait – when this option is set, a text can be entered which then is shown during operation as long as the software is waiting for the input signal to arrive as specified by the options above. This text is shown in a top-level window which is closed only when mark operation is stopped or when the expected input signal is detected and processing of the current project continues

This element is symbolised by this toolbar icon: 

6.9.4.16 External Trigger

This element does not contain geometry. After clicking the related tool-bar symbol it is added to the project immediately. Whenever the project reaches this elements position during execution, all output is stopped until an external trigger signal is detected. Here the external trigger according to the used scanner controller cards capability is used. Within the elements tab-pane the controller card has to be chosen, where the external trigger will appear.
For some controller cards, it is also possible to release such a trigger automatically when a given distance has been passed by (in marking on-the-fly applications), then the application does not wait for an external start signal. When this option is chosen by selecting the related checkbox “MOTF-Distance”, an encoder signal needs to be available and need to count into the correct direction, elsewhere the project will wait endless at this position.
This element also accepts an input element such as the CSV Data Input. When such an input element is configured, given input data are used to set a new MOTF-distance. The input data need to be whole-numbered values in unit micrometers. When a value equal or smaller than 0 is given or when invalid data (such as some texts) are given by the input element, the MOTF-distance is not modified and stays unchanged.

PLEASE NOTE: starting the geometry after a trigger is received is done by the scanner controller, the time between detection of the external trigger signal and output of the next geometry depends on the used scanner card and typically is less than some microseconds.
Waiting for a trigger after the last geometry has been sent (or after the last Primary Element was processed completely) is controlled by the application. Here the reaction time highly depends on the used operating system and the available computing power. So it is recommended to ensure a gap of of at least 20 milliseconds after the last element was processed until the external trigger may appear.

This element is symbolised by this toolbar icon: 

6.9.4.17 Serial port output

This element does not contain any geometry but can be used to output some data to a serial port connected to the host PC where the application is running at. To submit data via the serial interface integrated into a connected scanner controller, please refer to section “6.9.4.14 Custom Output” above.

Within the element tab-pane the serial port has to be configured which has to be used for data transmission. This includes the port name, transmission rate and port parameters.
Below of the serial port configuration elements, a text field “Data” is available where the data have to be entered which need to be submitted. Here all types of ASCII-texts can be given as well as the two special place-holders `[CR]` and `[LF]` for adding a carriage return (“`\r`”) and a line feed (“`\n`”) to the data. A line break done in the text field is not converted to such characters but dropped, so it is mandatory to use these place-holders at all positions where this is required.
This element also accepts an input element such as the CSV Data Input. When such an input element is configured, given input data are used to set new data to be sent via the serial port. These input data need to follow the same rules as described above in order to add correct line feed/carriage return characters to the data stream.

Sending of the data is done synchronously, means when such an element is found, application waits until processing of data is finished for all scanner controller cards. Then the data are sent but without waiting for a

response before processing the next element in current project.

This element is symbolised by this toolbar icon: 

6.9.4.18 Z-Shifter

This element does not have a visual representation but is similar to the motion primary element. It does not accept any additional data or post-processing elements but it can be used to influence the material processing in general by changing the lasers focus. The elements of a project are processed in the order they appear within the entity tree. At the position where this element is placed, a focus shifter or beam expander is accessed. These axes have to be connected with the two additional CTRL-outputs of the “BeamConstruct to Control”-plug-in within an ControlRoom project that makes use of this BEAMP-file. In case only BeamConstruct is used, the related Z-Shifter plug-in has to be configured in projects hardware settings.

The focus/z-position can be influenced by the parameters that have to be set for this element:

- Position – the new position to move the z-position to
- Speed – the speed for the movement to the given position
- Relative – defines a non-absolute movement, means the axis does not drive to the given position but changes it current position by the given distance
- Move axes to home-position – the axis is moved to its home / reference position, this option is available only for motion controllers that support this function and ignores the given position and speed values

All other parameters influence the whole motion element:

- Pre-Motion Delay – this delay is done before the motion is triggered

6.9.5 Additional Geometry Elements

Additional Geometry elements add new geometry based on the already available geometry of the primary element it belongs to. Here the additional element only takes care of the primary one, it ignores possibly existing, other additional elements that might exist for the same primary element. Additional Geometry elements do not modify any of the existing geometries but only add own ones to them.

To add an Additional Geometry element to an existing Primary Geometry, the drawing area doesn't has to be used. Here the desired primary element has to be selected within the element tree, then the additional element has to be clicked within the tool-bar. There the Additional Geometry elements are shown in purple colour and are added to the Primary Geometry element as soon as the tool-bar item is clicked.

6.9.5.1 Hatcher

Hatching is the process of filling a closed polygon with a specific pattern. This can be used to process a bigger area. It can be useful e.g. combined with texts to get characters that consist not only of its outline.

Hatching will work only with closed polygons and might fail on specific geometries that do not describe a closed area non-ambiguous. In such cases the resulting hatch pattern may be incomplete or may be located outside of the polygon.

In general only two-dimensional geometry can be hatched, lines and dots are not extended by this element. As an other condition a polygon can be hatched only at positions where the outline has a closed line style, geometry with dotted lines can't be hatched, geometry with a dashed outline is hatched only at positions where a line is drawn.

Also in case 3D mode or 3D geometry is used the hatcher offers a 2D-functionality only but it can be applied to 3D elements. In such cases the hatcher always uses the geometry in X and Y direction (similar to the top view) and ignores the Z-value to create the hatch pattern. After this pattern is generated the Z-information

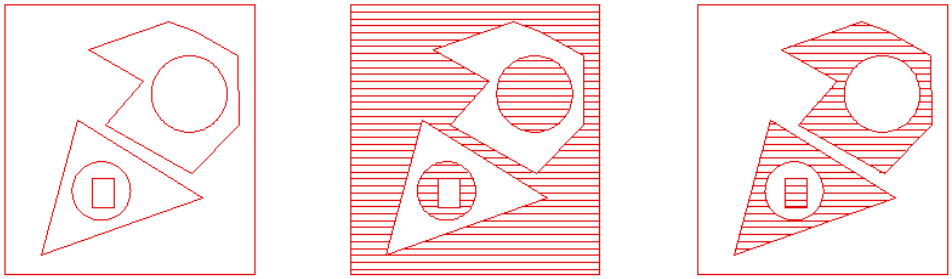
out of the lines they start from/end at are used to modify the hatch lines. This results in perfect 3D hatch patterns in most cases but may have unexpected results when complex geometries are combined with more advanced hatch options (like beam compensation or inner hatch lines).

The hatcher offers following parameters:

- Line Style – the style the lines the text consists of is created with, here following options exist:
  Continuous: the geometry consists of closed, connected lines with as less as possible points
  Connected Lines: the geometry consists of several short lines including additional points, all these lines are connected
  Dashed (simple): the geometry consists of several short lines including additional points, every second line is drawn that results in a dashed line style; this mode starts with a line at every beginning of a hatch line
  Dashed (interleaved): the geometry consists of several short lines including additional points, every second line is drawn that results in a dashed line style; this mode starts with a line and a space alternative at every new beginning of a hatch line
  Dashed (continuous): the geometry consists of several short lines including additional points, every second line is drawn that results in a dashed line style; this mode starts with a line or a space at the next beginning of a hatch line depending on the previous hatch line, the pattern is kept continuous
  Dotted: no lines are drawn, the geometry consists of a set of not connected dots
  Dotted (Custom): no lines are drawn, the geometry consists of a set of not connected points in custom dot style as configured within the parameter settings (please refer to section 6.8.1 above)
- Distance – whenever a non-continuous line style is selected, this distance value specifies the length of the additional points that form the selected line style
- Hatch Style – here the hatching mode is defined, depending on the selection made here a different pattern is used to fill a polygon
  Lines, forward – filling is done with parallel lines that go all into the same direction
  Lines, forward/backward – the polygon is filled with parallel lines that change their direction
  Connected lines – this mode extends the previous one, here the parallel lines are connected and form some kind of waved lines
  Zig-zag – these lines are no longer parallel, they exchange their direction alternative and are connected at their ends directly
  Inner lines – this mode repeats the outline of the hatched polygon in its inner side as often as it fits with the chosen hatch distance
  Outer lines straight – this mode is similar to "Inner lines" but not a plain hatch mode that can be used to really *fill* a geometry. It repeats the outline of the hatched polygon on its outer side as often as specified by the value given in field "Number of Lines". At corner points where additional lines are necessary to close the polygon, the shortest way is used by adding a straight line.
  Outer lines rounded – this mode is similar to "Inner lines" but not a plain hatch mode that can be used to really *fill* a geometry. It repeats the outline of the hatched polygon on its outer side as often as specified by the value given in field "Number of Lines". At corner points where additional lines are necessary to close the polygon, a round shape is created. Please note: this adds a lot of polygon points and may – dependent on the geometry – increase memory and processing time consumption dramatically. So this mode should be handled with care and only in cases where it is really necessary to use this special shape!
- Hatch Angle – this angle specifies the orientation of the hatch lines according to the coordinate system, it does not apply to the inner/outer lines hatch modes
- Angle Offset – this parameter applies only in case the hatcher is set as sub-element of a group that contains more than one geometric element to be hatched; in this case only the first primary element within that group is hatched with the original hatch angle, for all following elements the hatch angle is modified by this offset. So this parameter can be used to have some kind of rotating hatch angle for several elements that are concatenated together within a group. It does not apply to the inner/outer lines hatch modes.
- Repeat – this option becomes enabled when an angle offset is defined with previous parameter. When “Repeat” is set to values greater than 0, the hatch generation is repeated for the given number, each with a different angle according to the given offset. So this function can be used to create complex hatch patterns with rotating angles while using one hatch element only.
- Hatch Distance – this is the distance that is kept between the hatching lines, for the zig-zag hatching

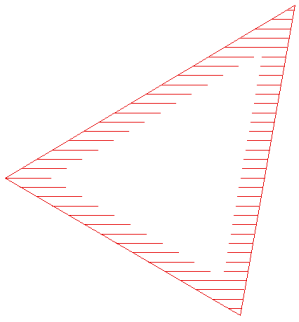
mode it applies to the ends of the lines

- Hatch Offset – normally hatching starts at the border of a polygon; this parameter allows to specify an offset (in positive and negative direction) to modify the position where hatching really starts
- Number of Lines – this input field is activated only when "lines" hatch styles are used; it specifies the maximum number of inner or outer lines to be calculated; when this value is smaller than the number of inner lines required to fill a geometry completely, inner part of the element will contain a non-hatched hole
- Interleave Lines – when this parameter comes with a value greater than 1, hatch lines are not marked one after an other but by dropping that number of lines after each, that is specified by this parameter. The remaining lines then are marked in additional cycles skipping the same number of lines in between. This mode slows down marking process caused by many additional jumps but gives the possibility to let the material not heat up too much. This can be useful in case of temperature-sensitive materials, there the currently marked areas get some time to cool down until the lines in their direct neighbourhood are processed. As an example: instead of marking lines 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 with an interleave-value of 4 it would mark lines 1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 12.
- Invert – this function applies to nested polygons only: when it is set, the order of hatched/not hatched polygons is inverted as shown in picture below:



From left to right: input geometry with nested polygons, hatched geometry (no-inverted), inverted hatched geometry

- From bounding box – hatching is started from the (not visible) bounding box, when this option is set, hatch lines start outside of the polygon; this function should not be used in cases where the bounding box is the same as the shape of the hatched geometries which is true for all rectangles which are aligned parallel to the working area border
- At border – when this option is set, polygons are no longer filled completely but only at the borders, the given distance in unit specifies the “thickness” of the resulting border hatch in unit mm:



- Beam Compensation – the laser beam that is used to process a working piece itself has an own size that results in the width of the lines that are processed by it; for the hatcher it might be necessary to compensate this lasers spot size to avoid parts of the working piece to be processed more than once by the overlapping beam, this compensation can be chosen here:
  Use spotsize from pen – here the tool compensation sizes are calculated automatically out of the spot sizes of the used pens
  Custom –when this option is used a custom value for compensation can be entered

6.9.6 Post-processing Tool Elements

The Post-Processing Tools influence all geometric elements that are located on an earlier position within the Element Tree and below of the same primary element. So the manipulation of geometry that is done by such a tool is always applied to the Primary Geometry element it belongs to and to all previously applied Additional Geometry elements of this primary one. Additional elements that follow a tool are not changed, they apply their additional geometry to the modified geometry.

Post-Processing Tools are symbolised by red tool-bar buttons and can be added to a primary element simply by clicking this tool-bar button. To do that, the related Primary Geometry element has to be selected within the Element Tree.

Post-Processing Tools do not add any new points or lines to existing geometry, thus it requires an outline that consists of lines that are short enough to let the tool apply its function, elsewhere the result would not look like expected. So within the parameters of the elements that have to be modified the line-style has to be set to a non-continuous mode (for an example connected lines, dashed lines or dots with a small distance should be used here).

6.9.6.1 Sine Distortion

This Post-processing Tool applies a sine waveform to existing geometry. This tool does not add any new points or lines to existing geometry, thus it requires an outline that consists of lines that are much shorter than the wavelength. Elsewhere the result would not look like a real sine waveform.

Following parameters can be set for this tool:

- Direction – the direction the waveform has to be applied to, beside the two values that are relevant for 2D data (“Horizontal” sets a sine parallel to X-axis, “Vertical” sets a waveform parallel to Y-axis) four additional options are available that are relevant for 3D data only. These options can be used to specify the direction of the amplitude and the direction the wave grows into
- Wavelength – the length of a single wave
- Amplitude – the amplitude of the sine wave
- Offset – an offset that defines at which position within the wave the sine calculation has to be started, this offset can be used to shift the waveform

6.9.6.2 Curve Distortion

Using this tool an existing geometry can be bent around a defined centre point to get a radial, round shape. It is useful e.g. in combination with the Text Primary Geometry element to define radial texts.

This tool does not add any new points or lines to existing geometry, thus it requires an outline that consists of lines that are short enough for a result that looks like a bent outline.

Following parameters can be used to influence the result:

- Alignment – this field specifies where the centre point is located relative to the geometry that has to be processed; the centre point can be located beside (X and Y direction) and below (Z direction) of the current geometry. To position it at the opposite side (right instead of left, above instead of below) a negative value for “Radius” has to be entered
- Radius – the radius of the circle the geometry is bent to, this is the distance from the middle of the geometry to the radial centre point
- Angle Offset – an offset for the bending angle starting point, using this value the geometry can be shifted

6.9.7 Data Input Elements

This type of element influences geometry only in a very indirect way, it provides data that can be used by Primary Geometry elements to create their geometry. So all types of Primary Geometry elements that accept

such input data can be combined with all types of Data Input elements. Of course it is not possible to have more than one Data Input element per Primary Geometry element.

To add such a Data Input element, the Primary Geometry element has to be selected within the Entity Tree. Next the green tool-bar symbol of the Data Input has to be clicked to assign it to the Primary Geometry element.

6.9.7.1 CSV Data Input

This plug-in offers connectivity to data out of tables in CSV format (comma separated values, a platform- and application-independent table file format). It loads data out of such a CSV table and jumps to next line/row of this file automatically during marking. The data read from this table are applied to the related Primary Element. Within the settings panel of this plug-in first the path to the CSV file has to be chosen, then the related input elements become enabled which can be used to configure the behaviour of this element:

- Column separator – here the delimiter character has to be chosen which separates the data columns within the selected CSV file
- Data column – using this field the column has to be specified where the data have to be read from, this value is 1-based
- Data repeat column – using this checkbox and input field a function can be activated where an other column in the same CSV file specifies how often the data from the previous configuration field have to be repeated until the next row of data has to be read out of the file. The repeat-value given here is 0-based, means with a value of 1 data out of the current row would appear two times (once the original value and once repeated due to the repeat column value)
- Endless loop – when this checkbox is set, the application will jump to the beginning of the CSV file as soon as all data have been read. When this option is not set and when there are no more data available within the CSV file, a marking process will be cancelled with a related error message.

During marking in Element Tree the name of this element is prepended by two numbers showing the current row of the loaded CSV file and the maximum number of rows it contains. Using these displayed values it can be seen how much CSV data of the current file are already processed and how much are still left.

6.9.7.2 OpenAPC Data Input

This input can't be previewed within the editor of BeamConstruct. It expects input data when the BeamConstruct project file is running within an ControlRoom project. This can be done by using the “BeamConstruct to Control” plug-in. This plug-in provides three Char-inputs IN2, IN3 and IN4. Data that are set at this input are used by this Data Input element to modify the assigned Primary Geometry element. This way a BeamConstruct-project can be changed dynamically directly within an ControlRoom project.

Because the data are provided by the ControlRoom project here, only the input has to be defined that has to be used for setting data into the BeamConstruct project. Whenever new Char-data arrive at this input within a ControlRoom environment they are used to re-create the assigned Primary Geometry element for the next marking cycle.

As an alternative possibility the three Char-inputs of this element also can be feed with new values out of BeamServer, it provides separate commands to send new values to any of the inputs (please refer to 6.20 for more details).

6.9.7.3 Serial/Date/Time

This is a very complex Data Input element that can be used to generate numbers, date or time information or combinations of them. The result can be used to create e.g. complex serial numbers, “best-before” information or others within a text or a bar code – depending on the Primary Geometry element where this input is assigned to.

The Data itself are updated before the next material processing cycle is started. Depending on the used configuration this update increments/decrements the serial number and/or updates the date or time

information to represent the current time.

Following parameters can be used to influence the resulting data and their behaviour:

- Start Value – the starting value where the serial number has to count from when the project is loaded initially
- Reset At – here a number can be defined that is used as limit for the serial number, as soon as the current serial number is equal to the value given here the serial number is reset to the start value defined above
- Increment – the value the serial number has to be incremented by for every update; this value can be negative which results in a decrement for every cycle
- Beat Count – this number specifies the number of cycles a serial number has to be left unchanged until it is updated, here a value of three – as an example – would mean, a serial number is updated and incremented only at every third marking cycle
- Beat Offset – this parameter corresponds to the preceding one, it adds an offset to the beat count, means it defines at which number it has to be started to count for the number of Beat Counts
- Numeric Base – by default a serial number consists of numerical values in range 0..9 which means the numeric base is 10; this parameter can be used to define any other numeric base so that less numeral values are used (Numeric Base < 10) or in a way where letters in range A..Z are added for counting the serial number (Numeric Base >10)
- Minimum Digits – this parameter specifies the number of digits a serial number has to consist of at minimum – independent from their real value; in case a number would have less digits than specified here, it would be filled with one or more preceding zeros
- Time Offset – here an offset can be specified using freely selectable numbers of years, days, hours and minutes, this offset is added to the time information that can be incorporated into the resulting data and can be used e.g. to define a best-before date counted from the current date/time
- Format String – this format string defines the layout and contents of the resulting data which are handed over to the assigned Primary Geometry element. Here any desired text can be added and combined with some place-holders which are replaced by serial number or date or time information. Here following place-holders are possible:
  - $S – this string is replaced by the serial number as defined with the parameters above
  - %A – is replaced by national representation of the full weekday name
  - %a – is replaced by national representation of the abbreviated weekday name
  - %B – is replaced by national representation of the full month name
  - %b and %h – are replaced by national representation of the abbreviated month name (these placeholders are not supported when using Windows)
  - %C – is replaced by the year divided by 100 (this placeholder is not supported when using Windows)
  - %c – is replaced by national representation of time and date
  - %D – is equivalent to "%m/%d/%y" (this placeholder is not supported when using Windows)
  - %d – is replaced by the day of the month as a decimal number (01..31)
  - %e – is replaced by the day of month as a decimal number (1..31); single digits are preceded by a blank (this placeholder is not supported when using Windows)
  - %F – is equivalent to "%Y-%m-%d" (this placeholder is not supported when using Windows)
  - %G – is replaced by a year as a decimal number with century. This year is the one that contains the greater part of the week (Monday as the first day of the week)
  - %g – is replaced by the same year as in "%G", but as a decimal number without century (00..99)
  - %H – is replaced by the hour (24-hour clock) as a decimal number (00..23)
  - %I – is replaced by the hour (12-hour clock) as a decimal number (01..12)
  - %j – is replaced by the day of the year as a decimal number (001..366)
  - %k – is replaced by the hour (24-hour clock) as a decimal number (0..23); single digits are preceded by a blank (this placeholder is not supported when using Windows)
  - %l – is replaced by the hour (12-hour clock) as a decimal number (1..12); single digits are preceded by a blank
  - %M – is replaced by the minute as a decimal number (00..59)
  - %m – is replaced by the month as a decimal number (01..12)
  - %p – is replaced by national representation of either "ante meridiem" ("AM") or "post meridiem"

("PM") as appropriate
- %R – is equivalent to "%H:%M" (this placeholder is not supported when using Windows)
- %r – is equivalent to "%I:%M:%S %p" (this placeholder is not supported when using Windows)
- %S – is replaced by the second as a decimal number (00..60)
- %s – is replaced by the number of seconds since January 1st, 1970, 00:00 o'clock, UTC (this placeholder is not supported when using Windows)
- %T – is equivalent to "%H:%M:%S" (this placeholder is not supported when using Windows)
- %U – is replaced by the week number of the year (Sunday as the first day of the week) as a decimal number (00..53)
- %u – is replaced by the weekday (Monday as the first day of the week) as a decimal number (1..7) (this placeholder is not supported when using Windows)
- %V – is replaced by the week number of the year (Monday as the first day of the week) as a decimal number (01..53). If the week containing January 1st has four or more days in the new year, then it is week 1; otherwise it is the last week of the previous year, and the next week is week 1.
- %v – is equivalent to "%e-%b-%Y" (this placeholder is not supported when using Windows)
- %W – is replaced by the week number of the year (Monday as the first day of the week) as a decimal number (00..53)
- %w – is replaced by the weekday (Sunday as the first day of the week) as a decimal number (0..6)
- %X – is replaced by national representation of the time
- %x – is replaced by national representation of the date
- %Y – is replaced by the year with century as a decimal number
- %y – is replaced by the year without century as a decimal number (00..99)
- %Z – is replaced by the time zone name
- %z – is replaced by the time zone offset from UTC; a leading plus sign stands for east of UTC, a minus sign for west of UTC, hours and minutes follow with two digits each and no delimiter between them
- %+ – is replaced by national representation of the date and time (this placeholder is not supported when using Windows)

- Locale – some of the date/time place holders are replaced by information that depend on a language and or on local specialities; using this parameter the locale can be chosen that has to be used for these time information

6.10 Edit Geometry

Geometries are mainly generated by Primary Geometry elements. Whenever the parameters of such an element are changed the geometry is re-created and all assigned Additional and Post-Processing Elements are executed again to apply their modifications to this new geometry. Thus modifications that are done for single vectors get lost as soon as such modifications are done. Nevertheless it is possible to perform such modifications in case one of the two conditions are met:

- the parameters of the elements that create a geometry are not touched
- a geometry is converted to a static one where no Primary Geometry element with variable parameters is available and where no Additional Geometry or no Post-Processing Element is assigned

Functions to edit geometries can be found at two places: in menu “Edit” and its sub-menus “Non-Destructive” and “Destructive” within the main window and within the context menu of the drawing area that opens when an element is selected and the right mouse button is clicked.

6.10.1 Edit Vectors

Editing of one or more separate vectors requires selection of these vectors. This can be done within the Coordinates-list of the Geometry-tab-pane when an element is selected. Here following functions are available:

- Delete selected Points – all selected points are deleted, resulting from that the lines between them are removed too

- Delete selected Lines – all lines between the selected points are removed, the points itself are left untouched
- Insert point before – a new point is added in the middle between the currently selected point ant its preceding one
- Insert point after – a new point is added in the middle between the currently selected point ant its subsequent one

6.10.2 Edit Non-Destructive

The non-destructive editing functions perform modifications that leave the geometries and the elements they are created from untouched. Some of them are available also from within the context menu of the Element Tree. Alternatively they can be found in Menu “Edit” sub-menu “Non-Destructive” and require at least one element to be selected:

- Group – several selected Primary Geometry elements are concatenated together within a group
- Group to Active Split – using this function geometries that are larger than the available working area can be cut into smaller pieces and processed step by step using an additional drive; for more information please refer the related section below
- Group to Active Move – here full geometries can be processed repeatedly and combined with movements so that the same geometry is arranged in tiles or at the circumference of a ring several times; for more information please refer the related section below
- UnGroup – an existing group of elements is removed, the contents of the group are put out of it
- Duplicate – a copy of the currently selected element is created, the copy is 100% identically to its original (except its position, it is placed a bit above the original element)
- Align left – this function can be used to left-align several selected elements at each other. Here the left-most-position of all these elements is taken to place all elements at this position in horizontal direction
- Centre horizontal – this function can be used to centre several selected elements along each other. Here the centre-position of all these elements is taken to also centre-align all elements at this position in horizontal direction
- Align right – this function can be used to right-align several selected elements at each other. Here the right-most-position of all these elements is taken to place all elements at this position in horizontal direction
- Align top – this function can be used to top-align several selected elements at each other. Here the top-most-position of all these elements is taken to place all elements at this position in vertical direction
- Centre vertical – this function can be used to centre several selected elements along each other. Here the centre-position of all these elements is taken to also centre-align all elements at this position in vertical direction
- Align right – this function can be used to bottom-align several selected elements at each other. Here the lowest position of all these elements is taken to place all elements at this position in vertical direction

6.10.2.1 Active Split Group

This is a special kind of group that concatenates several elements into one group. Due to the kind of this group it is recommended to use only geometries and scanner bitmaps and not to put other ones like motion elements into it.

An Active Split Group is able to handle geometries and bitmaps that are larger than the available working space. It cuts them into smaller pieces and combines them with motion control information so that these large and cut geometries and bitmaps can be put together during processing. So this functionality can be used to handle big working pieces that e.g. are moved by a XY-table. The cutting operation only applies to X-

Y-direction (equal to top view).

An other application is processing on the surface of rings, here a special radial mode is supported. This mode gives the possibility to cut geometries into segments and to process the whole ring (inner or outer side).

As soon as some geometric elements are grouped together within such an Active Split Group it's behaviour can be modified using the parameters that are available within the property panel of this group:

- Split horizontal – splits the grouped geometry along X axis, this option can be combined with “Split vertical” to define two-dimensional movements
- Split distance – the split size in horizontal direction
- Split vertical – splits the grouped geometry along Y axis, this option can be combined with “Split horizontal” to define two-dimensional movements
- Split distance – the split size in vertical direction
- Split radial – defines a splitting operation onto a ring, this option can not be combined with one of the previous splitting direction modes
- Direction – specifies the direction for the rotational splitting, here it can be chosen between “Horizontal” and “Vertical”, the axis that is assigned to this direction has to be aligned properly
- Diameter – the total diameter of the ring where the split pieces have to be marked at
- Split angle – the movement angle for every split, the size of the single split pieces result out of this angle and the diameter of the ring
- Keystone correction – when marking the inner side of a ring the laser beam has to be slanted in order to hit only the lower inner side of the ring and not the upper outer side too. Resulting from that the length of the beam varies dependent on the X/Y position of the geometry which results in a distortion when no special optic is used.
  Using the keystone correction this distortion can be suppressed so that the marking result will look like expected. This correction is done by setting an as a percentage correction factor and by defining the side of the geometry this factor may apply to. The correction factor specifies how much the geometry shall grow or shrink according to its original size at the outer position where the full distortion has to be expected.
- Invert motion direction – this option has to be used in case the inner side of a ring needs to be processed, there an inverted motion direction is required to concatenate the pieces in correct order
- Process order – this order applies to the planar splitting modes and defines in which order the pieces have to be processed:
  Left to right – horizontal direction always starts from left
  Right to left – horizontal direction always starts from right
  Zig-zag from left – horizontal direction starts from left and performs zig-zag movements
  Zig-zag from right - horizontaldirection starts from right and performs zig-zag movements
  Top to bottom – vertical movement starts from top
  Bottom to top – vertical movement starts from bottom
  In case two-dimensional motion is defined both process order selections are combined with each other to describe the 2D motion.
- Motion speed: the movement speed in unit mm/sec or degrees/sec that is applied to the motion between the splits.
- Overlap splits: here an as a percentage value can be set that specifies how much the pieces should overlap, means how much of the geometry should exists twice so that it is processed two times at the areas of intersection
- Try split between pieces – here the splitting algorithm tries to find empty spaces between geometries to avoid cutting of lines; in case this option is set the specified split distance or split angle is used as maximum value: a cut is set wherever space between geometries is found or latest after the given split distance/split angle. In last case cutting of lines is still done. So in this mode split distance/split angle define the absolute maximum size of a resulting piece while the distances where the geometry are cut really can be smaller than this.
- Axis mapping: at the lower end of the property panel a matrix of radio buttons exists where the axes

can be mapped to the related movements, there it can be specified which of the axes A, B or C of the two possible outputs OUT2 and OUT3 have to provide the related planar movements in horizontal or vertical direction or the angular movement.
Later, when such a project has to be used within a ControlRoom project, is has to be made sure the outputs OUT2 and OUT3 of the “BeamConstruct to Control” plug-in are connected with the correct motion plug-in. Within these motion plug-ins the related axes have to be configured correctly (planar or rotational mode) and they have to access the correct axes that belong to the defined motion modes

- Save as default – after application start-up new split-groups are created with some default parameters. When these parameters are changed, other Active Split Groups then are created with these changed parameters, but after restarting BeamConstruct, the default parameters are used again. When this button is pressed, the current split-group parameters are saved as new default so that they are used whenever a new Active Split Group is created after BeamConstruct was started newly. The related data are stored within the users working directory in a file named “.beamuiSPLT”

After such a splitting group has been created and after the correct parameters have been set, the appearance of the related geometry changes: now it is overlapped by horizontal and/or vertical lines. These lines specify the cutting positions and can be modified manually. They can be dragged with the mouse in horizontal or vertical direction exclusively to change the position where the geometry has to be cut. The related modifications are done as soon as such a cutting line is released.

Please note: the positions of the cutting lines do not influence the parameters given within the property panel but as soon as the parameters within the property panel are modified, the position of the cutting lines are reset to their default. Thus their positions should be changed only in case all other parameters do fit perfectly.

6.10.2.2 Active Move Group

This is a special kind of group that concatenates several elements into one group. Due to the kind of this group it is recommended to use only geometries and not to put other ones like motion elements into it.

An Active Move Group is able to repeat a grouped geometry several times and to combine it with movements so that e.g. a ring can be processed with a repeated pattern on its surface or to repeat the current geometry on a large working piece that itself is re-positioned by using an XY-table. The active movement functionality only applies in 2D mode or in X-Y-direction (equal to top view). Comparing to the Active Split Group here the contents of such a group are not cut or somehow else modified, they are marked all in once repeatedly.

As soon as some geometric elements are grouped together within such an Active Move Group the groups behaviour can be modified using the parameters that are available within the property panel of this group:

- Move horizontal – moves the grouped geometry along X axis, this option can be combined with “Move vertical” to define two-dimensional movements
- Move distance – the distance a movement has to be done in horizontal direction between two processing cycles
- Move steps – specifies how often processing of the geometry has to be repeated in horizontal direction
- Move vertical – moves the grouped geometry along Y axis, this option can be combined with “Move horizontal” to define two-dimensional movements
- Move distance – the distance a movement has to be done in vertical direction between two processing cycles
- Move steps – specifies how often processing of the geometry has to be repeated in vertical direction
- Move radial – defines a movement operation onto a ring, this option can not be combined with one of the previous splitting direction modes
- Diameter – the total diameter of the ring where the geometry has to be marked at
- Move angle – the movement angle for every split, the size of the single split pieces result out of this angle and the diameter of the ring
- Move steps – specifies how often processing of the geometry has to be repeated on the ring

- Keystone correction – when marking the inner side of a ring the laser beam has to be slanted in order to hit only the lower inner side of the ring and not the upper outer side too. Resulting from that the length of the beam varies dependent on the X/Y position of the geometry which results in a distortion when no special optic is used.
  Using the keystone correction this distortion can be suppressed so that the marking result will look like expected. This correction is done by setting an as a percentage correction factor and by defining the side of the geometry this factor may apply to. The correction factor specifies how much the geometry shall grow or shrink according to its original size at the outer position where the full distortion has to be expected.
- Process order – this order applies to the planar movement modes and defines in which order the grouped geometry have to be repeated:
  Left to right – horizontal direction always starts from left
  Right to left – horizontal direction always starts from right
  Zig-zag from left - horizontal direction starts from left and performs zig-zag movements
  Zig-zag from right - horizontal direction starts from right and performs zig-zag movements
  Top to bottom – vertical movement starts from top
  Bottom to top – vertical movement starts from bottom
  In case two-dimensional motion is defined both process order selections are combined with each other to describe the 2D motion.
- Motion speed: the movement speed in unit mm/sec or degrees/sec that is applied to the motion
- Axis mapping: at the lower end of the property panel a matrix of radio buttons exists where the axes can be mapped to the related movements, there it can be specified which of the axes A, B or C of the two possible outputs OUT2 and OUT3 have to provide the related planar movements in horizontal or vertical direction or the angular movement.
  Later, when such a project has to be used within a ControlRoom project, is has to be made sure the outputs OUT2 and OUT3 of the “BeamConstruct to Control” plug-in are connected with the correct motion plug-in. Within these motion plug-ins the related axes have to be configured correctly (planar or rotational mode) and they have to access the correct axes that belong to the defined motion modes
- Save as default – after application start-up new move-groups are created with some default parameters. When these parameters are changed, other move-groups then are created with these changed parameters, but after restarting BeamConstruct, the default parameters are used again. When this button is pressed, the current move-group parameters are saved as new default so that they are used whenever a new Active Move Group is created after BeamConstruct was started newly. The related data are stored within the users working directory in a file named “.beamuiMOVE”

After such a splitting group has been created and after the correct parameters have been set, the appearance of the related geometry does not change but within the entity list a different symbol for the group is shown in order to point to the fact a special kind of group is used here.

6.10.3 Edit Destructive

The functions of this category perform irreversible modifications to the input geometry. That means only the resulting geometry is used for further processing and all source elements that have been used to create this geometry get lost.

Resulting from that there is much less computing time and memory used to hold such geometry and editing on a per-vector-level is much less difficult because the data no longer can't be overwritten by a re-creation caused by some parameter changes. On the other hand the conversion to static geometry means that no more dynamic changes are possible like they can be performed when Data Input elements are used – as soon as these elements have been modified by such a destructive modification, all the (sub)elements it consisted of are lost. But afterwards is is possible to assign Additional Geometry and Post-processing elements to this static geometry again.

Following functions exist that perform such destructive modifications:

- Merge – the geometries of the selected elements are merged together, the result is one static element with all geometries included into it; this function can also be applied to one single Primary Geometry element and its sub-elements to reduce its memory usage to the absolute minimum so

that only the resulting static data of it are stored

- Split – tries to split the geometry of an element into several pieces, here splitting is done at positions where a geometry ends (e.g. at the end of a logic element or the end of a polygon). There are several levels of splitting so that it is possible to perform a splitting operation up to two times at the same data. As an example: when the splitting-function is applied to a geometry that represents the string “Text” for the first time, the result are four elements containing the geometries for the letters “T”, “e”, “x” and “t”. In a second step the splitting function can be applied to the geometry of the letter “e” again. Now it is split based on the contained poly-lines, the result are two elements that contain the two polygons the letter “e” consisted of (one with the outline and one with the small part within the head of the “e”).
- Optimize – this function checks geometry for inefficient structures and optimizes them. Here the visual representation of a geometry is not changed but the number of vectors and vertices is reduced if possible.
- Reduce Geometry – here the data a geometry consists of can be reduced. After selecting this menu item a dialogue is opened which provides different reduction methods and modes:
  - Smooth lines – this is not a real data reduction method but combined with the other options it can have stronger effects and better results; this expects a line-length in unit mm, when chosen all lines within a polygon that are smaller than this value are smoothed, means their values are combined in a way which tries to result in softer shapes with less edges (low pass filter); this function should be used with round shapes and can improve the effect of option “remove lines” when used in combination
  - “Remove lines” - here a parameter in unit mm is expected which specifies the minimum length a single line within a polygon shall have, lines that are smaller than this are concatenated wit htheir predecessor
  - “Remove points in angles” - this option expects a parameter in unit degrees and removes all points within a polygon which form an angle that is smaller than the specified value (means three points are inspected which form an angle and the middle point is removed when the angle is below the given threshold); the effect of this option becomes weaker when it is combined with function “smooth lines”
  - “Close polygon with endpoint” - this is not a real data reduction method but combined with the other options it can have stronger effects and better results; this function tries to create closed polygons out of a shape; when the end of a polygon line is found, it is checked if it is a closed polygon (means the end is located at the same position like its beginning) or not; in case it is not closed, the beginning of an other polygon line is tried to be found which is closer than the parameter entered for this function (in unit mm) – and when one could be found, the whole geometry is rearranged in order to get one continuous polygon; this function can improve the effect of option “remove lines” when used in combination
- Extrude – this function extrudes a flat, two-dimensional element into the third dimension. After clicking this menu item, a dialogue opens, where several options can be chosen which influence the extrusion result:
  - Extrusion axis – specifies along which axis the extrusion has to take place, this is done always parallel to one of the three axes X, Y or Z so that it is recommended to align and position the element properly before this function is called
  - Extrusion direction – specifies the extrusion direction (up or down, left or right, in our out)
  - Distance – in normal extrusion mode (3D mesh option not selected) this operation is performed by duplicating the element and by placing the duplicates close to the original. This value can be used to choose the distance between all these copies, the smaller is it, the more closed the resulting geometry is
  - Extrusion length – specifies the total extrusion depth, means how much the element shall grow into the selected direction
  - Create 3D mesh – then this option is set, the “Distance” parameter is no longer used. Here instead of a normal primary element a 3D mesh is created which is part of a slice group (please refer to related sections below) and which can be used to modify existing 3D meshes
- Delete Element(s) – deletes the selected element(s)

6.11 Process 3D Data

There are two ways to process 3D data in BeamConstruct: one possibility is to use their 3D information

including the Z (=depth) information to drive a 3D scanner and to mark them exactly as they appear. Alternatively a solid 3D model can be imported and cut into horizontal pieces (so called slices) which can be processed separately and optionally without the need to involve a 3D scanner system.

The first variant does not require any more than a scanner controller card that is configured for 3 axis operation. Resulting from such a configuration the user interface of BeamConstruct is switched into 3D mode too, now it is possible to edit Z-values within all property panels, to switch the drawing area to profile views or to a real 3D view and to draw and edit elements also in depth-direction.

6.11.1 Slicing 3D Data

The second processing method does not necessarily requires a 3D scanner, here a 2D system can be used too, the third dimension can be handled by external hardware (like a drive that moves the position of the working piece in Z-direction via a lift table). As a precondition to use this operation mode a ready-to use 3D model has to be imported. Currently BeamConstruct supports the RenderWare RWX format, the Surface Tessellation Language STL in binary and ASCII format, the WaveFront OBJ format and the PLY 3D printer format. Beside of that the Common Layer Interface CLI format is supported too, since this is a file format that already contains sliced data it can be used for processing but is not a real 3D format where a slicing operation can be applied to..

More directly supported 3D file formats will follow in future. In case 3D data are available using a different file format than the ones described above, a converter application can be used. Here we recommend AccuTrans 3D which is available at http://www.micromouse.ca/ and supports nearly every 3D file format that exists. AccuTrans 3D itself is available for an extremely low price.

After selecting a 3D file for import, a configuration dialogue opens where some initial slicing parameters can be configured. All the slicing parameters can be changed later within the application:

- Import Scale Factor – a factor that can be used to scale the whole 3D object during loading
- Slice Distance – the distance between every slice
- Slice Offset – specifies an offset from top/bottom of the 3D model where slicing has to start; in case this value is set to 0.0 it may result in a first slice that contains nearly no relevant data (dependent on the used 3D model)
- Reduction – expects a value in unit degrees to perform a lossy data reduction on sliced data; whenever an angle between two vectors of a layer is smaller than the angle given here, they will be concatenated to one common vector. For small angle-values and in most cases this results in a memory consumption which is dramatically lower and processing speed therefore much higher while there is no visible loss in quality of the result.
- Slice Direction – here the slice direction for the 3D model can be selected, “Top To Bottom” may be used e.g. for depth-engraving where operation starts on top of the model and moves into the material, “Bottom To Top” can be used e.g. for rapid prototyping applications where processing starts from bottom and builds up a model slice by slice
- Slice Mode – here it can be selected if the depth-information has to be handled by the Z-axis of a scanner or by usage of external equipment; when mode “Real 3D” is selected all the slices are located at their true Z-position according to the origin model. This mode is for usage with a 3D scanner.
  In case mode “Flat (Z externally driven)” is selected, all slices are located at Z-position 0 and the movement in depth-direction has to be done with a separate device. Control elements for this device can be added to a slice group later. In case the scanner card is configured for two-axis-operations, only the second option is available, in this case “Real 3D” is disabled and can't be selected
- Split to tiles – the working area is split into (invisible) tiles that are used to rearrange the contours of the sliced 3D mesh during marking. Using these tiles it can be avoided to heat up the material at a specific position too much, it lets the position where contour is marked make additional interruptions and jumps to other place in order to continue marking at a distant position (means a distant tile). When this option is set, the following two parameters can be used to specify behaviour of this feature. For a more detailed description and usage examples please refer below.
- Tile Size – the size of a tile in working area, this value should be such a fraction of the working area size, that lets all tiles fit fully into whole working area

- Tile Skip Size – skip distance between tiles, this value decides how many tiles in horizontal direction should be skipped before marking continues in next tile; remaining tiles are processed on next cycle; this value should be much smaller than the total number of tiles that fit into the working area

When this configuration dialogue is left by pressing the “OK” button the 3D model is loaded and sliced according to the specified parameters. Dependent on the size of the model, the number of slices that result out of this size, the slice distance and the complexity of the models geometry this operation may take a while. When the slicing operation is finished a “Sliced 3D Mesh” group element is added to the current project that contains the 3D model and the slices.

6.11.2 Sliced 3D Mesh Group

This is a very specific group that contains a 3D model that will not be marked and several slices which are generated out of this model. Such a group has to be combined with other elements carefully since it performs operations in Z-direction that have to fit to other elements or groups that are located before or after it within the current project. A Sliced 3D Mesh Group is shown as a group element with one 3D Mesh element for every imported 3D model, and one sub-element for every slice within the element tree on the windows right hand side.

When the group is selected a special property panel opens on the windows left hand side. There several parameters can be changed. Some of these parameter changes lead to a recalculation of the slices which means the currently existing slices, their sub-elements and possibly added motion-elements are removed from this group. So it is recommended first to optimise all the slicing parameters and then add more data and elements to such a group.

The Sliced 3D Mesh Group property panel itself is split into two major parts. Section "Model" contains all parameters that belong to the 3D mesh and the related data. Section "Support" is related to (automatic) creation of additional structures that are intended to help creating the model itself, especially at positions where parts of the model hang over and would not be stable during the phase of marking of single slices.

In section "Model" following parameters can be changed:

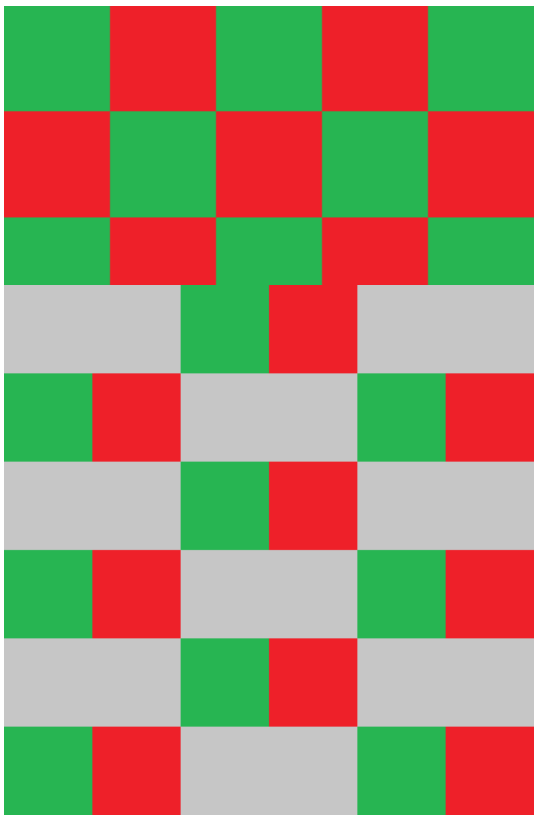
- Slice Distance – the distance between every slice; when this value is changed all slices and possibly existing additional elements are removed and the slices are generated again
- Slice Offset – specifies an offset from top/bottom of the 3D model where slicing has to start; in case this value is set to 0.0 it may result in a first slice that contains nearly no relevant data; when this value is changed all slices and possibly existing additional elements are removed and the slices are generated again
- Reduction – expects a value in unit degrees to perform a lossy data reduction on sliced data; whenever an angle between two vectors of a layer is smaller than the angle given here, they will be concatenated to one common vector. For small angle-values and in most cases this results in a memory consumption which is dramatically lower and processing speed therefore much higher while there is no visible loss in quality of the result.
- Slice Direction – here the slice direction for the the 3D model can be selected, “Top To Bottom” may be used e.g. for depth-engraving where operation starts on top of the model and moves into the material, “Bottom To Top” can be used e.g. for rapid prototyping applications where processing starts from bottom and builds up a model slice by slice; when this value is changed all slices and possibly existing additional elements are removed and the slices are generated again
- Slice Mode – here it can be selected if the depth-information has to be handled by the Z-axis of a scanner or by usage of external equipment; when mode “Real 3D” is selected all the slices are located at their exact Z-position according to the origin model, this mode is for usage with a 3D scanner; in case mode “Flat (Z externally driven)” is selected, all slices are located at Z-position 0 and the movement in depth-direction has to be done with a separate device. Control elements for this device can be added to a slice group later. In case the scanner card is configured for two-axis-operations only the second option is available, in this case “Real 3D” is disabled and can't be selected; when this value is changed all slices and possibly existing additional elements are removed and the slices are generated again
- Slice Pens – this table can be used to define patterns for the pens that have to be assigned to every slice. There separate pens can be specified for contour and hatch of a slice. First of all the check box

in front of a row of the list has to be selected to enable this entry. Then the pens can be selected via the combo box. Here the list rows have different meanings: “First Slice” is used for the very first slice only (that can be on top or bottom of the model dependent on the slice direction) and “Last Slice” is used for the very last slice only (that can be on top or bottom of the model dependent on the slice direction). All other rows within that list define an order of pens that is repeated until the last slice is reached. Here patterns of up to 5 different pen combinations can be defined.
Since the pen-definitions for sliced geometry have to be done here the related option within the “Element” tab-pane is disabled, pens can be assigned only via this pen mapping table.

- Show All Slices – in case this check box is selected all slices that belong to a 3D model are shown within the drawing area of BeamConstruct. This function has to be handled with care since a large amount of slices can slow down the application dramatically, here it is recommended to use “Display reduced” parameter to show only every nth number of slice
- Hide hatch from view – A hatched model can – dependent on the number of hatch lines calculated for a slice – slow down graphic visualisation quite dramatically. For such cases it is possible to enable this option which then disables drawing of the hatch patterns of a 3D slice group in order to have a faster and more smooth graphical representation of the current project
- Slice during marking – normally complete slicing of a model is done in advance and for the complete model. Depending on the complexity of a model such an operation can consume quite a lot of time while no other operations are possible. When this option is checked, only the very first slice is precalculated and all other ones are done during marking: while a slice is marked and the related inter-slice operations are done, the next slice is calculated and the related hatch-patterns are applied in parallel. Thus a lot of handling time can be saved since both operations – marking of a slice and completely preparing the next slice – are done fully parallel.
This function can be used only with models where no support-structures are generated, means activating this option automatically disables generation of support-structures.
After setting this option, slicing and adding additional elements to the generated slices has to be done exactly in the same way as before. But with this option set, only the very first slice is shown. All others are generated during marking and therefore are not available in advance.
- Display Reduced – reduces the number of slices that are shown in case option “Show All Slices” is selected, the bigger this number becomes the less slices are shown within the drawing area
- Split To Tiles – when this option is set, the working area is split into (invisible) tiles that are used to rearrange the contours of the sliced 3D mesh. Using these tiles it can be avoided to heat up the material at a specific position too much, it lets the position where contour is marked make additional jumps in order to continue marking at a distant position (means a distant tile). When this option is set, the following two parameters can be used to specify exact behaviour of this function. After enabling this option the 3D mesh is re-sliced but there will be no visible representation of the changes in working area, it can be noticed during marking only. This function influences only the contour of the slices, hatches are not interrupted and broken into tiles in order to avoid damaged marking results. To reduce heating of material during hatch processing, please refer to description of the hatcher above; there exists a similar functionality to skip hatch lines.Together with the following parameters this function can be used to e.g. mark every second tile:

  Assumed the working area has a size of 100 mm x 100 mm, for this behaviour "Tile Size" needs to be set to 20 mm and "Tile Skip Size" has to be set to 2. With these settings first all these contours are marked that have vectors within red tiles, finally all the remaining contours that have a vector in green tiles are marked. Tiles itself are processed from upper left to lower right corner row by row.
  It is also possible to have even bigger distances between tiles so that they do not touch at their edges:

  Parameters for a 90 mm x 90 mm working area are 15 mm for "Tile Size" and 4 for "Tile Skip Size". This results in a behaviour where marking starts with these contours that own vectors in red tiles, next the contours in green tiles are marked. This is continued until the whole working area was processed (in this example by processing the remaining two tile sets in grey areaswith two additional steps).



- Tile Size – the size of a tile in working area, this value should be such a fraction of the working area size that lets the tiles fit fully into whole working area
- Tile Skip Size – skip distance between tiles, this value decides how many tiles in horizontal direction should be skipped before marking continues in next tile; remaining tiles are processed on next cycle; this value should be much smaller than the total number of tiles that fit into the working area

Section "Support" offers the following options and parameters which are available only when a model is created from bottom to top (so this option is not available when doing deep-engraving with a Sliced 3D Mesh Group):

- Enable Support Structures – this checkbox has to be enabled in order to let the software automatically create support structures, when it is turned off, the model will be processed as is and with no additional geometries. When enabled, additional geometries are calculated and added to the model automatically at positions where an overhang could cause problems. These additional geometries are only contours and need to be extended by hatch-elements in order to get them filled and stable. Here for both, contour and hatch suitable pens and related pen-parameters have to be chosen in order to not to melt together the support structures and the model's structures.
- Distance to Model – here a distance can be specified that always has to be used at positions where the contour of model and support structure are neighbouring within a slice; using this distance it can be ensured the support structure can be removed easily after processing. When this value is too large, the effect of supporting the model may fizzle out.
- Enlarge Structure – the value given here can be used to make the support structures larger than necessary in order to overlap the model completely; this value has no influence on the distance to the model, it expands the support structure to its outer borders.
- Enclose Model Completely – normally a support structure is created from bottom to top until it is covered and replaced by parts of the model itself. This is done because support structures are created to inhibit negative effects caused by gravity and therefore are no longer necessary when there are (stable) structures of the model itself. Thus at some point there are no more support structures available but only the model. When this option is set, the support structure is created always, and will cover the whole model.
- Support Pens – this table can be used to define patterns for the pens that have to be assigned to every slice's support structure. There separate pens can be specified for contour and hatch. First of all the check box in front of a row of the list has to be selected to enable this entry. Then the pens to

be used for contour and hatch of a support structure can be selected via the combo box. All rows within that list define an order of pens that is repeated until the last slice is reached. Here patterns of up to 5 different pen combinations can be defined. Since the pen-definitions for sliced geometry have to be done here, the related option within the “Element” tab-pane is disabled, pens can be assigned only via this pen mapping table. Alternatively the pen-definitions for hatches can be disabled. To do that, the checkbox above the "Hatch"-column has to be unset. Now an own pen can be assigned to every hatch via the common way (pen-assignment in "Element" tab-pane). The difference between both methods: while pen assignment via the table in "Support"-section assigns the same pen to all hatches of a slice (which additionally may alter over slices), the pen-assignment via "Element" tab-pane can be used to assign different pens to each individual hatch that stay the same over slices.

Beside of that there is a slider available on the property panels left hand side: there the currently shown and used slice can be selected and it can be used to scroll through the sliced model. The same is true for the input field below of that slider: there a number of the current slice can be entered or chosen.

Such a Sliced 3D Mesh Group is shown as a group element with one sub-element for every slice within the element tree on the windows right hand side. Comparing to other groups it behaves a bit different. So it is possible to put new elements (like a motion control element or I/O elements) into that group directly. To do that, either the group node itself or one of its slices has to be selected and the Primary Element to be added has to be chosen within the tool bar. Now a dialogue is opened where the user can select if that element has to be added as a single one only, below of every slice or for a range of slices. This way it is possible to set motion controlling elements for every slice easily. More than this: all elements that are added via such a range will belong together afterwards: whenever a parameter of one of the elements is changed, the parameters of all others that belong to the same range (and therefore have the same number in its name) are modified too.

The same is true for Additional Elements: when there is a hatch added to a single slice the user can choose to apply it to a range of slices or to all slices. When the parameter of one of these hatches is changed the change is applied to all the other hatches that have been created together with this first one.

Using this way the same operations and parameters can be applied to all slices easily.

Between every slice-subelement of the group there is also a “Process”-element created automatically. This element does expect only one parameter: the path to an existing project file. This project file is executed at the position of every “Process” element and can contain processing-operations that are necessary between every slice (such as motion operations to move a Z-table, setting of digital outputs and waiting for digital inputs a to synchronise with external devices). Using such an external project it is not necessary to add these control-steps – which normally are always the same for a machine – manually for every newly sliced element.

6.11.3 Auto-Arrangement of 3D Meshes in Sliced 3D Mesh Groups

A Sliced 3D Mesh Group can hold several 3D Meshes. They appear as sub-elements of type "3D Mesh" in element tree. These 3D Meshes can be positioned with the standard functions in tab-pane "Geometry" in order to place them at suitable positions in working area. Alternatively it is possible to let BeamConstruct arrange these meshes automatically. To do this, the Sliced 3D Mesh Group has to be selected and right-clicked in element tree, then menu item "Auto-Arrange 3D Meshes" has to be selected. Now a dialogue opens which provides options to arrange the 3D Meshes in X- and Y-direction and to align them at a specific Z-height. This is done via the configuration possibilities within the dialogue:

- Arrange in X and Y – when enabled, the 3D Meshes are arranged on working area using the following parameters
- Distance to borders – specifies the minimum distance the 3D Meshes need to have to the border in both, X and Y direction of the working area
- Distance between meshes – specifies the minimum distance the 3D Meshes need to have between each other
- Arrange in Z – when enabled, the 3D Meshes are aligned at a specific height using the following options and parameters
- Align geometries at top or bottom – here it can be selected if all the meshes need to be aligned at

their upper or lower position

- Arrange in Z – this is an option which can be enabled when 3D Meshes have to be top or bottom aligned at a specific, absolute height; when it is set, the height has to be specified in unit mm; in case this option is not set but arrangement in Z is enabled, all meshes are aligned at the highest/lowest top/bottom position of that 3D Mesh, that is located at the highest/lowest position (according to its outlines top/bottom border)

6.12 Vision

BeamConstruct provides an integrated Vision system for automatic adjustment and correction of marking data. Using this vision system special markers – so called fiducials – can be teached. Later the Vision system is able to recognise such a fiducial on a working piece and adjust position and rotation of the current marking data according to the fiducials measured position. Vision functions can be found in menu “Edit” submenu “Vision” and in mark dialogue.

To use the vision system it is mandatory to connect a camera and to configure a Video Controller in hardware settings. This enabled the Vision menu entries and functionalities.

6.12.1 Vision Functions

Menu Vision → Start Video

When image capture is configured to be started manually this menu item can be used to start (menu item checked) and stop capturing (menu item unchecked)

Menu Vision → Freeze Video:

As soon as a Video Controller is configured, a live image is shown in background of the working area. This image changes permanently and can be freezed using this menu. This is a toggle menu, selecting it turns freeze mode on, selecting it a second time turns this mode off and enables the live-image again.

Menu Vision → False Colours:

In some cases captured images have a very low contrast in desired parts of the image. In such cases the shown image and the required details can be improved and made more visible by false colours. The sub-menu below of this menu item offers several false colour modes that can be used for this

Menu Vision → Calibrate Camera:

This functionality can be used to calibrate a camera and to adjust and correct distortions caused by the lens of the camera. This function does NOT correct camera angle/perspective distortions. When selecting this menu item, a calibration dialogue opens. For camera calibration following steps have to be performed:

1. A black/white checkerboard has to be placed in cameras view. Such a board can be found e.g. at http://halaser.eu/checkerboard.png. Here the board must be visible in cameras live image fully, none oft the squares have to be cutted.
2. The camera calibration dialogue has to be opened and the boards parameters have to be entered: here the number of crossing points between squares (the chessboards inner corners, not the number of squares itself!) have to be entered for horizontal and vertical direction.
3. The button “Try calibration” has to be pressed. It starts a calibration run where the checkerboards inner corners are searched. If this is successful all these crossing points are marked by blue circles and the “OK”-button of the dialogue is enabled.
4. The dialogue has to be left with “OK” in order to take over the calibration data. Now they can be saved by selecting menu “Project” → “Save as default configuration”.

Menu Vision → Crop Camera:

Here it is possible to clip the camera image, means to cut the borders of the captured image to limit the view to these parts, that are really needed. After selecting this menu item a dialogue opens where the horizontal and vertical crop values can be set – the result is shown via a preview, that contains blue lines marking the new borders of the image. Cropping of an image takes place after a (possible existing) calibration was applied.

Menu Vision → Drop Calibration:

The current calibration data are dropped, afterwards the shown camera image is uncalibrated

Menu Vision → Teach Fiducial

Here one or more fiducials can be taught that have to be recognised later. This can be done in fiducial teaching dialogue which opens after selecting this menu item.
This dialogue shows the current – probably calibrated – image. There an area can be marked by drawing a rectangle directly into that image. After releasing the left mouse button a fiducial detection is performed within that area. The result of this operation is a list of keypoints which are marked by circles within the image and listed on the dialogues left hand side. Now within that list all these keypoints have to be removed (by pressing the “Delete” key or by using the related button) that do not belong to a fiducial (e.g. because marking of the area within the image was inaccurate or because they are located between desired positions when more than one fiducial is marked). During all that time a fiducial detection is performed with the same image to get a rough recognition quality value. This value is shown on lower left side of the dialogue. It should be as close as possible to 100%. When too much keypoints are removed, this values decreases – and the expected detection accuracy decreases as well.
As soon as this dialogue is left by pressing the “OK”-button, a fiducial is taught and can be used. This fiducial can be saved in a separate file and can be loaded as needed with the following menu items.

Menu Vision → Load Fiducial

This menu can be used to load an already existing, saved fiducial definition. Currently used fiducial(s) is/are replaced by this function.

Menu Vision → Save Fiducial

This menu can be used to save the current fiducial definitions for later usage.

Menu Vision → Drop Fiducial

The current fiducial definitions are dropped, no more fiducial recognition is possible until new fiducial(s) is/are taught or loaded.

As soon as there is a fiducial definition active and valid, it is used for marking operations. When the marking dialogue is opened, a fiducial recognition is performed automatically. When the fiducial(s) can be detected in current live image, the orientation and position of the actual marking project is changed according to the fiducial(s) orientation and position. This procedure is repeated after every marking process automatically. Additionally it can be triggered again in mark dialogue by pressing button “Auto”, it starts a new cycle of recognition and correction.

6.12.2 Camera Calibration Cookbook

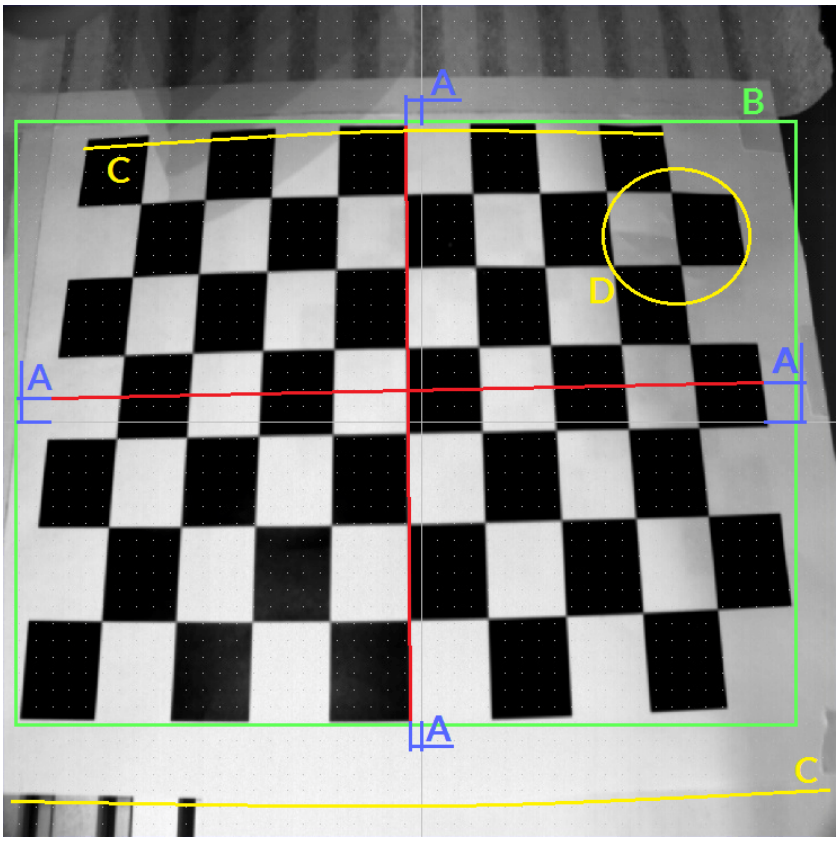
Sometimes the image shown by a camera may be distorted for different reasons. This may be caused e.g. by the optics of the camera or when it is not mounted 100% parallel to the working area. This may result in a somehow stretched and distorted image. Using camera calibration function this can be adjusted. To get an accurate and exact camera calibration is not an easy thing and may require a lot of trial and also some experience with such systems.
To make this easier, here some steps are described which should help to get an accurate calibration:

1. Print out the calibration template from http://halaser.eu/checkerboard.png or – when it does not fit to your visible working area size and aspect ratio – create an own one consisting of black and white squares. For printing an as thick as possible paper should be used so that it can be placed on the working area without being bent, warped or folded. If necessary stick the checker board on a flat sheet of metal or an other stable material.
2. Use a camera with a high resolution. The checker board from the calibration template should be visible very clear.
3. Adjust the camera parameters in BeamConstructs hardware settings. First of all a monochrome image should be used. The vision system does not make use of colours and would convert the image to monochrome representation in every case. So enable greyscale mode in camera plug-in and check the points 1) and 2) again. Next it is recommended to use the native hardware resolution of the camera and to avoid to scale the image to a different size. Also keep the native aspect ratio of

the camera.

4. Place the checker board template in cameras view, it should be located exactly centred, parallel to the borders of the camera, it should fit to the whole visible area completely. Beside of that it should not be bent.
5. Go into camera calibration dialogue, enter the number of visible checker board cross points and try a calibration.
6. Now crossing points all should be marked with a circle. When some crossing points of the checker board do not contain such a circle or when at least one of them does not fit to the real position of the crossing point exactly, the calibration recognition has failed. In this case you should adjust lighting of the calibration template as well as brightness and contrast settings until all crossing points can be detected exactly.
7. Now you can leave the camera calibration dialogue by pressing “OK” and you should be able to see a corrected camera image.

Following image shows some typical mistakes that would lead to a camera correction that damages the resulting image instead of correcting it:



A (blue) – the checker board is not aligned properly, it is rotated and shifted. Here the red lines mark where the crosshair from BeamConstructs working area should be located at really. The calibration template in this image is shifted to the left, to top and rotated to left. Since camera calibration system does not know this is the template only, it will assume this is caused by the camera and will try to compensate it.

B (green) – marks where the visible area of the camera really should end and how the calibration template should cover the cameras visible field. If necessary, an own checker board has to be created that has the required aspect ratio and that contains the required amount of squares. The more squares that are visible, the more exact the calibration becomes.

C (yellow) – here the calibration template is bent, it is not placed flat on and parallel to the working area. Here the camera calibration function will also try to compensate this bent part of the image – which will result in an overcompensated image since the real working area is not bent in the same way.

D (yellow) – the calibration template is folded. The calibration function will try to compensate this damage too

(which is in template only but not a camera-caused distortion).

6.12.3 Fiducial Calibration Cookbook

To get a vision system with a fiducial recognition that is working well and that produces exact results is not an easy thing and requires a lot of adjustment and experience. As a starting point and to get some basic information to make own experiences with, following some hints are given to get better results. The steps listed below should be performed as accurate as possible:

1. Use a camera with a high resolution. The fiducial(s) should be visible very clear and quite large so that all relevant details of it/them can be seen in cameras image.
2. Adjust the lighting. As first step a working piece should be placed in cameras view. Now adjust the light so that the fiducial(s) on that working piece can be seen clear and has/have a good contrast comparing to its underground and comparing to other markers on this working piece. Think about using coloured light which could result in better contrast between fiducial and background.
3. Adjust the camera parameters in BeamConstructs hardware settings. First of all a monochrome image should be used. The vision system does not make use of colours and would convert the image to monochrome representation in every case. So enable grayscale mode in camera plug-in and check the points 1) and 2) again. Next it is recommended to use the native hardware resolution of the camera and to avoid to scale the image to a different size.
4. Calibrate the camera as described above. Please note: in case your camera has a really good lens, does not distort the image and is mounted parallel and on top above the working area, this step should be dropped. The camera calibration always results in a loss of information and quality and therefore should be avoided when possible. Nevertheless: a lens distortion is much more worse than this loss of information, so this should be dropped only in case of a really good camera!
5. Teach the fiducials as described above. It is not possible to make any forecast which of the fiducial recognition keypoints has which influence on the recognition result. Thus several iterations are necessary here: select the fiducial(s) in teach dialogue, remove the unused keypoints, accept the result and try recognition with a changed position of the working piece. And depending on the results: start from the beginning with fiducial selection and keypoint removal.

6.13 Processing Operation

To manipulate a loaded project related to the processing operation several functions can be found in menu “Process” of the main window:

- Simulate – a full process cycle is simulated, means the complete project is updated, all serial numbers, date and time objects are triggered, the process time of geometries is calculated, delays are measured and movements are (partially) simulated. The resulting total process time is displayed to the user. This time is nearly the same time such a process would need within the target environment. It differs only in case motion elements are used that perform absolute movements, here the real processing time can't be calculated because the starting position of an driven axis is not known.
- Mark: this option requires at least a scanner card to be configured within the global project settings and gives the possibility to start a real marking process instead of simulating it. In case the current project uses motion elements the related motion controllers should be configured too within the project settings.
  After selecting this menu item a marking dialogue opens, for more information about its functionalities please refer below.
- Write Stand Alone Data – this function applies only to these scanner controllers that support the related feature; it gives the possibility to write scanner data to a disk for later usage in stand-alone mode
- Send Stand Alone Data – this function applies only to these scanner controllers that support the related feature; it gives the possibility to send unidentifiable scanner data to the controller without starting a mark operation immediately but for later usage in stand-alone mode

- Send Named Stand Alone Data – this function applies only to these scanner controllers that support the related feature; it gives the possibility to send named scanner data to the controller without starting a mark operation immediately but for later usage in stand-alone mode. Comparing to the preceding option this one opens a dialogue where following data can be entered:
  - the name the data are stored with at the scanner controller card and which can be used later to select this specific project in stand-alone mode; the rules this name has to be created with depend on the scanner card
  - an option “Send full data to all cards” which is enabled in multihead mode; when it is set, it can be used to copy the full vector data to all configured controller cards, elsewhere only these data are to every card, which really belong to it according to the multihead marking rules
- Show State – this option shows different scanner controller state information; which information are shown depends on the used scanner card and its capabilities but it may consist of
  - digital output data (reflects the data that have been sent to the output but not necessarily the data that really are set as long as the card does not support readback of these outputs)
  - digital input data
  - analogue input data
  - plug-in information
  - scanhead information
- Increment – all serial number elements of the current project are incremented, the date and time elements are updated
- Decrement – all serial number elements of the current project are decremented, the date and time elements are updated
- Reset – all serial number elements of the current project are reset to its starting value, the date and time elements are updated

6.14 Marking

The mark dialogue can be opened via menu “Process” menu item “Mark”. When this dialogue is opened for the first time or in case the hardware settings have been changed after it was used the last time, as first operation the scanner controller cards and the – optionally configured – motion devices are opened. Motion axes that are configured to reference on start-up are moved to its reference/home position. All these operations may take some time dependent on the connected hardware.

In case this could be completed successfully, the dialogues mark-buttons can be used afterwards. It provides a button for starting the pilot laser (requires fully configured pilot laser outputs in project settings) on the left side of the dialogue, a mark-button in the middle that starts the main laser and a stop-button on the right side that stops the main or pilot laser. Below of them some options can be found that influence the marking operation:

- Mark repeatedly – starts an infinite loop, as soon as the mark-button is pressed the current geometry is marked again and again until the stop-button is pressed
- Fast repeat – this check box becomes active only if a) “Mark repeatedly” is set and b) all used scanner controller cards within the current configuration support this function. When it is set, it gives the possibility to have a very fast loop without any delay between end of previous and begin of current mark cycle. This function buffers several mark-cycles on the scanner controller card in advance and is intended to be used with short projects which have to be marked very fast after each other (e.g. when marking of these projects is started by external triggers following in short time distances after each other).
- Mark selected elements – when this option is set only these parts of a project are marked that are selected within the Entity-Tree and highlighted within the drawing area
- Wait for external trigger – using this option marking is started only under two conditions: first the start-button is pressed, next an external trigger signal is applied. Together with the “mark repeatedly” function this can be used to start marking via an external hardware signal and no longer by pressing the start-button repeatedly.

Additional marking-related functionalities are arranged in separate tab-panes that are grouped according to the functionality they contain. tab-pane “Pilot Laser” contains functions for accessing the preview/pilot-laser

that can be started by the upper left mark button:

- Mode – this combo box gives the possibility to choose the pilot laser marking mode, “Off” disables usage of the pilot laser “Manual” expects that the user starts the pilot laser manually by pressing the big pilot-laser-button on the windows upper left hand side and “Auto” re-starts the pilot laser after every marking process automatically. Also in mode “Auto” the pilot laser has to be started manually for the first time because of security reasons.
- The radio buttons below this combo box can be chosen to select what the pilot laser shall show in its preview, “Use total projects outline” draws a simple rectangle that includes the whole project, “Use individual elements outline” draws a simple rectangle for every element within the current project (except dots, lines and triangles which are shown fully) and “Use real element shapes” draws the full shape of all elements except hatches.
  The pen – and therefore the related speed and delay values – that is used for the pilot laser can be defined within the pen settings dialogue (please refer above).

Tab panel “Manual” offers direct access to the position of scanhead. Here mirrors can be moved to a specific position:

- In upper combobox the scanhead can be chosen this operation has to be performed on (valid only for multihead configurations, elsewhere “Scanhead 1” has to be used
- four arrows give the possibility to move the scanner position in positive and negative X- and Y-direction; the input field between these arrows specifies the jump length every button press has to cause; pressing these buttons lets the scanner jump immediately
- In fields “X” and “Y” coordinates can be selected the scanner has to be moved to, by pressing button “Jump” motion to these positions is started
- The button with the camera symbol in upper right corner of this tab-pane gives the possibility to toggle a live-background image (in case an image capture device is available and it is configured to be shown within the drawing area). On weak computer systems this function can be useful to turn this live view on only as long as it is needed to not to slow down or disturb the main marking process.
- Below of the camera-button there is a small pilot-laser-button which gives the possibility to turn on and off the pilot laser (only available in case a pilot output is configured in hardware settings). Together with the coordinate buttons it can be used e.g. for referencing purposes – when the pilot laser permanently points to a fixed, defined position which also can be changed as needed.

In tab panel Geometry the current projects vector data can be manipulated:

- Arrows left/right/up/down – using these arrows the whole geometry of the project can be shifted into the given direction (in top view for 3D mode); the shift step size in unit mm is taken out of the input field below of these arrows
- Arrows rotation left/right – using these arrows the whole geometry of the project can be rotated into the given direction (in top view for 3D mode); the rotation step size in unit degrees is taken out of the input field below of these arrows
- Button Auto – this button can be used only when a live image is captured and when fiducials are teached using the integrated vision system, it tries to detect the teached fiducials in current image and aligns and rotates the current geometry automatically according to the position and orientation of these fiducials
- Button Reset – when this button is pressed all changes in position and rotation of the current projects geometry are taken back, it is reset to its default values

In tab panel Motion it is possible to drive configured motion axes manually and independent from any motion elements in marking project. Please note: when there are some marking elements used directly within the project, it has to be taken into account that the user may have changed the position of the axes via these manual functions. Thus at least the first motion element should perform a movement to an absolute position or should reference its position. Within the mark dialogue following motion functions are available:

- OUT2 / OUT3 – specifies which of the two possible motion controllers has to be used
- Speed – specifies the speed a manual motion operation can be performed with;
  PLEASE NOTE: when the speed value given here is bigger than the maximum speed of the related

axis, this value is clipped internally, means the axis moves with it's maximum speed which may be slower than the nominal speed entered here

- Camera – this button can be used to toggle the background image on and of in case manual image capture enabling is configured within project settings; this button has the same functionality like the related vision menu item
- movement value – the value that can be entered in middle between the arrows specifies the relative movement distance or the absolute motion position to move to (depending on the arrow button that was pressed)
- Double arrows – these buttons cause a movement as long as the button is pressed
- Single arrows – these buttons cause a single movement via the distance or angle specified with the number input field in the middle, here the time the button is pressed does not influence the motion
- Arrow to dot – when this button is pressed an absolute movement is performed, the number input field in the middle between the buttons specifies the target position to move to, the time the button is pressed does not influence the motion
- Ref – start referencing of the related axis

For planar movement axes the images of the buttons can be changed within the motion axis configuration panel of the general project settings. Depending on the real movement direction (left/right, up/down, in/out) a different symbol can be shown here in order to make it easier for the operator to decide which direction a motion will use. The same is true for the axis names, they can be configured via the axis alias value in general project settings.

The third tab panel Parts is related to parts and part counting, some of its buttons and functionalities will be enabled only when part counting is enabled for the current project (can be done in general project settings dialogue). It offers the following functions:

- When part counting mode is enabled the large counter shows the current number of parts already marked. When the limit specified in project settings or in input field “To produce” is reached, no more marking operations are possible until the user presses button “Reset Counter”
- The “Serial numbers” buttons can be used to change the state of all input elements of the current project for the next part to be marked, here “Increment” increments all of them by one, “Decrement” decrements them by one step and “Reset” sets them back to their start value. This is the same functionality like it is accessible from within menu “Process”.

The tab panel "Progress" shows some information regarding the current marking progress, its functionality is limited to some specific applications. First of all it is feed with timing and operational data only when sliced 3D marking data are processed. Next it will not show the real marking state of the connected controller but only the state of data sent to the scanner card. So when a scanner controller card with a large buffer is used, it may hold some none-processed data while for this progress bar they are already finished. So this progress information is mainly suitable for long-running processes (where relative deviation between mark-dialogue-estimated and real marking time is quite small) but not for getting a process forecast for fast marking operations. For last case it is still recommended to use the function "Simulate" in menu "Process".

6.15 Exporting Data

To use the created data on the target system for processing work pieces they have to be stored in a usable format. Here following possibilities exist:

BeamConstruct “.BEAMP” project files:

This is the project file format of BeamConstruct, here all information about used elements and sub-elements, modified geometry, scanner and laser parameters are contained, no information are lost when this format is used.

CSV format:

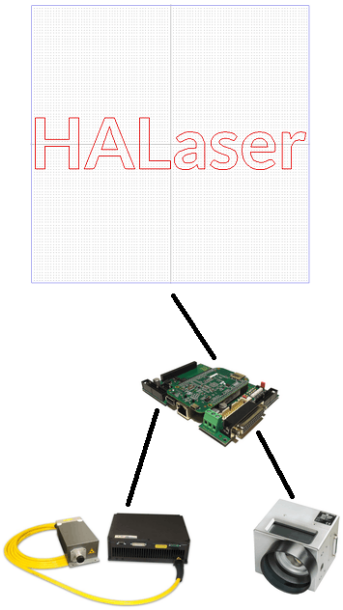
The “comma separated values” format is more a simple list of parameters that are comma-separated. This format includes all geometry and some pen information and can be used e.g. by the CSV to Control plug-in of the ControlRoom software to extract the processing information out of it. Here no dynamic information are contained, only static geometry can be marked when this format is used.

HPGL format:

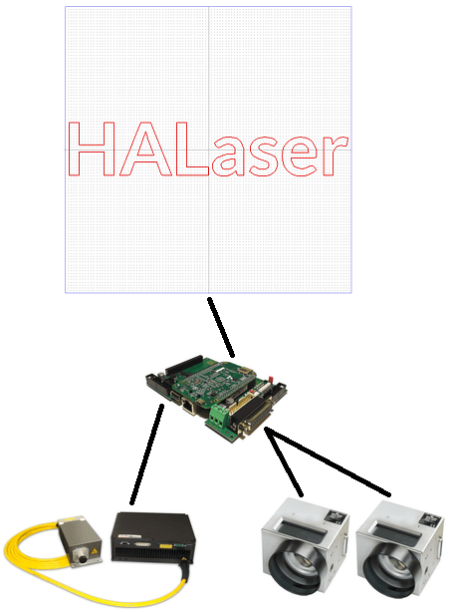
This is a vector graphics format that contains only geometry information, the pen (and thus the processing data) are not part of this format. So it should be used for data exchange with other applications only. This format can be used with the HPGL to Control plug-in. Because of the missing pen information they have to be entered within the plug-ins settings manually. Here no dynamic information are contained, only static geometry can be marked when this format is used.

6.16 Multihead Operations

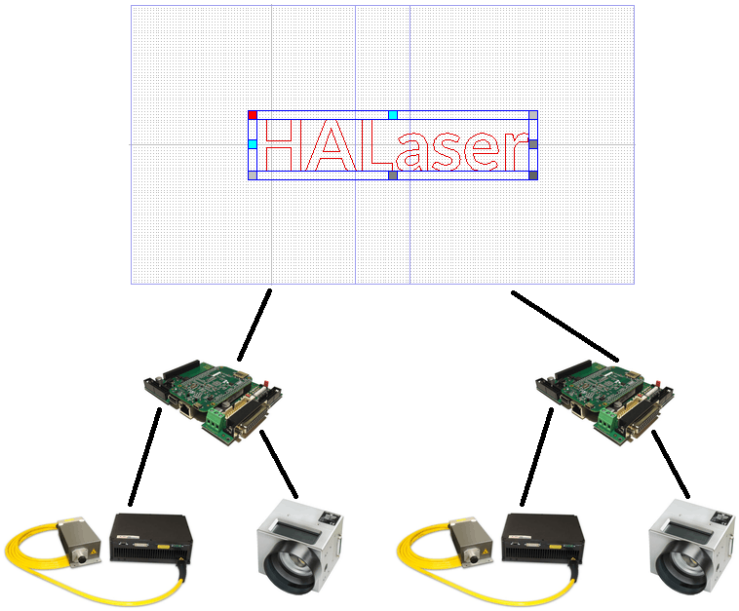
In many cases BeamConstruct will use one scanner card that uses exactly one rectangular working area. In this configuration there will be one scan head performing movements for this working area:



In some special cases this so called singlehead operation mode can make use of a second (and optionally third, fourth,…) working area and a second scan head where both heads perform the same movement. This mode is often called “secondary head”, it simply doubles the output but does not allow to mark different, independent geometries with these two heads:



Full independence of all used heads is possible only with a real multihead mode. In this mode BeamConstruct knows about more working areas that can overlap each other or that can exist beside each other. But all these working areas are completely independent, every area is assigned to an own scan head that performs separate movements:



This multihead mode can be used to:

- mark geometries that are larger than a single working area (in this case it is recommended to let the working areas overlap a bit to profit from vector data optimisations BeamConstruct performs automatically)
- mark large, complex geometries faster (in this case big overlapping areas have to be configured to let both heads work on same area in parallel)
- mark independent geometries faster (in this case working areas do not need to overlap each other)

In last case it is recommended to use scan heads with the same performance and lasers with same power, elsewhere it may happen one marking area slows down the full marking process.

In first two cases different lasers and scan heads can be used but it is recommended to have the more powerful lasers and the faster scan heads configured at the first position.

Setting up of a multihead configuration is quite easy and can be done in general project settings (please refer to detailed description of settings above). There in tab-pane “Hardware” two combo boxes can be found. One is for selecting a scanner controller and one for selecting a scan head. Both can be separate things but don't have to, this depends on the used scanner controller / the used scanner controller plug-in. The difference between both:

- the scanner controller is a piece of hardware / a plug-in that accesses this scanner controller device, where more than one scan head can be connected to.
- the scan head is assigned to a single working area and a single laser that is used for this working area, it is possible to have more than one scan head connected to a scanner controller.

In many cases there will be a 1:1 relation where scanner controller selection is equal to scan head selection, but in case one scanner controller plug-in is able to control more scan heads independently, this separation between “controller” and “head” is necessary.

The same can be found in pen settings (please refer to description above). There the scan head can be selected, means different pen parameters can be set for every working area – independent from the fact which scanner controller plug-in really controls this head.

To configure a multihead-environment, following steps have to be done in global settings of BeamConstruct:

- go to scanner-settings
- in upper combobox select the first scanhead (which is default)
- below of it choose a scanner controller card and configure it, optionally change the field left and top position within the scanner controller card's settings
- select the second scanhead in the combobox on top of the scanner settings tab-pane in global settings
- choose a scanner controller card and configure it → this step now creates a new working area which is assigned to this second scanner controller card and the laser and scanhead it is connected with, so in configuration of this scanner controller the field left and upper size have to be set to appropriate values in order to overlap or to not to be connected with the working field from the first scanner controller card; as soon as a second scanhead/scanner card is configured, BeamConstruct is in mutlihead-mode automatically
- optionally continue this procedure with a third, fourth, fifth,… scanner controller card and related lasers/scanheads/working fields

6.17 IOSelect Mode

Normally a project can be loaded by using the related menu item and by selecting the path and name of the project file manually. This can be automated, using IOSelect mode it is possible to read the state of digital input port of a connected scanner controller and to use the value received from there to automatically load a project file. When this mode ins enabled, marking dialogue is opened and the state of the used digital input port changes, a new project is loaded and can be processed during next marking operation. In case the input state changes when a marking process is still running, loading of the new project is performed as soon as the current marking cycle was finished. To set-up IOSelect mode following steps are necessary:

1. create a subdirectory that will hold all the project files
2. create the project files that have to be loaded, here the filename has to consist of a number followed by the file extension ".beamp". The number has to correspond to the bitpattern detected on digital input. So when bit 0 of an input is set, a file "1.beamp" would be loaded. When all bits of an 8-bit digital input are set, a file "255.beamp" would be loaded. When no bits of a digital input are set, a file "0.beamp" would be loaded - nevertheless it is not recommended to use this file name in order to have a neutral state with no active marking data. All these files have to be placed into directory created in previous step.
3. go to project settings, enable IOSelect-mode in "Misc" panel, specify the directory created above and

select the digital input that has to be used for reading the input state. This step requires a scanner controller card configured properly.

4. Select menu "Process" -> "Mark" to activate automatic IOSelect and to start marking operations

Following things have to be noticed:

- when the digital input provides an input pattern for a file name where no corresponding project file exists, current project is flushed so that there are no marking data available
- loading a project automatically does not overwrite the current hardware settings, so all project parameters found in loaded file are ignored and current settings are kept
- in IOSelect mode marking is not controlled by external trigger automatically, to do that all the numbered projects stored in IOSelect-directory have to start with an External Trigger Primary Element and mark-loop option has to be set in Mark-dialogue
- in multihead-environments currently only card 1 can be used to select a project via digital inputs
- IOSelect-mode is started only after mark-dialogue was opened; during switch to an other project mark-dialogue is closed and re-opens automatically afterwards

6.18 Customise BeamConstruct

BeamConstruct can be modified in a way so that it can be delivered together with machinery or additional equipment using an own brand. Here following attributes can be changed:

- the applications name, this value replaces the name BeamConstruct completely
- the homepage which is shown in “About”-dialogue
- the splash image shown during application start-up and in “About”-dialogue
- the application icon shown in taskbar and main window (not supported on all operating systems)
- the copyright as shown in “About”-dialogue; this custom copyright does not replace the default one but is added to it
- the images shown in toolbar and entity-tree
- the menu items which are visible to the end user

To customise the application, different things can be done, which are optional each:

A plain text file with the name `beam.oem` has to be created in UTF-8 format. To become active, it has to be placed in installation directory of the application for Windows operating systems (typically `C:\Program Files\OpenAPC`) or in `/usr/local/openapc` for Linux systems. Within this file every single line contains one customisation definition:

1. the name of the application
2. the homepage URL to be shown in “About”-dialogue
3. path to the splash image in PNG format
4. path to the icon image in PNG format
5. copyright text to be shown in “About”-dialogue

As an example:

```
Beamy Raymanipulator
http://www.beamconstruct.com
beamoem.png
beamoem_icon.png
Copyright (c) 2012 by LaserFreak
```

This file (in UTF-8 encoding) placed within the directory described above renames BeamConstruct to “Beamy Raymanipulator”, shows http://www.beamconstruct.com within the “About”-dialouge (instead of https://halaser.eu) as well as the additional copyright information. As splash image for the startup-screen the image out of file `beamoem.png` is used. When this files name is given without an absolute path, the application tries to find it in installation directory (Windows) or in directory `/usr/local/openapc` (Linux). The same is true for `beamoem_icon.png` which is used as image for the applications icon.

The images that are shown within toolbar and entity tree can be replaced by own ones too. Here two different methods have to be used to replace all of them.

To replace the toolbar- and entity-tree images, alternative images in PNG-format have to be put into the plug-ins location but with different names. The base name is always the name of the plug-in without extension. For the toolbar image a “1.png” has to be added to this base name, for the entity-tree image a “2.png” has to be added. As an example: The images of the barcode plug-in have to be replaced. It can be found in folder “priplugins” with the file name “libio_pri_barcode.dll” or “libio_pri_barcode.so”. Here two images “libio_pri_barcode1.png” with a size of 23x23 pixels and “libio_pri_barcode2.png” with a size of 15x15 pixels have to be put into the same folder. After next software start the barcode appears with the new images.

To replace all other images PNG files with special names have to be placed in folder “icons”:

- “newprj.png” for the “New”-symbol (23x23 pixels)
- “openprj.png” for the “Load project”-symbol (23x23 pixels)
- “saveprj.png” for the “Save project”-symbol (23x23 pixels)
- “saveprjas.png” for the “Save project as”-symbol (23x23 pixels)
- “quit.png” for the “Quit application”-symbol (23x23 pixels)
- “zoomin.png” for the “Zoom in”-symbol (23x23 pixels)
- “zoomout.png” for the “Zoom out”-symbol (23x23 pixels)
- “zoomworkarea.png” for the “Zoom full working area”-symbol (23x23 pixels)
- “zoomselected.png” for the “Zoom selected entity”-symbol (23x23 pixels)
- “pointdrag.png” for the “Point drag mode”-symbol (23x23 pixels)
- “openmarkdlg.png” for the symbol to open the mark dialogue (23x23 pixels)
- “dlgmark_pilot.png” for the pilot laser symbol in mark dialogue (113x113 pixels)
- “dlgmark_mark.png” for the mark-start symbol in mark dialogue (113x113 pixels)
- “dlgmark_stop.png” for the stop button symbol in mark dialogue (113x113 pixels)
- “penwizard.png” for the pen parameter wizard button in pen settings dialogue (55x55 pixels)
- “rotate_move_left_xpm.png”, “rotate_step_left_xpm.png”, “rotate_step_right_xpm.png”, “rotate_move_right_xpm.png” and “rotate_to.png” for rotational movement buttons in “Motion” tab-pane and mark dialogue (17x17 pixels)
- “arrow_move_down_xpm.png”, “arrow_step_down_xpm.png”, “arrow_step_up_xpm.png”, “arrow_move_up_xpm.png” and “vertical_to.png” for planar, vertically oriented movement buttons in “Motion” tab-pane and mark dialogue (17x17 pixels)
- “arrow_move_out_xpm.png”, “arrow_step_out_xpm.png”, “arrow_step_in_xpm.png”, “arrow_move_in_xpm.png” and “longit_to.png” for planar, longitudinally oriented movement buttons in “Motion” tab-pane and mark dialogue (17x17 pixels)
- “arrow_move_left_xpm.png”, “arrow_step_left_xpm.png”, “arrow_step_right_xpm.png”, “arrow_move_right_xpm.png” and “horiz_to.png” for planar, horizontally oriented movement buttons in “Motion” tab-pane and mark dialogue (17x17 pixels)

To hide specific menu items from BeamConstruct main window, a plain text file with the name `beammenu.oem` has to be created. To become active, it has to be placed in installation directory of the application for Windows operating systems (typically `C:\Program Files\OpenAPC`) or in `/usr/local/openapc` for Linux systems. This text file contains one number for each menu item which has to be disabled. Every number has to be placed in a new line. Following numbers exist and disable the menu items as described below (lists are ordered in appearance of menu items).

For menu “Project”:

- 1001 – New project
- 1002 – Open project
- 1013 – Insert Project…
- 1014 – Save hardware settings…
- 1015 – Save payload data…
- 1016 – Load hardware settings…
- 1017 – Load payload data…
- 1003 – Import…
- 1004 – Save project
- 1005 – Save project as…
- 1010 – Save with options…
- 1101 – Export CSV
- 1102 – Export HPGL
- 1103 – Export CLI
- 1006 – Project settings…
- 1007 – Pen settings…
- 1008 – Save as default configuration
- 1009 – Load default configuration
- 1018 – Machine settings...

For menu “Edit”:

- 2101 – Group
- 2102 – Group to Active Split
- 2103 – Group to Active Move
- 2104 – UnGroup
- 2105 – Duplicate
- 2106 – Align left
- 2107 – Centre horizontally
- 2108 – Align right
- 2109 – Align top
- 2110 – Centre vertically
- 2111 – Align bottom
- 2201 – Merge
- 2202 – Split
- 2203 – Optimise
- 2204 – Reduce Geometry
- 2205 – Extrude
- 2206 – Delete Element(s)
- 2308 – Start Video
- 2301 – Freeze Video
- 2302 – Calibrate Camera
- 2309 – Crop Camera
- 2307 – Drop Calibration
- 2303 – Teach Fiducial
- 2304 – Load Fiducial
- 2305 – Save Fiducial
- 2306 – Drop Fiducial
- 2001 – Undo
- 2002 – Redo
- 2901 – the complete “Vision” submenu including “False colours”
- 2902 – the complete “False colours” submenu

For menu “Process”:

- 3001 – Simulate
- 3002 – Mark
- 3005 – Write Stand Alone Data
- 3009 – Send Stand Alone Data
- 3010 – Send Named Stand Alone Data
- 3013 – Show State
- 3006 – Increment
- 3007 – Decrement
- 3008 – Reset

For menu “Help”:

- 4002 – License
- 4003 – Reset License
- 4005 – Check Dongle
- 4006 – Generate bugreport
- 4004 – Credits

As an example: a file `beammenu.oem` with following contents:

```
1013
2102
2103
3001
```

would hide the following menu items from the main window so that they can't be used any longer:

- menu “Project”, menu item “Insert project...”
- menu “Edit”, menu items “Group to Active Split” and “Group to Active Move”
- menu “Process”, menu item “Simulate”

6.19 BeamLock

BeamLock is a tool that can be used to apply some own lock-rules to a system. This might be useful e.g. for machine integrators that ship hardware with installed BeamConstruct to customers before the full invoice is paid. In this case a lock can be placed on such a system that becomes active after a given time. This means the machines functionality is time-limited. Now the user would have to enter a password that is provided by the vendor of the machine only when invoice is paid/licensing conditions are met/agreements are redeemed. Without the password the software installation is no longer usable, also deinstalling and installing BeamConstruct again will not solve this issue.

BeamLock provides the possibility to define several time ranges with own passwords for each of these time ranges. Thus it is possible to define several lock grades that become active after different time ranges. Additionally a master password has to be defined which can be used to override the time-dependent locks at once and to disable the full lock immediately. Beside of that it is also possible to dynamically create a password which incorporates a time range and gives the possibility to expand a given lock-time again and again without the need to use the predefined lock-ranges.

PLEASE NOTE: all passwords including the master passwords have to be completely different in order to have a good security. Beside of that it is recommended to use complex passwords with a length of at least 8 characters and with upper- and lowercase characters that itself do not form meaningful words. Examples for BAD PASSWORDS are: "mylock1", "mylock2", "mylock2" and "mylockm". Examples for good passwords are: "XdlfTr238", "yERf12rt67", "934trAwX4", "m4RT56ZZtgd34".

6.19.1 Fixed time range locks

To apply different locks with several fixed time ranges on a system, following steps have to be performed:

1. Start BeamLock on the system that has to be locked. For Windows BeamLock.exe can be found in OpenAPC installation directory, for Linux it has to be started out of a console by entering command "BeamLock". On none of the supported systems BeamLock comes with an icon in start menu.
2. Set the "Enable" checkbox for every row where a time range and a password has to be provided; specify the time after the lock has to become active and a password for unlocking it.
3. Specify a master password.
4. Press the button "Lock System". When a system is already locked you have to specify the current master password in order to override this lock. When a system was not locked before but contains a fresh and clean installation, it may be the software asks for a password too: in this case use "HALaser" as master password. When "HALaser" (without the quotation signs) does not work for you, your system was already locked and you cannot overwrite the current lock without knowing the master password that was used during this lock. In such a case you have to contact the vendor of the system, there is no way to remove such a lock when the master password is not known.
5. Write down your passwords and keep them secure. PLEASE NOTE: these passwords and especially the master password are the ONLY way to access your system after specified time has elapsed or to set a new lock! Once again: There is no way to remove such a lock when the master password is not known, so keep these passwords secure and do not press the "Close" or "Cancel" button before you got them!

PLEASE NOTE: For Windows locking procedure it depends on the privileges BeamConstruct is executed with by default. When it is running using an Administrator account the steps described above have to be done using BeamLock.exe together with the same Administrator account. When it is executed using a non-privileged user account the steps described above in every case have to be done twice using BeamLock.exe: once using this users account and once using a fully privileged Administrator account! When doing these steps two times, it is recommended to use the same passwords in both cases to avoid confusion when end user plays around with the used installation and does changes in environment BeamConstruct is used.

PLEASE NOTE: For Linux it is assumed BeamConstruct is not executed as root but as a non-privileged user. In this case the steps described above in every case have to be done twice using BeamLock: once using this users account and once using the root-account! When doing these steps two times, it is recommended to use the same passwords in both cases to avoid confusion when end user plays around with the used installation and does changes in environment BeamConstruct is used.

6.19.2 Dynamic time range lock

Beside the possibility to define one or more fixed time ranges before shipping of a system, it is also possible to generate new passwords which extend the lock time again and again. When such a password is sent to a user, a current lock can be removed for the time which is defined by this password. For creating such a dynamic time range lock, following conditions have to be met:

- the target system needs to be locked by the steps described in previous section and the time period of such a lock needs to have elapsed but the system is not yet unlocked by using one of the predefined passwords
- the master password of this system is known

Comparing to what is described in previous section, such a lock can be created also when the target system is no longer available:

1. Start BeamLock on any available system. For Windows BeamLock.exe can be found in OpenAPC installation directory, for Linux it has to be started out of a console by entering command "BeamLock". On none of the supported systems BeamLock comes with an icon in start menu.
2. Enter the master-password in related input field and confirm it by pressing “Enter”
3. In lower row of input fields right below of the master password, specify the time a system has to be unlocked for and press button “Create”

4. Now a dialogue opens where the unlock password which belongs to this master password and the given time is shown and where it can be copied out of the text field directly.

6.20 BeamServer Remote Control Interface

6.20.1 Overview

BeamConstruct provides an operation mode where whole application can be remote-controlled. This can be used to do some script-or program-controlled automation or to hide BeamConstruct while (visually) performing some other tasks in foreground. To dothis, a special program is provided: BeamServer. When it is started (BeamServer.exe under Windows, BeamServer when using Linux) instead of BeamConstruct, a TCP server socket is created using port 11350. Now an external application can connect to this socket and send some specific ASCII-commands. These commands then invoke usage of BeamConstruct itself. To make that clear: BeamConstruct does not need to be started in this case, this will be done by BeamServer automatically when related commands are sent.

6.20.2 Usage

A standard sequence of commands sent to BeamServer always looks like this:

1. Start BeamServer with a number as command line parameter specifying which parts of the UI have to be shown
2. 2. Create a TCP-connection to the IP of the computer where OpenAPC software package is installed and where BeamServer is running at, connection has to be established to port 11350
3. and following: send several ASCII-commands as described below to control BeamConstruct

finally: Send

ExitUI

to shut down and close BeamConstruct and BeamServer

When TCP connection to BeamServer is interrupted somewhere within this sequence, BeamConstruct is NOT shut down. After re-connecting to BeamServer a controlling application can continue at the point where it has been before interruption. Beside of that only one connection to BeamServer is allowed at the same time.

For testing purposes it is also possible to connect with a normal Telnet-client to BeamServer and to give commands manually. In this case Telnet client has to be used in RAW operational mode (auto-negotiation deactivated).

6.20.3 Starting BeamServer

BeamServer expects a command line parameter specifying which parts of BeamConstruct UI have to be shown. This parameter is a number. When this number is set to 0, BeamConstruct acts completely invisible. When it is set to 255, full GUI is shown. Between these numbers it is possible to define some subsets of the GUI to be shown just by adding the following values and giving the result as parameter:

1 – show drawing area (the big area in the middle where vector data are drawn)

2 – show toolbars (on top of main window where some basic functions and elements are accessible)

4 – show left hand side element panels

8 – show right hand side element tree

16 – show the menu bar with all menu items

32 – show the close button in frame of main window

64 – show the status bar at bottom of main window

131072 – disables the possibility for the user to exit the application; when this value is enabled, the application can be left via command "ExitUI" only

262144 – disable all possibilities for a user to edit projects or geometries, when this value is set, even dragging, rotating or scaling of elements is no longer possible

524288 – the warning-dialogue on start-up is disabled; this value should be used only very rarely and only when user is already informed what the software does so that there can't be any surprising and therefore dangerous operations

1048576 – applies only to mark-dialogue, when this value is used, the manual scanner control tab-pane in mark-dialogue is not shown

2097152 - applies only to mark-dialogue, this hides the part counter tab-pane

4194304 – applies only to mark-dialogue, it hides the "Motion" tab-pane

8388608 – applies only to mark-dialogue, when this value is set, the "Geometry" tab-pane in mark-dialogue is hidden

16777216 – applies only to mark-dialogue, this hides the pilot laser tab-pane

33554432 – applies only when value 2 (see above) is used and when value 1073741824 (see below) is not set, this option hides the "edit pen" button in "Element" tab pane

67108864 – this centers and locks drawing area (in case it is visible by setting the value 1) so that its positions can't be changed by dragging the mouse; the view can be changed only by sending commands that modify the current zoom

134217728 – applies only when value 2 (see above) is used, it inhibits the possibility to edit single points; when this value is used, the coordinates grid in "Geometry" tab-pane is disabled; it is not necessary to set this value when 262144 (see above) is already set

1073741824 – applies only when value 2 (see above) is used, if it is set, the "Element" panel is not shown

536870912 – applies only when value 2 (see above) is used, this hides the dynamically changing panel with the parameters of the currently selected element

1073741824 – applies only when value 2 (see above) is used, this disables and hides the layers-panel

2147483648 – do not load and use locally stored default settings, the software starts with an (empty) default configuration when this initialisation value is used

As an example: "BeamServer.exe 17" would show BeamConstruct consisting of drawing area (=1) and menu bar (=16) only

For Windows please note: BeamServer.exe has to be started using OpenAPC installation directory as working directory. To ensure this directory is used, it is recommended to create a shortcut to BeamServer.exe out of the OpenAPC installation directory, this configures the correct directory automatically. Additionally it is possible to set the parameter within the shortcut definitions too. When BeamServer.exe is started using any other directory, it will not work propery and will not be able to use any of the elements and plug-ins.

6.20.4 Remote Control Commands

Every command that is sent to BeamServer via TCP/IP is responded either by a text "OK" when operation was successful or by a text "ERROR" signalising the last command could not be executed successfully. "ERROR" is always followed by two error codes where the first one describes the kind of the error that happened and the second one is an internal error number that may be helpful in case of problems of the BeamServer itself. For first error code following values are possible:

3 – OAPC_ERROR_DEVICE – tried to access an external device (scanner controller card, motor controller,…) which was configured for usage but which was not accessible

7 – OAPC_ERROR_RESOURCE – tried to access a resource that does not exist (e.g. when using UID or name of an element that could not be found)

11 – OAPC_ERROR_NO_MEMORY – not enough memory available

16 – OAPC_ERROR_INVALID_INPUT – the data/parameters given within last command are incomplete or wrong

18 – OAPC_ERROR_CREATE_FILE_FAILED – creation of a new file/appending data to an existing file failed, file could not be opened

19 – OAPC_ERROR_OPEN_FILE_FAILED – opening a file for reading failed (happens e.g. when the file does not exist or is not accessible for some reason)

20 – OAPC_ERROR_WRITE_FILE_FAILED – writing data into an already opened file failed (e.g. because there is no more space left on a device)

21 – OAPC_ERROR_READ_FILE_FAILED – reading data from an already opened file failed either because the file is damaged or truncated or because the file format is not as expected

Following commands are supported by BeamServer:

ExitUI

After sending this command BeamConstruct is stopped and - if visible - the GUI of it is closed. The BeamServer and it's TCP/IP socket is shut down as well. Afterwards it is not possible to send any other command.

CreateLog <path>

Creates a (new) logfile at the given path. This logfile will contain all commands sent to the BeamServer, all responses on these commands sent back to the client which is connected to the BeamServer and all notes generated with command WriteLogNote. When this command is called and a logfile is already open and used, the old one is closed and log data are written to the new file. When the file specified by parameter path already exists, the contents of this logfile are deleted and a new file is created.
Every line within the logfile starts with a fixed character to identify the type of log entry:

C: - command sent from client to BeamServer

R: - response to a command, sent from BeamServer back to client

N: - log note as set by command WriteLogNote

WriteLogNote <text>

Writes the given text into a logfile. To use this command, a log has to be created by calling CreateLog before. The text specified as parameter to this command is written to the log file without any further modifications.

CmdStopMark

Tries to stop a running marking operation in the same way like it would be done by a user pressing the "STOP"-button in mark dialogue.
This command is intended to be used with mark-operations which work non-blocking, means where the command returns immediately and not only after operation has finished. For all blocking commands it would not make sense to send CmdStopMark since the BeamServer would not be able to receive and to process

this command.

CmdIsMarking

Checks if an asynchronous marking operation is still pending/a mark dialogue is still open. This function returns 0 when no mark operation is pending and 1 when marking is active.

TriggerUI <number>

This command can be used to invoke any action that would be possible by user interaction within BeamConstruct too. The given <number> specifies which operation has to be triggered, here every user action like selecting a menu item or pressing a toolbar is possible. Most of the commands return immediately, independent if the operation was successful or not. Synchronous commands that return only when related operation was finished are mentioned explicitly below. Currently following <number>-values are possible:

All four-digit numbers correspond to menu items and/or tool bar buttons and their functionality within the main application, when such a number is used the related operation is invoked (for a more detailed description please refer to BeamConstruct documentation above):

1001 – new project
1002 – load existing project file
1003 – import custom file format (as supported by BeamConstruct: vector and raster image formats)
1004 – save the current project using a known name
1005 – save the current project using a new name
1101 – export vector data in CSV format
1102 – export vector data in HPGL/PLT format
1103 – export layer vector data in CLI format
1006 – open general project settings dialogue
1007 – open pen settings dialogue
1008 – save the current configuration as default settings
1009 – load default settings
1010 – save project with options
1011 – import 3D custom file format (as supported by BeamConstruct: vector formats to be used as 3D mesh)
1012 – open general settings dialogue right as code 1006 but brings the “Scanner” tab pane to front
1013 – insert a new .BEAMP project file without deleting already existing project data
1014 – save hardware settings into a .BEAMH file
1015 – save geometries, control elements and pen definitions into a .BEAMV file
1016 – load hardware settings from a .BEAMH file and leave geometries, control elements and pen definitions unchanged
1017 – load geometries, control elements and pen definitions from a .BEAMV file and leave current hardware settings unchanged
1018 – open general machine settings dialogue
1019 – import custom file format (as supported by BeamConstruct: vector, 3D and raster image formats) in case of a 3D mesh add the data to an already existing Active Slice Group instead of creating a new, separate one
1020 – import custom 3D file format (as supported by BeamConstruct: 3D meshes) and add it to an already existing Active Slice Group instead of creating a new, separate one
1021 – import custom bitmap image file format and open the related import dialogue to let the user set import parameters

2001 – undo last operation
2002 – redo last un-done operation
2101 – concatenate all selected elements into one group
2102 – concatenate all selected elements into a splitgroup
2103 – concatenate all selected elements into a movegroup
2104 – ungroup the elements out of the selected group

2105 – duplicate the selected element
2201 – merge the geometries of the selected element into a static one
2202 – split the geometries of the selected element logically
2203 – optimise the geometries of the selected element
2204 – reduce the geometries of the selected element using a lossy algorithm
2205 – extrude the selected element into third dimension
2206 – delete the currently selected element(s)
2301 – freeze the currently shown background video
2302 – calibrate the currently connected camera
2303 – teach fiducials
2304 – load fiducials
2305 – save teached fiducial(s)
2306 – drop teached fiducial(s)
2307 – drop camera calibration data
2308 – toggle image capturing on or of (when configured to be started manually in global project settings)
2309 – configure cropping of camera image

3001 – start process simulation
3002 – opens the marking dialogue to let the user control the marking process
3003 – marks the current project once, here the mark dialogue is opened to let the user stop the marking process if necessary; the dialogue is closed automatically as soon as marking was finished; the return information "OK" for this command is sent only after marking process has finished, thus it is not necessary to check the marking state, as soon as a response is received, marking is complete (or was stopped)
3004 – marks the current project silently and without opening the mark dialogue; the return information "OK" for this command is sent only after marking process has finished, thus it is not necessary to check the marking state, as soon as a response is received, marking is complete (or was stopped)
3006 – increment the elements in current project
3007 – decrement the elements in current project
3008 – reset the elements in current project to their default/start values
3013 – open devices when not already done and show scanner card state dialogue
3015 – marks the selected elements out of the current project once, here the mark dialogue is opened to let the user stop the marking process if necessary; the dialogue is closed automatically as soon as marking was finished
3020 – this command starts marking of the current project in a loop while opening a mark dialogue which consists of a stop-button only. This mark dialogue and mark-loop can be left by pressing the stop-button, by clicking the close-symbol of the mark-dialogue itself or by sending command `CmdStopMark`.
3021 – start the pilot laser with the mark dialogue visible, pilot laser mode and number of loops can be set with commands `CmdSetPilotMode` and `CmdSetPilotLoop`
3022 – start the pilot laser without a visible mark dialogue, pilot laser mode and number of loops can be set with commands `CmdSetPilotMode` and `CmdSetPilotLoop`
3023 – this command is similar to command 3020, it starts marking of the current project in a loop but it does not show any mark dialogue where the operation could be stopped by the user. This mark dialogue and mark-loop can be left only by sending command `CmdStopMark`. As long as this looped mark operation is running, the user interface is locked and no manual operations are possible with BeamConstruct.

4001 – show the "about"-dialogue
4002 – show license information
4003 – reset the current license
4004 – show the "credits"-dialogue
4006 – generate a bugreport

5001 – switch the current drawing area view to (default) top view
5002 – switch the current drawing area view to front view
5003 – switch the current drawing area view to right side view
5004 – switch the current drawing area view to 3D view
5005 – switch the current drawing area view to split mode with all views combined

All five-digit numbers correspond to remaining tool bar buttons and their functionality within the main

application, please refer to section „BeamConstruct“ above to get detailled information about them:

10001 – zoom into view
10002 – zoom out of view
10003 – zoom to view full working area
10004 – zoom to fully view selected element
10005 – enable pointdrag mode for selected element
10006 – log in a user

CmdNewPrj

Clears all current data and flushes all existing elements, settings and pens. After calling this command default settings are restored and project does not contain any geometries. When `CmdAppendPrj` is used afterwards to load a new project, current (default) settings are kept.

CmdLoadPrj <path>

Loads a new .BEAMP project from given `<path>`. The elements contained in the given project are appended to the current ones (if there are any). The current settings and pens are overwritten by the ones found in the loaded project file.

CmdSavePrj <path>

Saves the current project into a .BEAMP project file as specified by `<path>`.

CmdLoadPenPrj <path>

Loads the pens out of a .BEAMP project from given `<path>`. This project has to contain only pen definitions, no vector data and no hardware settings. When this command is used, all current pens are removed, the pen definitions out of this file are loaded and when it contains less pens than the original project, the missing pens are filled up using the default pen settings out of the loaded file. So this function can be used to replace pen definitions without loading a completely new project.

CmdSavePenPrj <path>

Saves the pens out of the current project into a .BEAMP project at the given `<path>`. This project then will contain only pen definitions, but no vector data and no hardware settings.

CmdAppendPrj <path>

Loads a new .BEAMP project from given `<path>`. The elements contained in the given project are appended to the current ones (if there are any). The current settings and pens are kept, the ones in loaded project file are ignored.

CmdGetPrjName

Returns the (file) name of the currently loaded project. When no project is loaded, an empty line is returned.

CmdGetPrjPath

Returns the full path name of the currently loaded project. When no project is loaded, an empty line is returned.

CmdGetEstMarkTime

Returns the estimated marking time of the current project in unit milliseconds. When no project is loaded, the

returned marking time will be 0.

CmdImport <path>

Imports some vector data that do not have standard .BEAMP project file format. The imported element then gets a name which consists of path separator and file name. As an example: when a file is imported from `C:\tmp\test.svg` the imported element will get the name `\test.svg`

CmdImportSilent <path>

Imports some vector data that do not have standard .BEAMP project file format. Comparing to `CmdImport` here importing is done silently and without any configuration dialogues opening during this operation. Instead of this data are imported using default parameter. The imported element then gets a name which consists of path separator and file name. As an example: when a file is imported from `C:\tmp\test.svg` the imported element will get the name `\test.svg`

CmdSetCharIO2 <new text>
CmdSetCharIO3 <new text>
CmdSetCharIO4 <new text>
CmdSetCharIO5 <new text>

Set a new ASCII text to be used for all input elements that make use of the logical "Char" inputs. When such an input element is assigned to a text or barcode primary element, the text or barcode is replaced by the given `<new text>` for the next marking operation. The four commands for char IO 2..5 correspond to the inputs 2..5 that can be selected in used input elements. The `<new text>` has to be given in plain 8 bit ASCII (local default encoding). For changing texts in Unicode-encoding, please refer to functions CmdSetUniIOx below.

CmdSetUniIO2 <new text>
CmdSetUniIO3 <new text>
CmdSetUniIO4 <new text>
CmdSetUniIO5 <new text>

Set a new Unicode text to be used for all input elements that make use of the logical "Char" inputs. When such an input element is assigned to a text or barcode primary element, the text or barcode is replaced by the given `<new text>` for the next marking operation. The four commands for char IO 2..5 correspond to the inputs 2..5 that can be selected in used input elements. The `<new text>` has to be given in UTF-8 encoding. For changing texts using plain ASCII-encoding, please refer to functions CmdSetCharIOx above.

CmdLaserOn <pen>

This command turns on the laser for the currently selected head immediately and without dependence of any vector data. The parameter handed over together with this command specifies a pen which has to be used for controlling the laser. To use this command, the scanner controller device needs to be opened. This can be done with command `CmdOpenDevs`. To select a head, command `CmdSelHead` can be used.

CmdLaserOff

This command turns off the laser for the currently selected head immediately and without dependence of any vector data. When the laser has been turned on before by calling `CmdLaserOn`, it is mandatory to use this command before the device is closed with `CmdCloseDevs`. Elsewhere the state of the laser after closing the device is undefined. To select a head, command `CmdSelHead` can be used.

CmdGetCardInpVal <inputflag>

Read and return the status of a selected input of the card that belongs to the currently selected head. When

this card does not have such an input or when an illegal value is given for parameter `inputflag`, an error is returned. The parameter itself can have exactly one of these values which specifies the input to be read:

- `IOCTRL_ANALOGUE_8_1` – 4
- `IOCTRL_ANALOGUE_8_2` – 8
- `IOCTRL_ANALOGUE_10_1` – 16
- `IOCTRL_ANALOGUE_10_2` – 32
- `IOCTRL_ANALOGUE_10_3` – 64
- `IOCTRL_ANALOGUE_10_4` – 128
- `IOCTRL_ANALOGUE_10_5` – 256
- `IOCTRL_ANALOGUE_10_6` – 512
- `IOCTRL_ANALOGUE_12_1` – 1024
- `IOCTRL_ANALOGUE_12_2` – 2048
- `IOCTRL_ANALOGUE_12_3` – 4096
- `IOCTRL_DIGITAL_8_1` – 8192
- `IOCTRL_DIGITAL_8_2` – 16384
- `IOCTRL_DIGITAL_16_1` – 32768
- `IOCTRL_DIGITAL_16_2` – 65536
- `IOCTRL_DIGITAL_32` – 131072
- `IOCTRL_ANALOGUE_16` – 262144
- `IOCTRL_ANALOGUE_12_4` – 524288
- `IOCTRL_ANALOGUE_16_1` – 1048576
- `IOCTRL_ANALOGUE_16_2` – 2097152

To select the head and therefore the card this command has to be used with, prior to calling this command `CmdSelHead` can be used.

`CmdSetCardOutpVal`

This command has to be used before `CmdSetCardOutput`, it is used to set a value which has to be sent to one of the outputs of a connected controller.

`CmdSetCardOutpMask`

This command can be used before `CmdSetCardOutput`, it is used to set a mask-value which has to be sent to one of the outputs of a connected controller.

`CmdSetCardOutput`

Using this command a value is sent to an output of the currently connected scanner controller card. This value has to be handed over by a preceding call to `CmdSetCardOutpVal` and mask is whith an optional value set by a call to `CmdSetCardOutpMask`. It makes use of the card of the currently selected head (refer to `CmdSelHead`) and a single `IOCTL_`-flag which is handed over as parameter. When no mask is set or the mask is set to 0xFFFFFFFF, then the full value is handed over to the controller. When a mask is set, only these bits of the value are changed at the output, which are specified by a “1” (=bit set) in the mask value.

The flag which has to be handed over specifies the output where the value has to be set to, here it depends

on the currently used card which one is available and usable. The flags are the ones defined in header file oapc_libio.h. Following are supported (exclusively, it is not allowed to OR-concatenate them):

- IOCTRL_LASERPORT_8_1 – 1
- IOCTRL_LASERPORT_8_2 – 2
- IOCTRL_ANALOGUE_8_1 – 4
- IOCTRL_ANALOGUE_8_2 – 8
- IOCTRL_ANALOGUE_10_1 – 16
- IOCTRL_ANALOGUE_10_2 – 32
- IOCTRL_ANALOGUE_10_3 – 64
- IOCTRL_ANALOGUE_10_4 – 128
- IOCTRL_ANALOGUE_10_5 – 256
- IOCTRL_ANALOGUE_10_6 – 512
- IOCTRL_ANALOGUE_12_1 – 1024
- IOCTRL_ANALOGUE_12_2 – 2048
- IOCTRL_ANALOGUE_12_3 – 4096
- IOCTRL_DIGITAL_8_1 – 8192
- IOCTRL_DIGITAL_8_2 – 16384
- IOCTRL_DIGITAL_16_1 – 32768
- IOCTRL_DIGITAL_16_2 – 65536
- IOCTRL_DIGITAL_32 – 131072
- IOCTRL_ANALOGUE_16 – 262144
- IOCTRL_ANALOGUE_12_4 – 524288
- IOCTRL_ANALOGUE_16_1 – 1048576
- IOCTRL_ANALOGUE_16_2 – 2097152

When an invalid flag is handed over, the function call fails with error OAPC_ERROR_INVALID_INPUT immediately.

CmdSendMark

Using this command it is possible to send the current project data as stand-alone marking data to a connected scanner card. This requires a configuration where a scanner controller is set up properly, that supports this feature (like Scanlab RTC4 ScanAlone). Here no additional parameters are required since this function sends one exclusive set of marking data to the card.

CmdWriteMark <path>

Using this command it is possible to write the current project data as stand-alone marking data into a local file. This requires a configuration where a scanner controller is set up properly, that supports this feature (like HALaser E1701 or HALaser E1803 controller series). The given path specifies the full path of the file that has to be used for the stand-alone data. When the file already exists, it will be overwritten without further notice. Such a file can be transferred to the scanner controller manually in next step (e.g. by storing it onto the microSD card of E1701 and then putting this card back into the controller).

CmdSendNamedMark <name>

Using this command it is possible to send the current project data as stand-alone marking data to a connected scanner card using a given name. This requires a configuration where a scanner controller is set up properly, that supports this feature (like HALaser E1701 or HALaser E1803 controller series). The given <name> is an identifier the marking data are stored on the card with. Here the naming structure has to be according to the regulations of the used scanner card. For E1701 controller this means the name needs to be something like "0:/filename.epr" where "0:/" specifies microSD-card as storage place and ".epr" is the recommended file name extension for such stand-alone data. When a file with this name already exists on the controller, it will be overwritten without further notice.

CmdSelEntName <name of element>

Logically selects an element (but not highlights it in view). After an element with the given name was selected, it can be processed using other commands (as described below). In case the whole project was selected using `CmdSelPrj`, this selection is no longer valid.

CmdSelEnt <id>

Logically selects an element (but not highlights it in view). After an element with the given identifier was selected, it can be processed using other commands (as described below). In case the whole project was selected using `CmdSelPrj`, this selection is no longer valid.

CmdSelPrj

Logically selects the whole project including all elements (but does not highlight them in view). Afterwards this project and all of the contained elements can be processed using other commands (as described below). In case a single element was selected using `CmdSelEnt` or `CmdSelEntName`, this selection is no longer valid.

CmdSetSelEnt <0/1>

Select or deselect a selected element visibly in view. This operation is similar to a left-click on an element by the user while shift-key is held down. An additional parameter 0 or 1 decides if the element has to be selected or deselected. The element itself which is modified by this operation can be chosen by a call to function `CmdSelEntName` or `CmdSelEnt` which internally selects the element for further operations. After setting an element visibly selected by this command, all UI-operations can be performed that belong to selected elements. To fully zoom into a single element following sequence of commands can be used:

```
CmdSelEntName elementname
CmdSetSelEnt 1
CmdZoomElem
```

To zoom to two elements, this sequence of commands can be used:

```
CmdSelEntName elementname1
CmdSetSelEnt 1
CmdSelEntName elementname2
CmdSetSelEnt 1
CmdZoomElem
```

To unselect a previously selected element, parameter "0" has to be used

```
CmdSelEntName elementname1
CmdSetSelEnt 0
```

CmdZoomElem

Zooms view to fully show all selected elements. The element(s) to be shown after this command has/have to

be selected by calling CmdSelEntName/CmdSelEnt and CmdSetSelEnt before.

CmdDeleteEnt <id>

Deletes an element completely. When the parameter id is set, the element with this identifier is deleted. When no parameter id is given, the currently selected element is deleted: This element has to be selected by calling CmdSelEntName or CmdSelEnt before. To make the element disappear from current view, a call to CmdRedraw may be necessary.

An example about how an element with the name “Rectangle 5” can be deleted:

```
CmdSelEntName Rectangle 5
CmdDeleteEnt
```

An example about how an element with the unique identifier 18 can be deleted:

```
CmdDeleteEnt 18
```

CmdDeleteAllEnts

Deletes all elements within the current project. Different to CmdNewPrj this command only removes the geometries but leaves hardware settings, pens and all possibly opened devices unchanged.

CmdSetLoopRepeat <repeats>

Changes the amount of repeats which have to be done for this element. A value of 0 means, the element is marked once and without repeating this operation. The element to be modified has to be selected by calling CmdSelEntName or CmdSelEnt before.

CmdSetPosX <position>

In case an element was selected by calling CmdSelEntName or CmdSelEnt before:

This command changes the X-position of the current element (left corner) to the value given as parameter in unit 1/1000 mm.

In case the whole project was selected by calling CmdSelPrj before:

This command translates the X-position of all elements of the current project by the given value in unit 1/1000 mm. Here “translation” means, the position value given with the command is added to the current position of all elements of the project, their positions are not overwritten by it.

To make the changes visible which have been applied by this command, a call to CmdRedraw may be necessary.

CmdSetPosY <position>

In case an element was selected by calling CmdSelEntName or CmdSelEnt before:

This command changes the Y-position of the current element (top corner) to the value given as parameter in unit 1/1000 mm.

In case the whole project was selected by calling CmdSelPrj before:

This command translates the Y-position of all elements of the current project by the given value in unit 1/1000 mm. Here “translation” means, the position value given with the command is added to the current position of all elements of the project, their positions are not overwritten by it.

To make the changes visible which have been applied by this command, a call to CmdRedraw may be necessary.

CmdSetPosZ <position>

In case an element was selected by calling `CmdSelEntName` or `CmdSelEnt` before:

This command changes the Z-position of the current element to the value given as parameter in unit 1/1000 mm.

In case the whole project was selected by calling `CmdSelPrj` before:

This command translates the Z-position of all elements of the current project by the given value in unit 1/1000 mm. Here “translation” means, the position value given with the command is added to the current position of all elements of the project, their positions are not overwritten by it.

To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

CmdGetPosX

Gets the X-position (horizontal) of the current element and returns it using unit 1/10000 mm. The related element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetPosY

Gets the Y-position (vertical) of the current element and returns it using unit 1/10000 mm. The related element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetPosZ

Gets the Z-position (depth) of the current element and returns it using unit 1/10000 mm. The related element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdSetScaleX <factor*1000>

Changes the width of the current element by the factor given as parameter. The element to be modified has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

CmdSetScaleY <factor*1000>

Changes the height of the current element by the factor given as parameter. The element to be modified has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

CmdSetScaleZ <factor*1000>

Changes the depth of the current element by the factor given as parameter. The element to be modified has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetScaleX

Returns the scale factor which is applied to the width of the current element in unit *1000. The element the scale factor has to be retrieved for needs to be selected by calling `CmdSelEntName` or `CmdSelEnt`.

CmdGetScaleY

Returns the scale factor which is applied to the height of the current element in unit *1000. The element the scale factor has to be retrieved for needs to be selected by calling `CmdSelEntName` or `CmdSelEnt`.

CmdGetScaleZ

Returns the scale factor which is applied to the depth of the current element in unit *1000. The element the scale factor has to be retrieved for needs to be selected by calling `CmdSelEntName` or `CmdSelEnt`.

CmdGetSizeX

Gets the X-size (width) of the current element and returns it using unit 1/10000 mm. The related element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetSizeY

Gets the Y-size (height) of the current element and returns it using unit 1/10000 mm. The related element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetSizeZ

Gets the Z-size (depth) of the current element and returns it using unit 1/10000 mm. The related element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetOutline

Gets the complete outline (bounding box) of the current element and returns it using six values in unit 1/10000 mm. The returned values are separated by spaces and provide position X, position Y, position Z, size X, size Y and size Z consecutively in one single line. The element the outline has to be retrieved for needs to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

This is a convenience function which incorporates the functionality of commands `CmdGetPosX`, `CmdGetPosY`, `CmdGetPosZ`, `CmdGetSizeX`, `CmdGetSizeY` and `CmdGetSizeZ`.

CmdSetRotX <angle>

Changes the rotation around X-axis of the current element to the value given as parameter in unit 1/1000 degrees. The element to be modified has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

CmdSetRotY <angle>

Changes the rotation around Y-axis of the current element to the value given as parameter in unit 1/1000 degrees. The element to be modified has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

CmdSetRotZ <angle>

Changes the rotation around Z-axis of the current element to the value given as parameter in unit 1/1000 degrees. The element to be modified has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

CmdSetHatchAngle <angle>

This function sets a hatch angle parameter in unit 1/1000 degrees. The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatchDist <distance>

This function sets a hatch distance parameter in unit 1/1000 mm. The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatchOffs <offset>

This function sets a hatch lines offset parameter in unit 1/1000 mm. The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatchPen <pen-number>

This function sets a hatch pen. The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatchStyle <style-id>

This function specifies the hatch style to be used. It can be one of the following text values:

- “fwd” – separate lines, forwarded
- “fwdbwd” – separate lines forward and backward
- “connected” – connected lines forward and backward
- “zigzag” – connected zigzag lines
- “inner” – inner lines repeating the shape of the element-specific

The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatchLinemode <mode-id>

This function specifies the mode of the hatch lines to be used. It corresponds with the “linelength” parameter to be set with function `CmdSetHatchLinelen` and can be one of the following text values:

- “continuous” – hatch lines consist of single lines with one start and endpoint each, parameter “linelength” is ignored in this case
- “lines” – hatch lines consist of short lines combined to one longer, continuous line, parameter “linelength” specifies the length of the short line segments
- “dashed” – hatch lines are dashed lines, parameter “linelength” specifies the length of the lines and the distances between them
- “dotted” – hatch lines consist of single dots, parameter “linelength” specifies the distance between them
- “cust” – hatch lines consist of custom dots as specified in current projects settings, parameter “linelength” specifies the distance between the centre point of each of these custom dots

The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatchLinelen <length>

This function sets the length of a hatch line which is used when a custom line mode is used (please refer to function `CmdSetHatchLinemode` above for details). The value is set internally but not yet applied to any element. To do this, an additional call to `CmdSetHatch` is necessary.

CmdSetHatch <id>

When this command is executed, the previously set hatch parameter are applied to an element. This element has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. To make the changes visible which have been applied by this command, a call to `CmdRedraw` may be necessary.

When no parameter `id` is given, a new hatch element is added to the selected element. The ID of the new element is returned. When the identifier of an existing hatch element is handed over as parameter `id`, this hatch element is modified and the new parameters are applied to it.

The returned value is the unique identifier for the hatch element and can be used with commands like `CmdDeleteEnt`, `CmdSelEnt` or `CmdSetHatch`.

An example about how to set a new hatch with 0,5 mm hatch distance, 45 degrees hatch angle and pen number 2 to an existing element “Circle 1”:

```
CmdSetHatchDist 500
CmdSetHatchAngle 45000
CmdSetHatchPen 2
CmdSelEntName Circle 1
CmdSetHatch
```

An example about how to modify an existing hatch element with the identifier 8 to use 0,1 mm hatch distance, 180 degrees hatch angle and pen number 3:

```
CmdSetHatchDist 100
CmdSetHatchAngle 180000
CmdSetHatchPen 3
CmdSetHatch 8
```

Two examples about how to remove an existing hatch element with the identifier 8 from a parent element:

```
CmdSelEnt 8
CmdDeleteEnt
```

or

```
CmdDeleteEnt 8
```

CmdRefresh

Refresh the current view. This updates the shown geometries without recalculating and redrawing everything new

CmdRedraw

Redraw the current view. This performs a full re-calculation and re-drawing of all geometries and therefore is slower than `CmdRefresh`. This command should be called only when vector data of elements of the current project have been changed and the changes are not yet visible.

CmdListName <num>

This command expects a positive number `num` as parameter and can be used to list all available elements with their name. To get all elements, it has to be called repeatedly with increasing numbers (starting from 0). For every call with a different number the related element name will be given. When there are no more elements, an error is returned.

CmdListUID <num>

This command expects a positive number `num` as parameter and can be used to list the UIDs of all available elements. To get all elements UIDs, this command has to be called repeatedly with increasing numbers (starting from 0). For every call with a different number the related element UID will be returned. When there are no more elements, an error is returned.

CmdSetElemFile <path>

Changes the file that is assigned to an element and replaces the related data. This command can be used e.g. with Scanner Bitmap Elements to replace the current image.

The `path` to the new file has to be given as parameter and needs to point to an existing file in appropriate file format.

The element where the data are replaces by this command has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdSetElemFlag <name>

Sets a flag to an element that was selected by calling `CmdSelEntName` or `CmdSelEnt` before. The flag itself is specified by the given `name` and is activated by this function. The name of the flag and it's functionality depends on the type of element that is selected. For an overview of available flags please refer to section below.

CmdResetElemFlag <name>

Resets a flag of an element that was selected by calling `CmdSelEntName` or `CmdSelEnt` before. The flag itself is specified by the given `name` and is deactivated by this function.

The name of the flag and it's functionality depends on the type of element that is selected. For an overview of available flags please refer to section below.

CmdSetElemText <text>

Changes the text of a barcode or text primary element to the value given as parameter. The barcode or text element this new value is set for, has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before. In case the GUI is visible, the element typically does not change its appearance immediately after this command is set. Although the internal data already have changed after this call, an additional call `CmdRefresh` is necessary to update the visual representation.

CmdGetPen

Returns the pen-number that is assigned to a selected element. The element where this command retrieves the pen-number from, has to be selected by calling `CmdSelEntName` or `CmdSelEnt` before.

CmdGetPenFreq <pen-number>

Returns the frequency in unit Hz that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenPower <pen-number>

Returns the power in unit 1/1000% that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenColour <pen-number>

Returns the RGB-colour that is assigned to the pen with the given number, to the head selected with command `CmdSelHead` (or head 0 by default) and which is used for drawing mark lines. Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenOffColour <pen-number>

Returns the RGB-colour that is assigned to the pen with the given number, to the head selected with command `CmdSelHead` (or head 0 by default) and which is used for drawing jump lines when the related option is enabled. Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenJSpeed <pen-number>

Returns the jump speed in unit µm/sec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenMSpeed <pen-number>

Returns the mark speed in unit µm/sec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenLOffDelay <pen-number>

Returns the laser-off-delay in unit µsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenLOnDelay <pen-number>

Returns the laser-on-delay in unit µsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenJDelay <pen-number>

Returns the jump-delay in unit µsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenMDelay <pen-number>

Returns the mark-delay in unit µsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenPDelay <pen-number>

Returns the in-polygon-delay in unit µsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenSpotsize <pen-number>

Returns the lasers spotsize in unit 1/10 µm that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenWobbleAmp <pen-number>

Returns the wobble amplitude in unit µm that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenWobbleFreq <pen-number>

Returns the wobble frequency in unit 1/1000 Hz that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenPLength <pen-number>

Returns the pulse length in unit µsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenFPK <pen-number>

Returns the first pulse killer pulse length in unit nsec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenStdByPLength <pen-number>

Returns the stand-by pulse length in unit usec that is assigned to the pen with the given number and to the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdGetPenValues <pen-number>

Returns a list of all pen-parameters in the order of the single `CmdGetPenXXX`-commands listed above using the given pen number and the head which can be selected with command `CmdSelHead`. Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenFreq <pen-number> <frequency>

Sets the `frequency` (in unit Hz) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenPower <pen-number> <power>

Sets the `power` (in unit 1/1000%) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenColour <pen-number> <colour>

Sets the RGB-`colour` which is used for drawing mark lines to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenOffColour <pen-number> <colour>

Sets the RGB-`colour` which is used for drawing jump lines to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenJSpeed <pen-number> <speed>

Sets the jump `speed` (in unit µm/sec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenMSpeed <pen-number> <speed>

Sets the mark `speed` (in unit µm/sec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenLOffDelay <pen-number> <delay>

Sets the laser-off-`delay` (in unit µsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenLOnDelay <pen-number> <delay>

Sets the laser-on-`delay` (in unit µsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenJDelay <pen-number> <delay>

Sets the jump-`delay` (in unit µsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenMDelay <pen-number> <delay>

Sets the mark-`delay` (in unit µsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenPDelay <pen-number> <delay>

Sets the in-polygon-`delay` (in unit µsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenSpotsize <pen-number> <spotsize>

Sets the lasers `spotsize` (in unit 1/10 µm) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenWobbleAmp <pen-number> <amp>

Sets the wobble amplitude (in unit µm) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenWobbleFreq <pen-number> <frequency>

Sets the wobble frequency (in unit 1/1000 Hz) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenPLength <pen-number> <length>

Sets the pulse `length` (in unit µsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenFPK <pen-number> <length>

Sets the first pulse killer pulse (FPK) `length` (in unit nsec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSetPenStdByPLength <pen-number> <length>

Sets the stand-by pulse `length` (in unit usec) to the pen with the given number and the head selected with command `CmdSelHead` (or head 0 by default). Which pen is assigned to a specific element can be evaluated by calling function `CmdGetPen`.

CmdSelHead <head-number>

Sets the number of a head which has to be used for a subsequent command. Here head-number is a 0-based value that specifies a special head in multihead-environments. When this function is never called, all commands that are head-dependent, use head number 0 by default.

CmdSetWAX <position>

Set the X-position of the current working area using unit µm. The head this new value has to be applied for needs to be specified by a preceding call of `CmdSelHead`.

CmdSetWAY <position>

Set the Y-position of the current working area using unit µm. The head this new value has to be applied for needs to be specified by a preceding call of `CmdSelHead`.

CmdSetWAZ <position>

Set the Z-position of the current working area using unit µm. The head this new value has to be applied for needs to be specified by a preceding call of `CmdSelHead`.

CmdSetWAWidth <size>

Set the width (=horizontal size) of the current working area using unit µm. The head this new value has to be applied for needs to be specified by a preceding call of `CmdSelHead`.

CmdSetWAHeight <size>

Set the height (=vertical size) of the current working area using unit µm. The head this new value has to be applied for needs to be specified by a preceding call of `CmdSelHead`.

CmdSetWADepth <size>

Set the depth (=Z direction) of the current working area using unit µm. The head this new value has to be applied for needs to be specified by a preceding call of `CmdSelHead`.

CmdSetEAX <position>

Set the X-position of the global editing area using unit µm.

CmdSetEAY <position>

Set the Y-position of the global editing area using unit µm.

CmdSetEAZ <position>

Set the Z-position of the global editing area using unit µm.

CmdSetEAWidth <size>

Set the width (=horizontal size) of the global editing area using unit µm.

CmdSetEAHeight <size>

Set the height (=vertical size) of the global editing area using unit µm.

CmdSetEADepth <size>

This function returns the number of elements which exist in current project. Here not only primary elements are counted but every group, every groups sub-element(s), every hatch and every input element is counted as an own, separate one

CmdGetElemNum

Set the depth (=Z direction) of the global editing area using unit µm.

CmdSetPilotMode <mode>

This command corresponds to `TriggerUI 3021` and `TriggerUI 3022` and can be used to set the mode the pilot laser is working with. Following values are allowed to be set for `mode`:

- `PrjOutline` – let the pilot laser show the whole projects outline
- `ElemOutline` – let the pilot laser show every elements outline
- `ElemShape` – let the pilot laser show every elements real shape (without hatching)

CmdSetPilotLoop <cnt>

This command corresponds to `TriggerUI 3021` and `TriggerUI 3021` and can be used to set the number of loops the pilot laser has to perform at once. Here for `cnt` every value greater than 0 can be set.

CmdCaptureView <width> <height> <path>

Captures the drawing area and saves it as image into a file. Here `width` and `height` specify the size of the image to be saved. When positive values for both, `width` and `height` are specified, the image is scaled to that size. When one of both values is -1, the image is scaled using the value given for the other one by keeping the aspect ratio for that side that is set to -1. When both values are -1, the view is saved in it's original resolution.

`path` is the full path where the image has to be saved at. The file extension used for the name of the file in this path decides what format has to be used for saving. Supported file formats are .png, .bmp, .gif and .jpeg (while JPEG is not recommended to be used for unscaled images).

Example:

```
CmdCaptureView -1 -1 F:/viewcapture.png
```

will save the view in PNG format using it's original size on drive F: using file name viewcapture.png.

6.20.4.1 Element Flags

This section describes the flags that are available for the different element types that can be set or reset with commands like `CmdSetElemFlag` and `CmdResetElemFlag`.

6.20.4.1.1 Scanner Bitmap Element Flags

`ditherFS`, `bw`, `greyscale` – these flags are special, they can only be set and not reset and only one of them can be set. Means when a flag `bw` is set, the two other flags `ditherFS` and `greyscale` are automatically reset. These flags decide the format of the image which can be either a Floyd-Steinberg-dithered black-and-white image, a plain black-and-white image or a greyscale image

`mark_bidir` – this flag has an influence on behaviour during marking and enables a mode where lines are marked from beginning to end and from end to beginning alternating. This saves some marking time since the jump back to the beginning of the next line is saved.

`mark_fromlastline` – this flag has an influence on behaviour during marking and lets the image be processed from bottom to top when it is set

`mark_wo_lineincr` – this flag has an influence on behaviour during marking and marks all lines of an image at the same y-position

`mark_inverted` – this flag has an influence on behaviour during marking and inverts the power values resulting in an image that is negative to the visual representation; depending on the used material this finally may result in a positive image

`aspect_fixed_x`, `aspect_fixed_y` – these flags have no counterpart in BeamConstruct's UI and can be set or reset via BeamServer only; they are used when an image of an existing Scanner Bitmap Element is replaced by command `CmdSetElemFile`. When none of these flags is set, the new image is scaled to the same size like the previous image which may result in a loss of the current aspect ratio. When one of both flags is set, the aspect ratio of the new image is kept while one of the two sides X or Y uses the same width or height like the original image. When both flags are set, behaviour is undefined

6.20.5 Compatibility commands

To give the possibility to migrate existing software fast and easy, some non-standard compatibility commands are available. These commands behave like the ones in their original API, for details about their usage please refer to the related vendors manual. When developing new BeamServer applications these functions should not be used but the native remote control commands described above.

Following compatibility commands are supported at the moment:

- `ScCciChangeTextByName()`
- `ScCciIsMarking()`
- `ScCciLoadJob()`
- `ScCciMarkEntityByName()`

6.21 Smart Interface

The interface described here is read-only. It can be used to implement Smart Factory (Industry 4.0) applications and provides different status informations which give the possibility to react on events during marking/production.

When only this interface is used, it is possible to watch processing of one or more BeamConstruct-controlled machines without the need to have an eye on them all the time. As soon as something important happens which requires user interaction, a notification is sent via the Smart Interface. This notification can be received by a suitable device (e.g. a Smartphone-application) which then informs the user that something has to be done at a specific machine. Now the operator can check that machine.

In combination with the BeamServer remote control interface or when programming interface of libbeamconstruct is used out of an own application, this interface gives extended possibilities to automate production with BeamConstruct according to the concept of smart factories. Now not only detailed state information are available but it is possible to react on them automatically. So e.g. when marking came to an end the application can decide if it starts the next marking operation or if it is necessary to have some operator interaction in order to continue production.

For more details about the BeamConstruct programming interface, please refer to the information about BeamSDK at https://halaser.eu/feature.php#BeamSDK.

6.21.1 Accessing the Smart Interface

The smart interface data are provided via an unencrypted, unauthenticated TCP/IP socket using port number 11355. BeamConstruct sends several UTF-8-encoded XML documents via this interface. These documents provide state information but the interface does not accept any data sent by the connected client. So independent from what (unauthorized) client connects here, it is not possible to cause any dangerous operation via this connection.

This interface accepts an unlimited number of incoming connections. But to avoid an unnecessary high load on a running instance of BeamConstruct, it is recommended to not to connect with too much clients to this interface. For security reasons it is highly recommended to not to make this interface available via the Internet but only within a protected, walled-off production network.

The data provided at the Smart Interface all come in XML format and all have one common basic structure:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg Time="2017-08-17T15:17:31" ID="901B0E5A06A2">
 ...
 </BCMsg>
```

The first line is mandatory and specifies the beginning of a new XML document according to the XML specification. The second line starts the root-node `BCMsg` of every Smart Interface message. This root-node can contain two optional attributes `Time` specifying the time the message was sent (ISO 8601 time format, local time) and `ID` uniquely identifying the Machine/software installation this message was sent from.

Dependent on the kind of message which is sent, it may be extended by some more sub-nodes and attributes (symbolised by "..." in the example above).

Following the messages are described which are supported by the Smart Interface. PLEASE NOTE: although these messages are described as separate XML structures, they all can be combined within one `BCMsg`-node:

Alive Message

This message is sent latest 40 seconds after last transmission in order to signal connected clients the software is still alive. When no messages have been received by a client for a longer time (e.g. for more than 150 seconds), the client should close the connection and try to reconnect. This alive-message does not provide any further information and therefore comes with an empty `BCMsg`-node:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg/>
```

Status Message

This message is sent whenever the operational state of the application changes. It comes with a sub-node `Data` which contains an attribute `State`:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg>
  <Data State="<state>"/>
 </BCMsg>
```

Here `<state>` can have following values:

`idle` – software is started but marking dialogue is not open and no mark operation is running

`ready` – mark dialogue is open but no marking operation is active yet

`marking` – a marking process is active

`paused` – a sliced 3D project was halted by pressing the “Pause”-button in mark dialogue; when this project is continued later and no error happens, it switches back to state “`marking`”

`error` – an error occurred, no marking is possible at the moment; in case this state is used, an additional Attribute `Text` provides the error text related to the current error (using the language the software is running with)

Project Message

This message specifies which project is loaded at the moment. It comes with a sub-node `Data` which contains an attribute `Project`:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg>
  <Data Project="<pathname>"/>
 </BCMsg>
```

Here <pathname> is the full path to the project file which just was loaded. When it is empty, the "New" function was used and the last project was flushed completely.

Number-Of-Parts Message

This message returns information about the current part number which has been produced and corresponds to the parts-parameter in global settings. When a <parts>-value of 0 is sent, the parts-counter was reset by the operator. Additionally there can be some optional attributes ID which uniquely identifies the current working piece and ProdID which uniquely identifies the product which is currently process. These identifiers typically are set externally, e.g. when BeamConstruct is integrated into an automated production line where previous machines provide these data via a suitable communication interface (like a Hermes connection):

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg>
  <Data Parts="<parts>" ID="<id>" ProdID="<prodid>">
   <Trace Name="<name>"/>
   <Trace Name="<name>" ID="<identifier>"/>
  </Data>
 </BCMsg>
```

The <Data>-section may encapsulate one or more nodes of type <Trace> when the “Trace data” option is enabled for elements in the current project. Every <Trace> section contains an attribute Name which keeps the name of the element in project which has been processed successfully by the last marking operation. For input elements an additional attribute ID is contained which holds the input data which have been processed. So this functionality can be used e.g. for traceability purposes. Since the Number-of-Parts message is emitted only in case a marking cycle was performed successfully, did not fail due to an error and was not interrupted by a stop-operation, the data contained in this message can be used for further automated processing of the produced working piece. Together with an input element that provides unique data, it can be identified uniquely. Please note: the information provided here only inform about the fact a marking cycle has be performed successfully. Since this software can't check if e.g. the working piece really was in place and was properly aligned, it does not necessarily mean the marking result was as expected, so additional inspection may be necessary with the working piece in order to ensure the correct result.

Number-Of-Slice Message

This message is sent when a sliced 3D SLS/SLM model is built. Here the slices are announced as they are submitted to the scanner controller card. So this does not necessarily mean the current slice specified by value <slice> is finished, but it was handed over to the scanner card for production. The second attribute MaxSlices provides information about the maximum number of slices this model consists of:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg>
```

```
  <Data MaxSlices="<maxslices>" CurrSlice="<slice>"/>
 </BCMsg>
```

Process data Message

This message is sent when a process control plug-in is used and when this plug-in sends measured process data. These process data are provided via this interface:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg>
  <Process Name="<name>" Unit="<unit>" Idx="<index>" Value="<value>"/>
</BCMsg>
```

The data are encapsulated in a `Process`-tag which can exist several times within a XML package and which contains the following attributes:

`Name` – the name of the measured `<value>`

`Unit` – the measurement unit the `<value>` is given with

`Idx` – an index value in range 0..3; a plug in can measure up to four different values, this index specifies which of these values is provided

`Value` – the measured `<value>` which is further specified by the other attributes

Notification message

Using this message, generic information can be submitted to a client. Such a notification message contains some text which is useful to the user and which can have different priorities:

```
<?xml version="1.0" encoding="UTF-8"?>
 <BCMsg>
  <Msg Src="<sender>" Text="<shorttext>" Prio="<prio_number>"          PrioText="<prio_text>"
Desc="<detailtext>"/>
</BCMsg>
```

Different to the preceding ones this message provides different attributes within a `Msg`-tag:

- `Src` specifies the sender of this notification message, here either a unique machine identifier is provided (in case it is available) or a generic description
- `Text` is a short variant of the notification text, it acts as some kind of headline to give an overview what the notification is about
- `Desc` is the complete, detailed notification message
- `Prio` is a number and `PrioText` is a text corresponding to the number, both specify the priority this notification has:
  - 1 = “Fatal error”
  - 2 = “Error”

- 3 = “Warning”
- 4 = “Info”
- all values >4 = “Other”

6.22 MQTT Data Interface

The interface described here is one-way only. The application connects to a MQTT broker (as configured in “6.7 Machine Configuration“) and sends state information to this broker which then can be used by other clients which are connected to the same MQTT data broker.

When only this interface is used, it is possible to watch processing of one or more BeamConstruct-controlled machines without the need to have an eye on them all the time. As soon as something important happens which requires user interaction, a notification is sent via the MQTT Data Interface. This notification can be analysed by a suitable client application which then informs the user that something has to be done at a specific machine. Now the operator can check that machine.

In combination with the BeamServer remote control interface or when programming interface of libbeamconstruct is used out of an own application, this interface gives extended possibilities to automate production with BeamConstruct according to the concept of smart factories. Now not only detailed state information are available but it is possible to react on them automatically. So e.g. when marking came to an end the application can decide if it starts the next marking operation or if it is necessary to have some operator interaction in order to continue production.

For more details about the BeamConstruct programming interface, please refer to the information about BeamSDK at https://halaser.eu/feature.php#BeamSDK.

For more details about the MQTT protocol, please refer to https://mqtt.org

6.22.1 MQTT Messages

BeamConstruct sends several MQTT-messages to a connected broker automatically when a related event occurs. These messages provide state information but the interface does not accept any data sent by the broker BeamConstruct has connected to. So independent to what (unauthorized) client the software may connect to, it is not possible to cause any dangerous operation via this connection.

The data provided via the MQTT data interface are structured according to the MQTT specification and contain information as described below. Here the field “Topic” always contains a part `<basetopic>`, this is the topic as configured in “6.7 Machine Configuration“. Unless otherwise noted all data come as plain ASCII-texts.

Status Message

This message is sent whenever the operational state of the application changes:

Topic: `<basetopic>/state/`

Data: `IDLE` – software is started but marking dialogue is not open and no mark operation is running

`READY` – mark dialogue is open but no marking operation is active yet

`MARKING` – a marking process is active

`PAUSED` – a sliced 3D project was halted by pressing the “Pause”-button in mark dialogue; when this project is continued later and no error happens, it switches back to state “`MARKING`”

`ERROR` – an error occurred, no marking is possible at the moment; in case this state is used, an additional error-message is submitted

Error Message

This message is sent whenever the operational state of the application changes to or from “ERROR”:

Topic: <basetopic>/error/

Data: an error text in case a previous status message signalled an error state or an empty string in case no error happened or in case a previous error was reset

Project Message

This message specifies which project is loaded at the moment:

Topic: <basetopic>/project/

Data: the full path to the project file which just was loaded. When it is empty, the "New" function was used and the last project was flushed completely.

Number-Of-Parts Message

This message returns information about the current part number which has been produced and corresponds to the parts-parameter in global settings. When a value of 0 is sent, the parts-counter was reset by the operator:

Topic: <basetopic>/parts/

Data: the number of parts (as human readable ASCII-text)

Number-Of-Slice Message

This message is sent when a sliced 3D SLS/SLM model is built. Here the slices are announced as they are submitted to the scanner controller card. So this does not necessarily mean the current slice is finished, but it was handed over to the scanner card for production:

Topic: <basetopic>/slices/

Data: <curr_slice>/<max_slices> - the slice information (as human readable ASCII-text) where <curr_slice> is the slice currently processed/sent to the controller and where <max_slices> is the total number of slices to be processed in current project.

7 CorrCorrect

This section describes CorrCorrect, a tool to create and modify scanhead correction tables.

7.1 Overview

When a scanhead – a system with two mirrors that deflect a laser in X and Y direction – is used, the resulting movement on the working area can be distorted. Here a few major reasons for distortions can be found:

1. a pincushion distortion that is caused by the mirrors and the used optics
2. a 3D focal distortion that happens because the distance between last mirror and working area differs dependent on the position where the laser hits it – the distance is shorter in the middle directly below of the mirror and longer on the borders of the working area, resulting in a loss of focus for some of the positions
3. a spatial correction because the scanhead is not mounted in parallel to working area because it has to work inside of a concave working piece

To correct such distortions special correction tables are used. In most cases these correction tables are specific to and delivered together with a scanhead. They use a vendor-specific file format that has to be converted depending on the used software/the used scanner controller plug-in. In other cases no such correction file exist and has to be generated from scratch. As a third case for some very high accuracy applications existing correction files have to be modified so that they fit to the used hardware exactly.

All these three tasks can be done with CorrCorrect correction file tool that is described here. The resulting correction table afterwards can be saved in BeamConstruct HD (.bco) correction file format – a special format that provides much better accuracy than most other formats. Such a correction file can be used for some of the scanner controller plug-ins of OpenAPC package directly (like the ETH6608 plug-in).

7.2 Usage

To start with generation of a new correction file the “New”-toolbar button or the menu Project → New has to be selected. This opens a dialogue where parameters for focal and pincushion distortion can be set. Additionally it can be specified which of both corrections have to be applied to the correction table. So for plain 2D applications or hardware set-ups where no third axis or no possibility to change the focus exists, only pincushion distortion parameters can be set. For a real 3D set-up focal correction parameters can be applied too.

Alternatively it is possible to load an already existing correction file via menu Project → Open or by using toolbar button “Load Project”.

Next this correction table can be modified by specifying one or more nominal and actual positions. To get the values required for this, the current correction file has to be used for marking. During the marking operation one or more single points have to be marked. These points should be located at positions where the accuracy has to be increased or at positions spread over the whole working area. After marking the positions it has to be measured where the points have been marked really. The resulting values have to be entered in single or multiple point correction dialogue that can be opened via menu Edit → Correct Single Point or menu Edit → Correct Multiple Points. After entering the nominal coordinate(s) – the coordinates of the points within BeamConstruct – and the actual coordinate(s) – the real coordinates as measured after marking – the current correction table is modified at the given position(s) in order to reflect the deviation(s) at this/these point(s). This operation has to be repeated with the same or varied point coordinates until the resulting correction table offers the required accuracy.
Please note: these correction operations influence the correction table only in X- and Y-direction. A possibly existing focal correction is damaged by these operations. Because of that for full 3D correction the focal correction has to be applied again after this: By selecting menu Edit → Recalc Focal Correction the related correction is re-calculated according to the changed positions of 2D correction coordinates.

Additionally the CorrCorrect tool provides the same “Geometry”-panel like it is known from BeamConstruct. So when the current correction table – that is shown in the middle of the application – is selected, here

several operations can be performed. It can be scaled, mirrored, centred, rotated, the total position can be changed and other things more.

7.3 Correction Definition Dialogue

When menu Project → New or toolbar button “New Project” is selected, a dialogue opens, where focal and pincushion distortion parameters can be set. There exists an other dialogue that provides only focal correction parameters and that can be selected via menu Edit → Recalc Focal Correction. This dialogues parameters are a subset of the parameters of the correction definition dialogue and therefore are not described separately.

This dialogue provides following general parameters:

- Z-Distance – the shortest distance between last mirror and working area

The correction dialogue provides following parameters for pincushion distortion:

- Checkbox Pincushion Distortion – the 2D / pincushion distortion parameters are applied to the newly generated correction table only when this checkbox is set
- X-angle – the distortion strength/angle in X direction
- Y-angle – the distortion strength/angle in Y direction

This dialogue provides following parameters for focal correction:

- Checkbox Focal Correction – the depth / focal distortion parameters are applied to the newly generated correction table only when this checkbox is set
- Working Area Size – the size of the working area; independent from its real shape here always a squared size is assumed and the biggest used size has to be entered here

7.4 Correct Single and Multiple Points Dialogue

Both, the single and the multiple point dialogues expect the same values and behave the same except the following:

- the single point dialogue expects exactly one nominal and one actual coordinate value pair
- the multiple point dialogue asks for a CSV file that contains pairs of nominal and actual coordinate values and shows an editable list of them within the dialogue;
  The CSV-file contains comma-separated values in format
  `nominal x, nominal y, actual x, actual y,`
  using dot-separated floating point numbers

The dialogues offer the following fields and input values:

- Nominal Position X, Nominal Position Y – the nominal coordinate(s) in unit mm
- Actual Position X, Actual Position Y – the actual, measured coordinate(s) in unit mm
- Working Area Position X and Y – the top left position of the working area in unit mm
- Working Area Size – the size of the working area in unit mm, here a squared working area is assumed

When the dialogue is left with “OK” the current correction table is modified according to the given nominal and actual value(s).

7.5 Spatial Correction Dialogue

This correction is inteded for cases where the scanhead is not mounted in parallel to the working area but is inclined (e.g. in order to mark in inner side of a ring). It requires a correction that already fits to the used scanhead and optic, the spatial correction then can be applied to it as additional parameter. Thus following

steps have to be performed in order to get an exact spatial correction:

- mount the scanhead in parallel to the working area in a way where the distance between scanhead and working area is the same length like the shortest distance between scanhead and working area when mounted inclined
- check if the correction for this scanhead fits and possibly modify it using the functions described above
- once you can mark an exact square that does not show any distortions, measure the size of that square and mount the head inclined
- mark the same square again; now that side of formerly square where distance between working area and scanhead is biggest has to be measured

The resulting two values - the normal (expected) length of the square measured in first step and the new, bigger length of the square measured after inclining the scanhead - have to be entered in dialogue for spatial correction. As an other information that side of the current correction table where the square has been distorted and where its edge is longer than before has to be marked using the radio buttons.

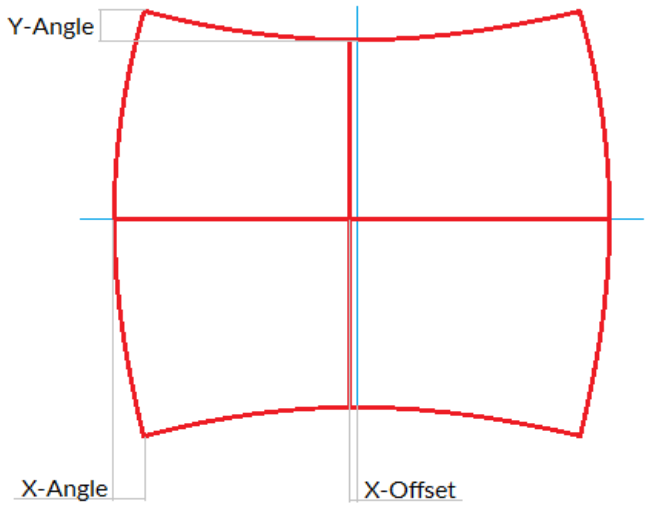
After pressing OK-button the spatial correction that stretched the squares width with growing distance will be applied to the correction table. Now the result is a rectangle. To correct its length to get back to a square, the gain factor of this direction has to be changed in scanner card settings (within BeamConstruct) or the width/height of the whole correction table has to be changed in Geometry-Panel of CorrCorrect directly.

7.6 CorrCorrect Cookbook

Following some steps are described to get a suitable and accurate correction file that fits to a custom scanhead hardware set-up. Since every combination of scanhead with a specific lens results in an own, specific distortion, a separate correction file is needed for every scanhead-lens-combination. For the same combination typically the same correction file can be used.

By performing following steps a suitable correction file can be created:

1. Generate a project within BeamConstruct that consists of a square which is nearly as large as the whole working area (edge length should be about 96% of the maximum working area size that is possible with a scanhead and can be 100% in a second step when an already created correction file has to be verified and refined).
2. Add a horizontal and a vertical line to this project, both should be exactly centred vertically and horizontally to the working area, crossing each other in exact centre.
3. Mark this project on a material where the marking lines can be seen as clear and sharp as possible.
4. Check the marking result, the edges of the square and the position of the horizontal and vertical lines (please refer to example in image below).



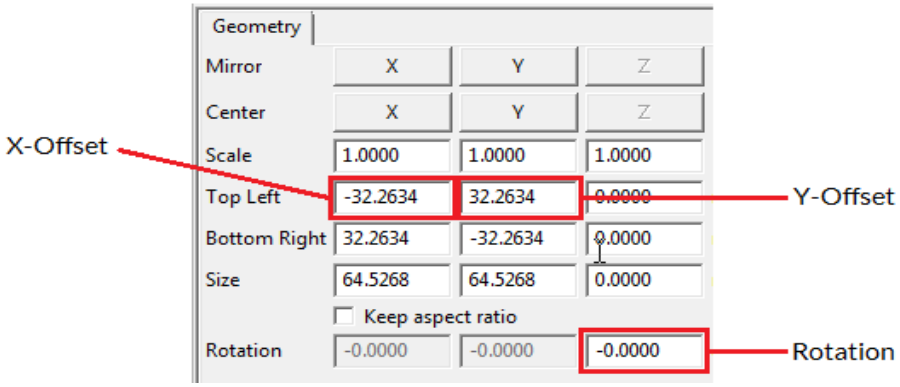
Now switch over to CorrCorrect. When the square is distorted at top and bottom edge (distance marked in upper left corner in picture above), this distortion has to be corrected with the "Y-Angle" parameter in Correction Definition Dialogue. The stronger this deviation is, the larger the Y-Angle value should be.

When the square is distorted at left and right edge (distance marked in lower left corner in picture above), this distortion has to be corrected with the "X-Angle" parameter in Correction Definition Dialogue. The stronger this deviation is, the larger the X-Angle value should be.

When the horizontal line is not marked at exact centre position of the working area (distance marked in lower middle position in picture above), after creation of a new correction file via Correction Definition Dialogue it has to be selected and shifted in vertical position. This can be done in tab-pane "Geometry" of CorrCorrect by moving it against the direction of its deviation (depending on the strength of distortion this line may also be bent a bit, in this case the distance from centre point has to be used for shifting).

When the vertical line is not marked at exact centre position of the working area, after creation of a new correction file via Correction Definition Dialogue it has to be selected and shifted in horizontal position. This can be done in tab-pane "Geometry" by moving it against the direction of its deviation (this deviation is not shown in example picture above).

In some cases it also may be necessary to rotate the correction table (e.g. when the marking result



and its distortion is rotated by 90 degrees comparing to the example picture above). The rotation angle of 90 degrees can be entered in tab-pane "Geometry" once. Smaller/other rotation angles that compensate mounting accuracies should not be set here but by using parameter "Rotation" in scanner card settings.

5. Save the generated correction file, configure BeamConstruct to make use of it and mark the same geometry again. Depending on the result you may now continue with 4.) to refine the correction or finish here when the marked square does not show any distortions.

Some important things that have to be mentioned for the process of creation of correction tables:

- There is no direct relation between the pincussion distortion of an edge of the marked square and the angle value to be set to adjust it, means there is no simple formula to calculate the angle-value out of a measured distance in marking result. Thus it is necessary to try in several steps to find out the correct values. When a given angle does not fully remove a distortion, it has to be set to a larger value in next step. When a used angle over-compensates a distortion and causes an edge to be bent in other direction, it has to be set to a smaller value.
- A correction table shown in CorrCorrect always has a base-size of 65,535 x 65,535 mm. This is an artificial size value that has to be used by CorrCorrect because this application does not know anything about used hardware set-up and real working area size. Thus a measured X- or Y-offset in real working area always has to be set in relation to this artificial working area size. The offset value to be entered within CorrCorrect then has to be a value calculated out of the real offset and the relation between real working area size in BeamConstruct and 65,535 mm artificial working area size in CorrCorrect.
- After saving a new correction file, it is mandatory to let BeamConstruct close a possibly existing connection to a scanner controller card. This can be done by going into project settings. BeamConstruct loads a correction file only once on initialisation of scanner controller card, thus a new initialisation has to be forced this way. When this is not done, BeamConstruct will continue working with a preceding correction table and will not notice there are any new data saved into the same file.

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z