Novanta

InScript Software

User Manual
Version 5.0.1.65

Read carefully before using.
Retain for future reference.

Manufacturer

Novanta Europe GmbH
Werk 4
92442 Wackersdorf
Germany

Phone: +49-9431-7984-0
Email: Photonics@Novanta.com

Customer service

Before contacting us for assistance, please review appropriate sections in this manual that may answer your questions. After consulting the manual, please submit a request through our website: https://novantaphotonics.com/technical-support-request-form-header/

Americas, Asia Pacific

Novanta Headquarters
Bedford, USA
Phone: +1-781-266-5700

Email:
Photonics@Novanta.com

China

Novanta Sales & Service Office

Shenzhen, China
Phone: +86-755-8280-5395

Suzhou, China
Phone: +86-512-6283-7080

Email:
Photonics.China@Novanta.com

Europe, Middle East, Africa

Novanta Europe GmbH
Wackersdorf, Germany
Phone: +49-9431-7984-0

Milan, Italy
Phone: +39-039-793-710

Email:
Photonics@Novanta.com

Japan

Novanta Service & Sales Office

Tokyo, Japan
Phone: +81-3-5753-2460

Email:
Photonics.Japan@Novanta.com

Table of Contents

List of Figures

List of Tables

1 Manufacturer and customer support

Manufacturer

Novanta Europe GmbH

Werk 4

92442 Wackersdorf

Germany

Phone: +49-9431-7984-0

Email: `Photonics@Novanta.com`

Customer service

Before contacting us for assistance, please review appropriate sections in this manual that may answer your questions. After consulting the manual, please submit a request through our website: https://novantaphotonics.com/technical-support-request-form-header/[1]

Americas, Asia Pacific

Novanta Headquarters

Bedford, USA

Phone: +1-781-266-5700

Email: `Photonics@Novanta.com`

China

Novanta Sales & Service Office

[1] https://novantaphotonics.com/technical-support-request-form-header/

Shenzhen, China

Phone: +86-755-8280-5395

Suzhou, China

Phone: +86-512-6283-7080

Email: `Photonics.China@Novanta.com`

Europe, Middle East, Africa

Novanta Europe GmbH

Wackersdorf, Germany

Phone: +49-9431-7984-0

Milan, Italy

Phone: +39-039-793-710

Email: `Photonics@Novanta.com`

Japan Novanta Service & Sales Office

Tokyo, Japan

Phone: +81-3-5753-2460

Email: `Photonics.Japan@Novanta.com`

2 How To Use This Document

2.1 Pictographs And Signal Words

This user manual uses the following pictographs and signal words for information of particular importance.

> **⚠ CAUTION**
>
> … indicates a hazardous situation with *low degree of risk* which, if not avoided, *could* result in *minor or moderate injury*.

> **NOTICE**
>
> … is used to address practices not related to personal injury which, if not followed, *could* result in *material damage*.

> **TIP**
>
> … is used to address practices which facilitate work.

2.2 Notation Conventions

This user manual uses notation conventions as listed in table 2.1 on page 4.

Table 2.1: Notation conventions

Notation	Meaning
underlined (only on screen)	cross reference or hyperlink
[<ID>]	cross reference to external document listed in chapter *Bibliography*
Bold	element in the graphical user interface that shall be clicked
Italic	name of element in the graphical user interface or emphasized text
`Teletype`	text to be typed in by the user or file name or location in the file system

2.3 Storage And Replacement

Keep this user manual with the InScript software to access the information at any time during the InScript software's lifetime.

This user manual is part of the InScript software. When selling the InScript software also deliver this user manual to the new owner.

You can request replacement for this user manual at the manufacturer.

2.4 Specifics And Structure Of This Document

This user manual is geared to the InScript software's life cycle.

The InScript software can connect to multiple ASC system controllers. Each ASC system controller provides by its firmware controller-specific functionality and firmware user manual. That firmware user manual contains, amongst others, information on supported devices and job nodes. As the firmware user manual comes with the firmware it is not contained here. Later on, when getting to know the InScript software, read about how to access the firmware user manual in section 8.4 on page 26.

3 General Safety Instructions

3.1 Normal Use

Optical scanning systems require a control environment that allows the scan head, the laser, as well as all connected hardware to be used to their full performance potential.

This control environment consists of the ARGES system controller (ASC) hardware and the associated InScript software that is running on a PC which is connected via Ethernet to the ASC. Find information about how to connect the PC to the ASC in the operation manual for the ASC [1].

Normal use of the InScript software is to control the devices which are connected to the ASC. ARGES scan heads, lasers and other devices supported by the InScript software, firmware and ASC hardware are counted among these devices. Find information about how to connect these devices to the ASC in the instruction manual for the ASC [1] and the documentation to the respective devices.

The InScript software provides an user-friendly, object-oriented user interface that allows integrating all external devices, to define complex process sequences and to set up and manage all controlling parameters. Find information necessary to operate the InScript software in this user manual and the documentation on devices and job nodes [3].

To read, understand and follow this user manual, the documentation on devices and job nodes [3] and the operation manual for the ASC [1] is part of normal use.

Any other use is not normal.

3.2 Audience And Qualification

Before installing, commissioning and using the InScript software, completely read through this user manual to employ all functions in a correct and safe way.

This user manual addresses *qualified personnel with administrator rights* who are assigned with installing and commissioning a machine where an ASC system controller is incorporated, but not necessarily using the InScript software directly.

This user manual also addresses *qualified personnel* who are assigned with using the machine where an ASC system controller is incorporated and thus assigned with using the InScript software.

3.3 Safety Instructions

The InScript software is part of the incomplete machine ARGES System Controller (ASC). One of the ASC's functions is to control a laser.

Commissioning of the ASC is prohibited until it has been established that the machine into which the ASC is incorporated complies with all essential requirements of the 2006-42-EG *Machinery Directive* and the safety requirements and measures of standard DIN EN ISO 11553-1 *Safety of machinery – Laser processing machines – Part 1: General safety requirements*.

4 Overview

4.1 What Is The InScript Software?

Optical scanning systems require a control environment that allows the scan head, the laser, as well as all connected hardware to be used to their full performance potential. The control environment provided by ARGES consists of a system controller and the included InScript software.

InScript provides an user-friendly, object-oriented interface that allows integrating all external devices, to define complex process sequences and to set up and manage all controlling parameters. InScript has the ambition to be the most powerful commercial laser processing software.

4.2 Integration

There are 3 possible scenarios how to integrate the InScript software into your system.

In all scenarios the InScript software is installed on a PC with Windows operating system. The PC is connected via Ethernet to an ASC system controller. The operation manual for the ASC [1] gives information about how to connect the ASC to this PC. And finally the hardware to be controlled is connected to this ASC system controller.

Figure 4.1 on page 8 illustrates these scenarios.

The difference between the 3 scenarios is the way operating the InScript software. The skills that the user needs, increase from the upper to the lower scenario in the figure.

In the upper case you operate the InScript software directly via its graphical user interface. Find details in this user manual.

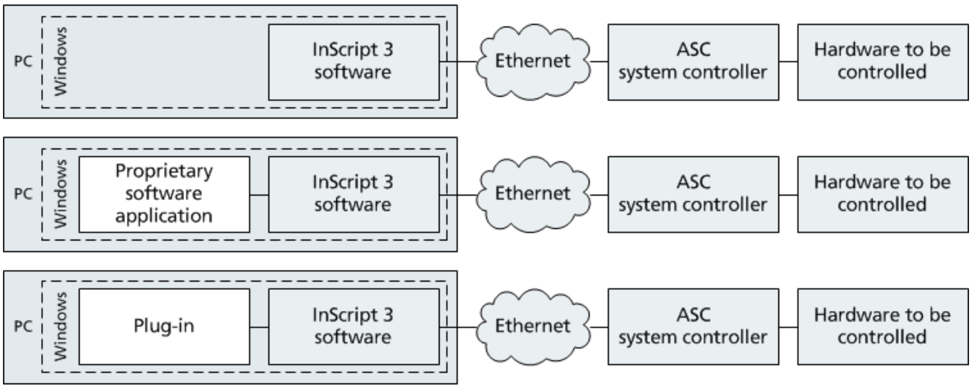


Figure 4.1: Possible scenarios integrating the InScript software

In the middle case your proprietary software application operates the InScript software via its integrated ControllerLib. Find details in the ControllerLib interface description [2].

In the lower case plugins operate the InScript software. Find details in the plugin documentation [4].

Functions for integration:

- Use drivers for all common industrial lasers
- Synchronize laser processing with connected devices
- Synchronize laser processing with moving workpieces via encoder inputs
- Monitor status and make use of the parallel system messaging
- Track the workpiece and machining quality via a connected camera
- Control laser machining by communication between InScript and a connected image processing system
- Write scripts in the integral language resembling in syntax and semantics the programming language C
- Mark groups of objects selected through hardware inputs
- Use multiple scan heads or tiling to mark areas bigger than the scan field
- Teach-in coordinates via mouse and trackball

- Preview the marking on the workpiece by using the pilot laser
- Control jobs via InScript's DCOM interface from your own application while InScript is running in the background
- Embed the ControllerLib program library into your own software application and use full read/write access to variables, functions and parameters, e.g. for personalization and automation tasks
- Order additional drivers for further devices at ARGES

4.3 Features

Use the user-friendly graphical user interface of InScript for all ARGES system controllers.

Create and manage complex laser machining processes, so called jobs, for precession drilling, cutting, trepanning, engraving, welding, hardening, laser direct imaging as well as for eye treatment, image acquisition and more.

Control and manage devices connected to the system controller, such as beam switches, beam polarizers, beam attenuators, shutter, pilot lasers, positioning stages, PWM-controlled servomotors, stepper motors, laser cooling units, crossjets, gas nozzles and actuators of every description.

Control beam position with an automatic beam position stabilization, the illumination and camera recordings of the workpiece or the machining process.

Read data from laser power and pulse energy meters, sensors and encoders of every description, distance measuring equipment, USB- and firewire-cameras, tracking systems, optical coherence tomographs, Shack-Hartmann wavefront sensors and more.

Synchronize your laser process with the workpiece movement and industrial PLC systems.

Protect your colleagues, patients and your system by emergency stop switch, safety circuits and real time monitoring of the scanner movement.

Integrate other software such as image processing and CAD systems as well as proprietary software applications.

Control from your own software applications, as an alternative to InScript, many aspects of the ARGES system controller by using the included ARGES ControllerLib, a C/C++-based library with DLL

4.4 Functions

Find more detailed information about the functions in this user manual. Find a brief overview below.

An assistant supports you during the correction of the scan field distortion

- Correct the scan field distortion for laser, pilot laser and camera separately and thus enable high precision laser marking, preview on the workpiece and teach-in of coordinates
- Correct the scan field distortion for f-Theta objective or focus translator in 2D/3D and high resolution
- Visualize correction data in 2D/3D for manual post-editing
- Compensate position dependent variation of laser power with the power equalizer
- Use the interface to make use of external measuring and calibration devices

Manage the hardware through precisely tailored drivers either via directly connected system controllers or via network

- Drive all common industrial lasers
- Drive single and master/slave scan heads with up to 8 channels
- Drive sensors and actuators via Ethernet, EtherCAT, PROFIBUS and USB
- In- and output user-specific data via RS-232/485, Ethernet and discrete in- and outputs
- Drive USB- and Firewire-cameras via Windows device drivers

Edit complex processes in a sophisticated graphic editor and in a clear object-oriented tree structure

- Quickly arrange text, images, barcodes and other objects
- Simply modify graphical and other objects via specific editors
- Scale, rotate, slant and move objects
- Distribute objects to any vector path
- Use a backdrop images to align markings
- Manage unlimited groups of process parameters, so called pens
- Reduce time of design by inheriting process parameters or behavior to branch of objects in the tree structure
- Use the internal scripting language in order to read or write parameters via Ethernet and external software applications
- Visualize and optimize sky-writing with the vector editor

Text objects in 2D and 3D

- Type text in TrueType-, dot-matrix-, high-speed-vector-fonts without crossovers, or in your own fonts
- Insert date, time, serial numbers, process parameters or user-defined values of variables on-the-fly
- Import text on-the-fly from text files (∗.ini, ∗.txt), MS Excel spreadsheets (∗.xls) and MS Access (∗.mdb), or dBase (∗.dbf) databases
- Control paragraph breaks and alignment
- Align text to circles and ellipses
- Use UTF-8 and native 8 Bit encodings

Raster image objects in 2D

- Import the file formats: BMP (∗.bmp), JPEG (∗.jpg, ∗.jpe, ∗.jpeg, ∗.jfif), TIFF (∗.tif, ∗.tiff), PNG (∗.png), GIF (∗.gif), Targa (∗.tga, ∗.vst, ∗.icb, ∗.vda, ∗.win), PCX (∗.pcx, ∗.pcc, ∗.scr), PhotoShop (∗.psd, ∗.pdd), Portable Map (∗.ppm, ∗.pgm, ∗.pbm) and SGI (∗.sw, ∗.rgb, ∗.rgba, ∗.sgi)
- Output black and white, pseudo and real grayscale images
- Output raster images bidirectionally and optimized and in high speed omitting white spaces in the raster image

- Generate raster images from vector, text objects and/or raster images on the fly
- Re-sample raster images and make use of the sophisticated constraints management (size, resolution and pixel mapping), e.g. for personalization tasks

Vector drawing objects in 2D and 3D

- Output dots, shapes (line, rectangle, ellipse), polygons, concentric circles, perfectly smooth circles and spirals
- Output barcodes (code 39, code 93, code 128, code 2/5 interleaved, codabar, EAN 13, EAN 8, UPC A, UPC E, 2D Datamatrix)
- Fill with customizable hatching pattern, spacing and reduction from outline
- When resizing a filling either maintain spacing or re-scale it
- Achieve any drilling geometry by using a special 3D spiral with power attenuation and laser beam precession
- Import and edit the file formats: ARGES Rawlines (∗.arl), Scalable Vector Graphics (∗.svg), AutoCAD Drawing eXchange Format (∗.dxf) and HPGL (∗.hpgl, ∗.plt),
- Transform objects in real 3D including offsetting, rotating and scaling

5 System Requirements

The PC, you want to install InScript on, has to comply with the following requirements:

Operating system

64-bit Microsoft® Windows® 10 semi-annual channel

CPU

Recommended: 3.0 GHz or greater; 4 or more cores

Minimum: 2.0 GHz; single core

Memory

Recommended: 16 GB RAM or more

Minimum: 4 GB RAM

Disk space

Installer plus full installation: 2 GB

Graphics

Recommended: 4 GB GPU with 192-bit memory bus; OpenGL 3.2 compliant

Minimum: 1 GB GPU with 64-bit memory bus; OpenGL 3.2 compliant

Display resolution

Recommended: 1920 × 1080 pixel or higher

Minimum: 1280 × 1024 pixel

Pointing device

MS-mouse compliant

Network

For commissioning InScript an ASC system controller with firmware has to be connected via an Ethernet (100/1000 Mbit/s; TCP/IP protocol) with the PC, where InScript is installed. For how to connect the ASC system controller with this PC see the instruction manual of the ASC [1].

System components

System components that shall be controlled by the ASC system controller, but at least the scan head and the laser, have to be mechanically and electrically installed and connected to the ASC system controller. For how to connect the ASC system controller with the scan heads see the instruction manual of the instruction manual of the ASC [1] and the respective scan head model. For how to connect the ASC system controller with these devices see the instruction manual of the ASC [1] and the respective device manuals.

6 Installing

Audience And Qualification

This chapter addresses *qualified personnel with administrator rights* who are assigned with installing a machine where an ASC system controller is incorporated, but not necessarily using the InScript software directly.

Procedure

1. Start the installer `InScript_<version>.exe`.
2. Follow the instructions on screen.

   > **TIP**
   >
   > Depending on your decision during installation to install InScript either for all users or for the current user only, all files will be copied to the subfolder `\Novanta\InScript` in the `ProgramData` path or `Documents` path of the user respectively.
   >
   > InScript will use this folder as the base for all file operations. The `ProgramData` path or `Documents` path of the user respectively has different locations on different Microsoft Windows versions. Locate the path on your system by clicking **File** → **Explore application data** or **File** → **Explore documents** respectively when running InScript later on.

3. Once InScript is installed, start InScript as described in chapter 7 on page 16.

7 Starting

7.1 Audience And Qualification

This chapter addresses *qualified personnel* who are assigned with starting a machine where an ASC system controller is incorporated, but not necessarily using the InScript software directly.

7.2 Requirements

- The system components that shall be controlled by the ASC system controller have to be switched on and ready.
- The ASC system controller has to be switched on and ready.

7.3 Starting Via The Windows Start Menu

Procedure

1. In the Windows *Start* menu, scroll down and expand the *Novanta* folder.
2. Click one of the following flavors.

   InScript starts InScript

   InScript (Support) starts InScript without plugins, deletes all InScript log-files, enables the RequestLog for this session and shows more details in some views of InScript

   InScript (without Plugins) starts InScript without plugins, even if plug-ins are installed

   InScript starts.

3. If you are running InScript for the first time or updating it then commission it as described in chapter 9 on page 28.

7.4 Starting Via The Keyboard

Procedure

1. Press the **Windows**-key.

   The Windows *Start* menu opens.

2. In the search box, type in `inscript`.
3. Press the **Enter**-key.

   InScript starts.

4. If you are running InScript for the first time or updating it then commission it as described in chapter 9 on page 28.

7.5 Starting Via The Command Line

Requirement

You already added the *InScript.exe* program file's path to environment variable *Path*.

Procedure

1. Press the key combination **Windows**+**R**.

   The *Execute* window opens.

2. In *Execute*, type in `inscript`.

   You may add command line arguments from section 7.6 on page 18.

3. Click **OK**.

   InScript starts.

4. If you are running InScript for the first time or updating it then commission it as described in chapter 9 on page 28.

7.6 Command Line Arguments

Command line arguments can be specified in one of these forms:

```
-name=value

-name value

--name=value

--name value
```

Command line argument description

configuration *file* specifies a configuration file which InScript shall use instead of file `InScript.ini`, where *file* is the path and file name of this configuration fie

Example:

```
-configuration=”C:\Users\<user name>\Documents\ARGES\InScript3\
Config\My Configuration.ini”
```

controller *UniqueHardwareID* specifies the controller which InScript shall use instead of the controller list which is defined in file `InScript.ini`, where *UniqueHardwareID*is the unique hardware ID of this controller

Example:

```
-controller=ASC#192.168.100.216:1610
```

debugContext writes detailed error messages from the OpenGL graphics driver to file `OpenGL.log`

h, help, ? shows this list of command line arguments

layout *file* specifies a layout file which InScript shall use instead of file

`standard.lay3`, where *file* is the path and file name of this layout file

Example:

```
-layout=”C:\Users\<user name>\Documents\ARGES\InScript3\Layouts\
My Layout.lay3”
```

noPlugins starts InScript without plugins, even if plugins are installed

support deletes all InScript log files, starts InScript without plugins, enables the RequestLog for this session and shows more details in some views of InScript

v, version shows the version number

The InScript software is written with the Qt application framework. All Qt programs automatically support the following additional command line arguments:

Command line argument description

graphicssystem *value* sets the backend to be used for on-screen widgets and QPixmaps where *value* is one of *raster* or *opengl*

qmljsdebugger *port* activates the QML/JS debugger with a specified port. The *port* must be of format `port:1234[,block]`, where `block` is optional and will make the application wait until the debugger connects to it

reverse sets the application's layout direction to *Qt::RightToLeft*

session *value* restores the application from an earlier session

style *value* sets the style of the graphical user interface, where *value* is on of *motif*, *windows* or *platinum*. If you have compiled Qt with additional styles or have additional styles as plugins then these will be available to the argument

stylesheet *file* sets the application style sheet file, where *file* is the path and file name of this style sheet file; Note: Relative URLs in the style sheet file are given relative to the style sheet file's path.

widgetcount shows a debug message about the number of widgets left undestroyed and the maximum number of widgets that existed at the same time when exiting the application

8 Getting To Know InScript

The InScript software is a classical desktop application.

It consists of several different views, a status bar, a menu and toolbars. Additionally some of the views have a context menu. The menu, the toolbars and the context menu are used to execute the operations which the software provides. The views and the status bar visualize different aspects of the software.

8.1 Menu, Toolbars and Context

In InScript all toolbars and the menu belong to the main window of the application. Thus, no view does have its own toolbar or its own menu, except a context menu. Because of that it may not always be clear which object will be modified by a menu or a toolbar action. To solve this problem we developed the concept of so called *contexts*. Each operation in the InScript menu or on one of the toolbars applies to the associated context. Since there are 3 different groups of objects which may be modified by InScript we have 3 different types of contexts – a node context, a controller context and a view context. The node and the controller context are set by the selection in the *Navigator* view, the view context is defined by the current view.

8.2 Views

InScript provides several types of views to visualize different aspects of the application. Generally we distinguish between fixed views and user views.

Fixed views The *Navigator* and *Messages* view are so important that they are required most of the time. That is why these views cannot be closed and they cannot be moved inside the application window. Because they have a fixed position we call them fixed views. Fixed views can be hidden for a short time to maximize the workspace for other views.

User views All other kind of views can be created by the user on demand. That's why we call them user views. User views can be docked together in the main window of the application, undocked (made floating) or stacked.

For more details see chapter 10 on page 47.

8.2.1 Navigator View

The *Navigator* view is one of the most important elements of the InScript software. On the top level it is a list of controllers you want to work with. If a controller is connected with the InScript software (that means both communicate with each other), the controller may be expanded to a subtree of several nodes. Each connected controller has 4 subnodes:

- Jobs
- Pens
- Fonts
- Devices

Each of these nodes is a container for a certain type of node. If you add, remove or change a job, pen, font or device then this will result in a change of one of these nodes. If you load a new job and the function executes successfully, you will see that a new subnode respectively subtree was created inside the *Jobs* node.

The focus respectively the selection in the *Navigator* view plays a very important part in InScript – it defines the node context and the controller context. If the selected object in the navigator tree represents a node on the controller, this node will be set as the node context and because each node belongs to a certain controller this controller will be set as the controller context accordingly. If the selected object in the navigator tree represents no node but a controller, then the root node of this controller will be set as the node context and the controller itself will be set as the controller context accordingly. If the selected object in the navigator tree represents the *Devices* node, then the node context will be set empty and the controller context will be set to the controller this node belongs to.

The tree in the *Navigator* view has 2 columns – *Resource* and *State*. The *Resource* column contains the actual tree and the *State* column serves as a display for several states like the controller connection state, the selection, or which views are associated with the object.

The *Navigator* view has a context menu which will usually be opened by a mouse right click on a tree node. The context menu provides many actions for the different node types. Among other things it can be used to expand and collapse a single tree node (menu entries *Expand node*, *Collapse node*) or a complete subtree (*Expand subtree*, *Collapse subtree*). Due to the fact that the context menu may have lots of entries several entries will only be shown if they make sense e.g. the *Expand node* menu entry will not be shown for nodes which are expanded already.

- The *Navigator* view shows controllers and their nodes.
- The *Navigator* view is a fixed view.
    - There is exactly 1 *Navigator* view.
    - You cannot undock the *Navigator* view.
    - You cannot close the *Navigator* view.

For more details see chapters 18 on page 89.

8.2.2 Messages View

The *Messages* view displays system messages. It provides 2 different kinds of view on messages. In the list view all current messages are shown in a list. If there are messages that have submessages then the list will be expanded to a tree and the submessages will be shown in the subtrees. The detail view shows all elements of a single message in detail. So it can be well used to get more information about a message, its cause and possible remedies.

All actions which may be performed on messages are grouped in the *Messages* menu.

- The *Messages* view shows system messages.

- The *Messages* view is a fixed view.
  - There is exactly 1 *Messages* view.
  - You cannot undock the *Messages* view.
  - You cannot close the *Messages* view.

For more details see chapter 19 on page 99.

8.2.3 Job Control View

The *Job Control* view serves as the central control for the job execution on the controller. The job execution may be started, paused or aborted immediately.

- The *Job Control* view controls the selected job on the selected controller (controller context; see also section 8.1 on page 20).
- The *Job Control* view is a fixed view.
  - There is usually 1 *Job Control* view.
  - You cannot undock the *Job Control* view.
  - You can close and open the *Job Control* view.

For more details see chapter 20 on page 102.

8.2.4 Vector Editor View

With the help of the *Vector Editor* view jobs and their output can be visualized and edited interactively.

- The *Vector Editor* views show graphical representations of job nodes.
- The contents of a *Vector Editor* view can be locked. If the view is not locked, its contents follows either the node context or the job context which is the job the node context belongs to; see also section 8.1 on page 20.
- The *Vector Editor* view is a user view.
  - You can open none, one or several *Vector Editor* views.
  - You can dock, stack or float *Vector Editor* views.

For more details see chapter 21 on page 106.

8.2.5 Node Properties View

The purpose of the *Node Properties* view is the convenient editing of the most important variable values (properties) of a node. For that a predefined user interface (UI) for each node will be shown. If no UI is available for a certain node this view remains empty. In that case or if a certain variable is not shown in the predefined UI then use the *Inspector* view to modify these variable values.

- If a predefined UI is available for the selected node then the *Node Properties* view presents the most important variables or properties of a node for editing else edit them in the *Inspector* view.
- The contents of a *Node Properties* view can be locked. If the view is not locked, its contents follows the node context.
- The *Node Properties* view is a user view.
    - You can open none, one or several *Node Properties* views.
    - You can dock, stack or float *Node Properties* views.

For more details see 22 on page 138.

8.2.6 Inspector View

Use the *Inspector* view to inspect, rename or edit all subnodes of a node, including its variables. If the node shown in the inspector is the root node of a controller, then the complete variable tree of a controller will be shown in the *Inspector* view.

- The *Inspector* views shows variable values and variable containers of nodes for editing.
- The contents of an *Inspector* view can be locked. If the view is not locked, its contents follows the node context; see also section 8.1 on page 20.
- The InScript *Inspector* view is a user view.
    - You can open none, one or several *Inspector* views.

  - You can dock, stack or float *Inspector* views.

For more details see chapter 23 on page 139.

8.2.7 UI File Container View

The *UI File Container* view can be used to load and show special *User Interface files* (or forms). *User Interface files* (`*.ui`) which use ARGES Widgets make it possible to show and/or to edit a bunch of variables of the context node or of one or more certain nodes of a controller. Thus they can act as node or device editors as well as an user interface for all variables of a job which shall be changeable by the operator.

- The *UI File Container* view shows a bunch of variables of the context node or of one or more certain nodes of a controller for editing.
- The contents of an *UI File Container* view can be locked. If the view is not locked, its contents may follow the node context. This depends on the UI file.
- The *UI File Container* view is a user view.
  - You can open none, one or several *UI File Container* views.
  - You can dock, stack or float *UI File Container* views.

For more details see 24 on page 144.

8.3 Undo / Redo Support In InScript

InScript supports undo/redo for almost all operations which are performed on the controller directly. That covers all operations on nodes and variables like creating, deleting, renaming and changing values. Even most of the interactive modifications done in the vector editor are undo-able, because these are modifications of one or more variables in fact.

Operations that cannot be undone:

- changing the dimensions of the main form
- changing the dimensions of a floating view

- changing the docking state of a view
- creating and removing a view
- changing the vector editor selection
- adding and removing controllers to/from the navigator view
- changing the connection state of a controller
- changing the properties of a view, e.g. switch between list and detail mode in the message view
- changing the preferences
- file operations

8.4 Where To Find Information

8.4.1 Viewing The User Manual On Screen

The InScript user manual consists of 2 parts.

InScript software provides the first part, the InScript User Manual. This part contains information on how to use InScript software.

The ARGES System Controller (ASC) provides the second part, the Firmware User Manual. This part contains information about controller specific functionality. Essentially, this is information about executable job nodes and supported devices.

Procedure

1. In the InScript software in the *Navigator* view select the desired controller.

   If you cannot find the controller you want in the *Navigator* view, you will need to add it first. See the *InScript User Manual* to learn how to add a controller. You can always open this user manual by performing step 3 of this procedure without first connecting InScript software to the controller.

2. In the controller's context menu, click **Connect**.
3. In the menu click **Manuals** > **Online Manuals**.

The *Help* window opens. It displays the *InScript User Manual* and, if a controller is connected to InScript software, the controller-specific *Firmware User Manual*.

8.4.2 Printing The Documentation

If you wish to print the documentation then find printable PDF-files in the `documentation` folder of the InScript software installation.

8.4.3 Other Documentation

Find the following documentation in the `documentation` folder of the InScript software installation.

If your proprietary software application handles the InScript software via its integrated ControllerLib then find additional information in the ControllerLib interface description [2].

If Plug-ins handle the InScript software then find additional information in the Plug-in documentation [4].

9 Commissioning

9.1 Audience And Qualification

This chapter addresses *qualified personnel* who are assigned with commissioning a machine where an ASC system controller is incorporated.

9.2 Requirements

System components that shall be controlled by the controller, but at least the scan head and the laser, have to be mechanically installed and electrically connected to the controller. This is subject of the respective instruction manuals.

9.3 Safety Instructions

The InScript software is part of the incomplete machine ARGES System Controller (ASC). One of the ASC's functions is to control a laser.

Commissioning of the ASC is prohibited until it has been established that the machine into which the ASC is incorporated complies with all essential requirements of the 2006-42-EG *Machinery Directive* and the safety requirements and measures of standard DIN EN ISO 11553-1 *Safety of machinery – Laser processing machines – Part 1: General safety requirements*.

9.4 Adding A Controller

When running InScript for the first time you have to add a controller to InScript in order to assign it to InScript.

Procedure

1. In the menu, click **Controller** → **Add**.

   The *Enter IP Connection* window opens.

2. Enter the IP-address of your ARGES System Controller.
3. Click **OK**.

   The controller will be listed by its IP-address in the *Navigator* view.

   > **TIP**
   >
   > InScript is able to manage more than 1 controller. If you want to add more controllers then repeat the steps above.

4. If you want to use the added controller with InScript then connect it to InScript as described in section 9.5 on page 29.

9.5 Connecting To The Added Controller

Procedure

1. In the *Navigator* view, select the controller.
2. In the Menu, click **Controller** → **Connect**.

   If the connection can be established then the controller node automatically expands to nodes *Jobs*, *Pens*, *Fonts* and *Devices*.

   InScript saves the state of the connected controller and automatically reconnects it after restart.

   > **TIP**
   >
   > If you want to connect to another controller then repeat the steps above.

> Prevent working on the wrong controller by disconnecting the controllers which you do not intend to work on. To do so, select the controller and click **Controller** → **Disconnect** in the main menu.

3. Optionally install licenses, as described in section 9.6 on page 30.

   -OR-

   Set up suitable devices for your laser and further devices now, as described in section 9.7 on page 34.

9.6 Licensing

9.6.1 Overview

InScript itself does not need to be explicitly licensed. InScript and the ASC controller can be used normally but without the functionality of the license packages. The next section describes these license packages.

If you want to use one or more of the license packages then you need a license file. You can obtain license files from Novanta. Each license file is bound to a specific ASC controller.

9.6.2 License Packages

9.6.2.1 Precession Drill Micromachining

> **TIP**
>
> Note, in addition to a license also either a PE 2 or PE III scan head is required.

The *Precession Drill Micromachining* package allows the user to control both the position of the laser spot on a work piece as well as the beam incidence angle. For drilling processes, the beam incidence angle is also called the precession angle. The scan head generates the desired motion of the beam focus and of the precession angle as a sequence of elementary motions. Each elementary motion is described by a defined set of parameters. Sequences of elementary motions then allow the user to generate circular, elliptical, spiral, helical or other focus motions, and to vary the precession angle. The extremely flexible programming approach enables the user to drill holes with innovative geometries: it is possible to vary the shape, conicity and taper angle along the height of the hole.

Contains the following features:

- PrecessionDrill job node

9.6.2.2 OCT Process Monitoring

> **TIP**
>
> Note, in addition to a license also WFI hardware is required.

The *OCT Process Monitoring* package is an Optical Coherence Tomography system for process monitoring and topology measurement.

Contains the following features:

- WFI device
- WFI job nodes
- WFI plugin

9.6.2.3 Battery Foil Cutting

The *Battery Foil Cutting* package allows the user to connect position encoders to the ASC controller.

Contains the following features:

- On-The-Fly
- Always-On On-The-Fly

The difference between *On-The-Fly* and *Always-On On-The-Fly* is that during the *Always-On On-The-Fly* process the laser power and line speed are adjusted to match the current role speed. During the *On-The-Fly* process the laser is turned on and off between segments.

9.6.3 Viewing the installed licenses

There are various ways to find out which licenses are installed.

> **TIP**
>
> Please note that the license file contains the license for one or more packages. If no license file is installed then these packages count as *not licensed*.

9.6.3.1 Controller Context Menu

Requirement

- InScript is connected to the ASC controller in question.

Procedure

1. In the *Navigator* view, open the ASC controller's context menu (right click).
2. In the context menu, click **View license information**.

   This opens the *License Information* window.

9.6.3.2 Main Menu

Requirement

- InScript is connected to the ASC controller in question.

Procedure

1. In the *Navigator* view, select the ASC controller.
2. In the main menu, click **Licensing** > **View license information**.

   This opens the *License Information* window.

9.6.3.3 About Dialog

Requirement

- InScript is connected to the ASC controller in question.

Procedure

1. In the main menu, click **About** > **About**.
2. In the *About InScript* window, click **License information**.

   This opens the *License Information* window.

9.6.4 Obtaining a license file

Each ASC controller has an unique ASC ID. To obtain a license file you need to know this ASC ID and which packages you want to license.

> **TIP**
>
> Please note that the license file contains the license for one or more packages.

Requirement

- InScript is connected to the ASC controller in question.

Procedure

1. In the *Navigator* view, click on the ASC controller you want to get the ASC ID from.
2. In the *Node Properties* view, click **Connect Controllerservices**.
3. On the *Licensing* tab, click **Write to file** to save the ASC ID.
4. Send this file together with the package names you want to license to Novanta.

9.6.5 Installing a license file

Requirement

- You have received a license file from Novanta. This license file has the file name extension `LIC`.

> **TIP**
>
> Please note that the license file is only valid for an ASC controller with a specific ASC ID.
>
> Please note that the license file contains the license for one or more packages.

Procedure

1. In the *Navigator* view, click on the ASC controller you want to install the license file on.
2. In the *Node Properties* view, click **Connect Controllerservices**.
3. On the *Licensing* tab, click **...** (3 dots).
4. Browse to the license file and select it.
5. Click **Install** and wait until the license file is installed.
6. On the *General* tab in the *Firmware* group, click **Restart**.
7. Restart the InScript software.
8. Set up suitable devices for your laser and further devices now, as described in section 9.7 on page 34.

9.7 Setting Up A Laser And Further Devices

In order to be able of controlling devices via InScript, you have to create, configure, activate, parametrize these devices in InScript and save their configuration. In this chapter you will learn how to do this using the example of your laser. In either case you need a laser device to finish the setup of a basic system.

> **TIP**
>
> You can use this method not only to set up the laser device but to set up further devices later on. Follow the instruction manuals of the respective devices in addition to the following procedure.

Requirements

- The devices you want to set up must be connected to the controller.
- This controller must be connected to the InScript PC.
- InScript must be connected to the controller.

Procedure

1. In the *Navigator* view, double-click the controller you want to create a new device for.

   The node of the controller expands to nodes *Memory*, *Devices* and *Filesystem*.

2. Create a device:

   a) In node's *Devices* context menu, click **Create device**.

   The New Device window opens.

   b) Select a *Category*, e.g. Laser.

   c) Select a *Type*, e.g. your laser model or *ConfigurableLaser*.

   Find the description of supported devices in [3] in chapter *Devices*.

   d) In *Name*, you can rename the device.

   > **TIP**
   >
   > The device cannot be renamed later on.

   e) Click **OK**.

   The device will be created and appears as child of node *Devices*.

3. Configure the device:

   a) Expand node *Devices*.

   b) Select the device.

   The *Node Properties* window shows the device's configuration.

   c) In the *Node Properties* window, configure the device.

   Find the description of supported devices in [3] in chapter *Devices*.

> **TIP**
>
> You can only configure a device while it is inactive.

4. Activate the device:

> **NOTICE**
>
> Wrong configuration
>
> may cause material damage.
>
> a) Make sure the device is configured correctly.
>
> b) In the *Navigator* view in the device's context menu, click **Activate**.

5. Parametrize the device:

   a) Select the device.

      The *Node Properties* window shows the device's settings.

   b) In the *Node Properties* window, parametrize the device.

      Find the description of supported devices in [3] in chapter *Devices*.

6. Save the configuration:

> **NOTICE**
>
> Resetting the controller
>
> causes loss of configuration data.
>
> a) In the *Navigator* view, select the controller the device is subordinate to.
>
> b) In the controller's context menu, click **Save configuration**.

7. Correct the scan head's scan field distortion as described in 9.8 on page 37.

9.8 Correcting The Scan Head's Scan Field Distortion

Procedure

1. In *Navigator* view, open the context menu of the scan head's superordinate controller and click **Manage scan field correction**.

   The *Manage Scan Field Correction* wizard will be opened.

   While the wizard is open, several actions in InScript which modify the job are disabled, as e.g. JobStart, JobAbort, ClearAndLoad.
2. Follow the wizard's on-screen instructions.

   If you need a more detailed description or instructions then click **Info** in the wizard's window.
3. Optimize the scan head actuator dynamics as described in 9.9 on page 37.

9.9 Optimizing The Scan Head Actuator Dynamics

9.9.1 Prerequisites

Procedure

1. In the *Vector Editor* view, open the context menu (right-click in empty area).
2. Click **Layers**.
3. Activate **JobLines**, **CommendedScannerPositionXYZ** and **TrackedScannerPositionXYZ**.

   This shows the activated elements in the *Vector Editor* view. You will need this later.
4. In the *Navigator* view, expand **Pens** > **default (systempen)**.
5. Click **linepar**.
6. In the *linepar – Node Properties* window on the *Common* tab, set **Processing speed** to a value appropriate for your intended application.

7. In the *Navigator* view in the *default (systempen)*, open the context menu (right-click) of head.
8. In the context menu, click **Show in Inspector** > **New window**.

   The *head – Inspector* window opens.
9. In the *head – Inspector* window, expand **tiger** > **actuators**.

   The designation 'tiger' goes back to the first implementation of this software function, which was done for the TIGER scan head.

   > **TIP**
   >
   > Do not close the *head – Inspector* window.
10. Calibrate as described in 9.9.2 on page 38.

9.9.2 Calibration

9.9.2.1 Dynamics Parameters

In the actuators path there are sections for each actuator: x, y and z.

In each actuator's section there are 4 configuration variables we are interested in:

`usr.pens.default.head.tiger.actuators.[xyz].dynamics.position`

`usr.pens.default.head.tiger.actuators.[xyz].dynamics.velocity`

`usr.pens.default.head.tiger.actuators.[xyz].dynamics.acceleration`

`usr.pens.default.head.tiger.actuators.[xyz].groupDelay`

The dynamics parameters are used during the calculation of transitions between the end of one path segment and the start of the next.

These parameters and their effects will now be explained in more detail.

Position

The `usr.pens.default.head.tiger.actuators.[xyz].dynamics.position` variable defines the allowed positive and negative *position* range centered around zero, measured in the appropriate units for the actuator (mechanical degrees for rotary actuators, millimeters for linear actuators).

Procedure

- In `usr.pens.default.head.tiger.actuators.[xyz].dynamics.position`, enter the **position** values.

Velocity

The `usr.pens.default.head.tiger.actuators.[xyz].dynamics.velocity` variable defines the maximum positive and negative *velocity* for this actuator during any transitions between path segments. The value is specified in actuator units per second (mechanical degrees per second for rotary actuators, millimeters per second for linear actuators).

These values should be calculated based on the scan field size and the greatest Processing speed that will be used in the application; see also figure 9.1 on page 39.

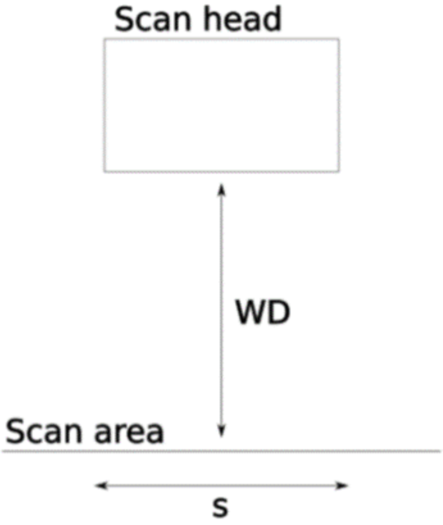


Figure 9.1: Scanning speed s and working distance WD

For a given scanning speed *s* [mm/s] and a working distance *WD* [mm], the angular actuator velocity *v* in degrees per second can be approximated for positions near the scan field center by this formula:

v = s ∗ 180 / (2 ∗ WD ∗ π)

where *v* corresponds to the *velocity* limit value in the X and Y scan head actuator dynamics parameters.

Conversely, the greatest allowed *Process speed s* can be calculated from the dynamics parameters velocity limit like this:

$s = v * (2 * WD * \pi) / 180$

If the velocity values are too low for the *Processing speed* used in a job, the system will issue an error message about dynamics parameters limits being exceeded.

Procedure

1. Calculate the *velocity* value for each actuator as described above.
2. In `usr.pens.default.head.tiger.actuators.[xyz].dynamics.velocity`, enter the **velocity** values.

Acceleration

The `usr.pens.default.head.tiger.actuators.[xyz].dynamics.acceleration` variable defines the maximum positive and negative *acceleration* for this actuator during any transitions between path segments. The value is specified in actuator units per second squared (mechanical degrees per second squared for rotary actuators, millimeters per second squared for linear actuators).

Lower *acceleration* values mean that the acceleration and deceleration parts of the transition trajectory become longer (both in size and in duration), and they may even exceed the position limits for this actuator. When that happens, the system will issue an error message about dynamics parameters limits being exceeded.

Higher *acceleration* values mean that the acceleration and deceleration parts of the transition trajectory become shorter (both in size and in duration), which leaves the scan head actuators less time to reach a steady state after a transition. When this happens there will be some kind of visible distortion to the start of the path segment after a transition.

Procedure

- In `usr.pens.default.head.tiger.actuators.[xyz].dynamics.acceleration`, enter the **acceleration** values.

Group Delay

The `usr.pens.default.head.tiger.actuators.[xyz].dynamics.groupDelay` variable corresponds to the delay between the position command signal and the measured position signal for this actuator, and it specifies the duration by which a path segment's first and last line are extended.

Increasing the *groupDelay* value gives the scan head more time to reach a steady state at the start of a path segment, and more time at the end of a path segment before the command signal for the actuators starts accelerating away from there. This will improve the position accuracy at the start of path segments, but it will also make the overall job cycle time longer.

Procedure

1. Get the Laser *Gate delay*:
   a) In the *Navigator* view, expand **Devices**.
   b) Click **[your laser device]**.
   c) On the *Osc* tab, read the **Gate delay** and note it down.
2. In `usr.pens.default.head.tiger.actuators.[xyz].dynamics.groupDelay`, enter the **groupDelay** value.

   These values should be equal or greater than the laser *Gate delay* value.

   The *groupDelay* value must be the same for all actuator channels that are used in the system. The option to use different values for different actuators is reserved for future extensions to the InScript software.

Example

The *groupDelay* value, multiplied by the *Processing speed* value, defines the length of the straight extension before the start of the first and after the end of the last line of each path segment. In this example, 300 microseconds times 6000 millimeters per second gives a length of 1.8 millimeters; see also figure 9.2 on page 42.

9.9.2.2 Threshold Angle For Joint 2

The *Dynamics Parameters* settings are used between the end of one path segment and the start of the next path segment when the two are not connected, or when the angular change of direction between two connected

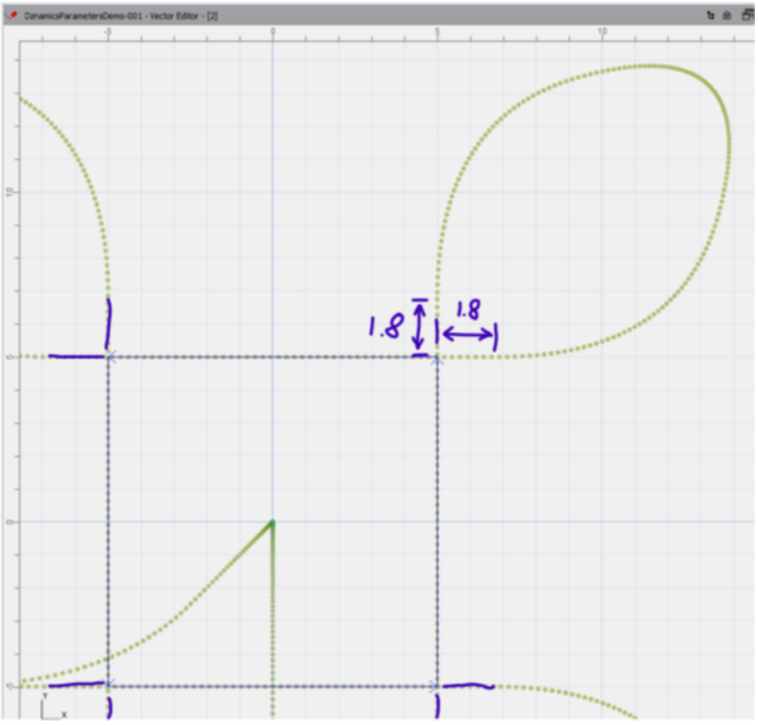


Figure 9.2: Length of straight extension (example)

lines is greater than a threshold angle.

In the *linepar* (line parameters) section of the InScript *default (systempen)* pen settings you find the *Min. angle for joint 2* parameter. This controls the threshold angle between two connected lines in a path. When the angular change of direction is greater than this threshold, the path will be interrupted, the laser will be switched off temporarily and a transition trajectory will be inserted between the lines.

Procedure

1. In the *Navigator* view, expand **Devices**.
2. Click **linepar**.
3. In the *linepar – Node Properties* window on the *Common* tab, set **Min. angle for joint 2** to a value that gives you the best balance between speed and accuracy.

   We recommend that you run a short InScript job which is representative of the intended application and look at the difference between the 'commanded' and the 'tracked' signals in the InScript Vector Editor view, repeatedly every time after adjusting the dynamics parameters, until you are satisfied with the balance between process cycle time and position accuracy.

The parameters that are most likely to have an impact on that balance are:

`usr.pens.default.linepar.common.angle2`

`usr.pens.default.head.tiger.actuators.[xyz].dynamics.acceleration`

`usr.pens.default.head.tiger.actuators.[xyz].groupDelay`

Example

Two connected lines with an angular change of direction of 47.7 degrees; see also figure 9.3 on page 43.

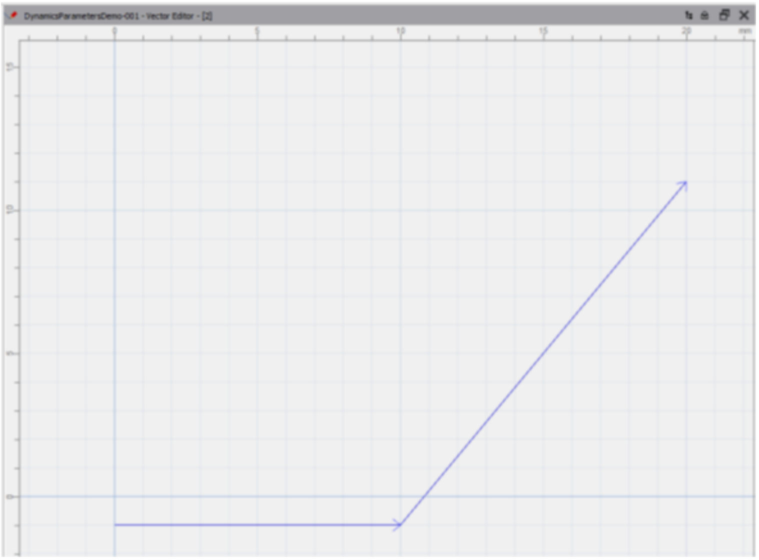


Figure 9.3: Angular change of direction (example)

With a threshold angle of 80 degrees and the change of direction less than the threshold angle, there is no transition inserted between the first and the second line; see also figure 9.4 on page 44.

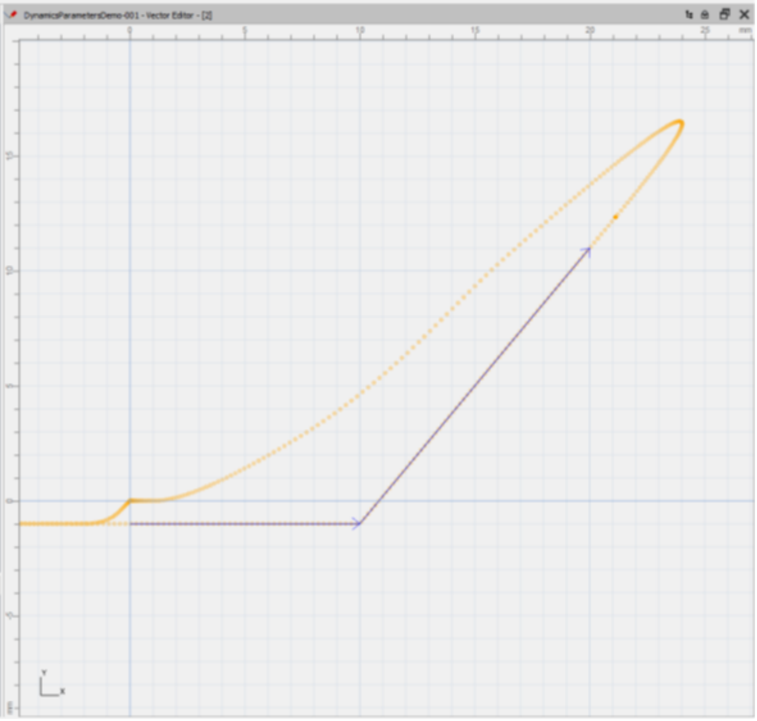


Figure 9.4: Angular change of direction less than threshold angle (example)

With a threshold angle of 30 degrees at slower *acceleration* values, the acceleration and deceleration parts of the transition between path segments take longer and require more space; see also figure 9.5 on page 45.

With a threshold angle of 30 degrees at faster *acceleration* values, the length and duration of the acceleration and deceleration parts of the trajectory are shorter, but the scan head may not have enough time to reach a steady state at the start of a path segment; see also figure 9.6 on page 46.

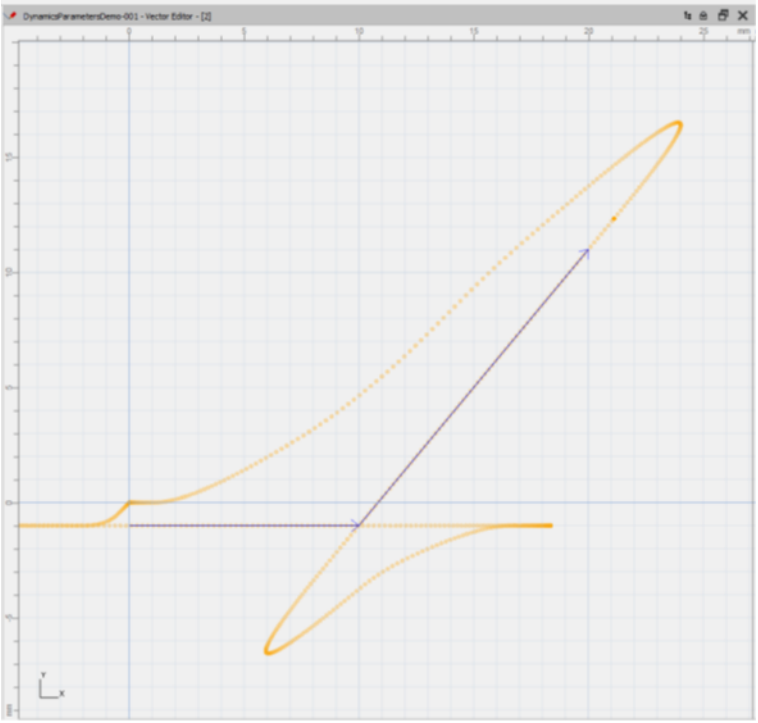


Figure 9.5: Angular change of direction greater than threshold angle (example)

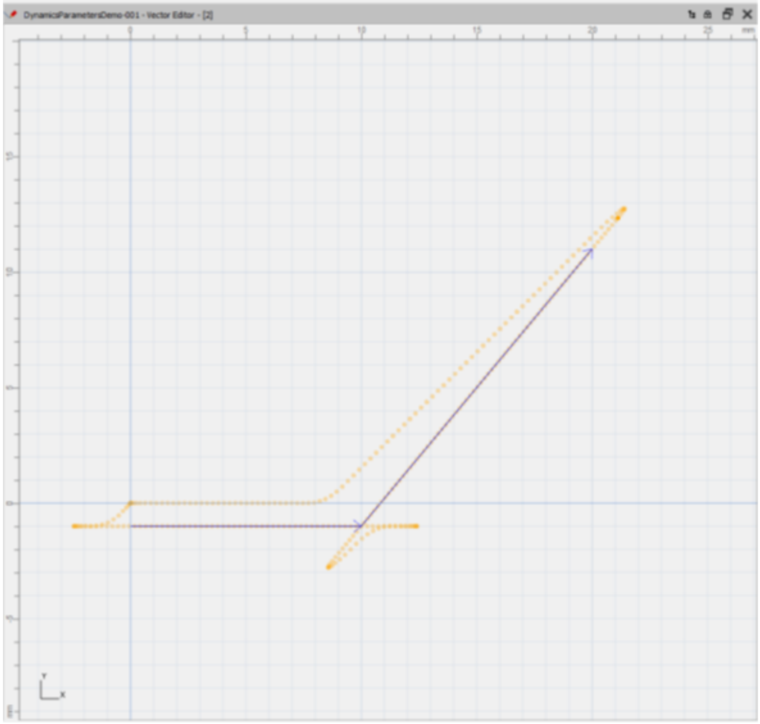


Figure 9.6: Angular change of direction greater than threshold angle at faster acceleration (example)

10 Working With Views

10.1 View Functions

All functions related to views are grouped in the *View* menu. Additionally each entry in the *View* menu has a corresponding button on the view toolbar. Whatever entry point you use – menu or toolbar – the underlying function is the same. That is why in the following only the menu function will be referenced.

The number of views in InScript is limited to 10. Because there is always a *Navigator* and a *Messages* view a maximum of 8 other views can be open. If the maximum number of views is reached then no further views can be opened and the menu items *Split vertical* and *horizontal* will be disabled.

10.2 Creating And Removing Views

10.2.1 Creating A Floating View

The views, that are mentioned below, can only be created floating. If you want them docked then dock them manually.

Procedure

- In the menu, click one of the following:
    - **View → Vector Editor**
    - **View → Node Properties**
    - **View → Inspector**
    - **View → UI File Container**

10.2.2 Creating A Job Control View

There can be only 1 *Job Control* view. The *Job Control* view is the only fixed view that can be closed and opened.

Procedure

- In the menu, click **View** → **Job Control**.

10.2.3 Creating An Inspector View On A Node

Procedure

1. In the *Navigator* view, select the node that shall be viewed in an *Inspector* view.
2. In the node's context menu, click **Show in Inspector** → **New window**.

10.2.4 Creating A Node Properties View On A Node

Procedure

1. In the *Navigator* view, select the node that shall be viewed in a *Node Properties* view.
2. In the node's context menu, click **Show in Node Properties** → **New window**.

10.2.5 Splitting An Existing View

This procedure only works with docked views. It will create a new docked view.

Procedure

1. Select the view that shall be split.
2. In the menu, either click **View** → **Split vertical** or **View** → **Split horizontal**.

10.2.6 Closing A View

The ”close” symbol may look different in different operating systems. Please refer to the documentation of your operating system on how the window ”close” symbol looks like in your system.

Procedure

1. Select the view that shall be closed by clicking its window title.
2. In the menu, click **View** → **Remove view**.

   –OR–

   In the view’s window title, click the ”close” symbol.

10.3 Arranging Views By Docking

Most InScript views can be arranged by a mechanism called docking. We distinguish the docking states *docked* and *floating*.

Docked views are views which are integrated in the InScript main window side by side.

Floating views act very similar to normal windows. In InScript floating views are always top level windows, i.e. they are always visible in the foreground of the application. Floating views are limited in the operability, i,e. toolbar function can not be used for floating views. That is by design and not a bug.

As indicated above, there are InScript views which cannot be arranged or change their docking state. They are called immobile or not dockable. These views are the *Navigator* view and the *Messages* view. They are of such importance that they are always visible in InScript.

Docking is typically performed by dragging and dropping a view. Dragging means to click the the view’s title and keeping the mouse button pressed while moving the object to its new position. While dragging the view InScript will temporarily arrange the other views and indicate the dropping position by a blue rectangle. Dropping means to release the mouse button when the view is on its new position.

10.3.1 Docking A View

Procedure

- Drag the view to its new position in the main window of InScript and drop it there.

  While dragging the view InScript will temporarily arrange the other views and indicate the dropping position by a blue rectangle.

10.3.2 Making A Docked View A Floating View

Procedure

- Drag the view outside the main window of InScript and drop it there.

  –OR–

  Double-click the view's title.

  > **TIP**
  >
  > If you have made a view floating by double-clicking its title and want it back to its former position, simply double-click its title again.

10.3.3 Preventing A Floating View From Docking

Procedure

- Press and hold the **Ctrl**-key while dragging the view to its new position over the main window of InScript.

10.4 Layout Functions

The arrangement of all views, including the floating ones, is called a layout. InScript has functions to save, restore and reset the layout.

10.4.1 Saving The Current Layout To A File

Procedure

- In the menu, click **View** → **Save layout**.

10.4.2 Loading A Layout From A File

Procedure

- In the menu, click **View** → **Load layout**.

10.4.3 Resetting The Layout To Default

Procedure

- In the menu, click **View** → **Reset layout**.

10.5 Locking And Unlocking A Docked View

By default a newly created view follows the context. This means that the view will visualize either the node context or the controller context, i.e. the selection in the *Navigator* view. If you navigate through the nodes on a controller in the *Navigator* view then the view will change each time you focus another node. This behavior can be switched off by *locking* the docked view.

Requirement

The view that shall be locked or unlocked must be docked; see also section 10.3.1 on page 50.

Procedure

- In the view's title, click the ”lock” button.

  –OR–

  In the view's context menu, click **View** → **Unlocked** or **View** → **Locked**.

10.6 Modifying The Content Of A Locked View

The content of the *Navigator* view and the *Messages* view cannot be modified, i.e. the *Navigator* view always shows controllers and their nodes and the

Messages view always shows messages. In all other InScript views you can modify the content respectively what the view shows.

Requirement

The view that shall be modified must be locked; see also section 10.5 on page 51.

Procedure

- Drag the node you want to visualize from the *Navigator* view and drop it on the locked view.

  –OR–

1. In the *Navigator* view, position the mouse pointer over the node that should be shown in the locked view.
2. In the context menu of the node, click **Show in Inspector** or **Show in Vector Editor**.
3. Click the entry with the title of the locked view.

10.7 Getting Information About A View

Each InScript view shows detailed information about itself as a tooltip.

Procedure

- To see the tooltip, position the cursor on the view's title and stay there for 2 seconds.

11 Managing Controllers

11.1 Adding A Controller

When running InScript for the first time you need to add a controller to InScript in order to assign it to InScript.

Procedure

1. In the menu, click **Controller** → **Add**.

   The window opens.

2. Enter the IP-address of your ARGES System Controller.

3. Click **OK**.

   The controller will be listed by its IP-address in the *Navigator* view.

   > **TIP**
   >
   > InScript is able to manage more than 1 controller. If you want to add more controllers then repeat the steps above.

4. If you want to use the added controller with InScript then connect it to InScript as described in section 11.2 on page 53.

11.2 Connecting To A Controller

Procedure

1. In the *Navigator* view, select the controller.

2. In the Menu, click **Controller** → **Connect**.

   If the connection can be established then the controller node automatically expands to nodes *Jobs*, *Pens*, *Fonts* and *Devices*.

InScript saves the state of the connected controller and automatically reconnects to the controller after restart.

> **TIP**
>
> If you want to connect to another controller then repeat the steps above.
>
> Prevent working on the wrong controller by disconnecting the controllers which you do not intend to work on. To do so, select the controller and click **Controller** → **Disconnect** in the main menu.

11.3 Disconnecting A Controller

Procedure

1. In the *Navigator* view, select the controller.
2. In the Menu, click **Controller** → **Disconnect**.

11.4 Removing A Controller

Requirement

- In the InScript software the controller has to be disconnected; see also section 11.3 on page 54.

Procedure

1. In the *Navigator* view, select the controller.
2. In the Menu, click **Controller** → **Remove**.

11.5 Starting A Job

Requirements

- In the InScript software the controller has to be connected; see also section 11.2 on page 53.

- The job which shall be executed has to be selected.

Procedure

> ⚠ **CAUTION**
>
> Visible and / or invisible laser radiation
>
> Hazard of eye or skin injury
>
> - Wear protective glasses that are appropriate for the laser in use.
> - Avoid eye or skin exposure to direct or scattered radiation.

- In the *Job Control* view, click the **Start job** button.

  –OR–

  In the *Job* toolbar, click the **Start job** button.

  –OR–

  In the main menu, click **Controller** → **Job start**.

11.6 Aborting A Job

Requirement

- In the InScript software the controller has to be connected; see also section 11.2 on page 53.
- The job which shall be aborted has to be selected and running.

Procedure

- In the *Job Control* view, click the **Abort job** button.

  –OR–

  In the *Job* toolbar, click the **Abort job** button.

  –OR–

  In the main menu, click **Controller** → **Job abort**.

11.7 Managing The Scan Field Correction

Requirement

- In the InScript software the controller has to be connected; see also section 11.2 on page 53.

Procedure

1. In the *Navigator* view, open the context menu of the scan head's superordinate controller and click **Manage scan field correction**.

   The *Manage Scan Field Correction* wizard will be opened.

   While the wizard is open, several actions in InScript which modify the job are disabled, as e.g. JobStart, JobAbort, ClearAndLoad.

2. Follow the wizard's on-screen instructions.

   If you need a more detailed description or instructions then click **Info** in the wizard's window.

11.8 Connecting/Disconnecting The Controllerservices

Requirement

- In the InScript software the controller has to be connected; see also section 11.2 on page 53.

Procedure

1. In the *Navigator* view, click the controller's IP address.
2. In the *Node Properties* view, click **Connect Controllerservices** or **Disconnect Controllerservices** respectively.

   The Controllerservices connect to the controller and the dialog is enabled; see also chapter 26 on page 168.

11.9 Saving The Configuration

Requirement

- In the InScript software the controller has to be connected; see also section 11.2 on page 53.

Procedure

1. In the *Navigator* view, select the controller.
2. In the Menu, click **Controller** → **Save configuration**.

   The configuration of the controller will be saved.

12 Managing Devices

12.1 What Are Devices?

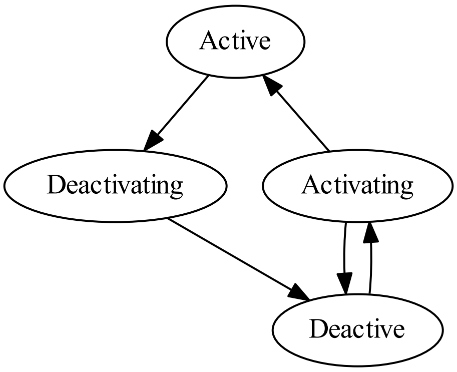
Devices are the software representation of hardware items which are on the controller or connected to the controller. The controller uses specific drivers to control these devices. Devices and their drivers can be accessed via the Devices subtree of each controller in the InScript *Navigator* view.

At the first time a controller will be connected there are already some devices and their drivers present in its Devices subtree. The presence of these devices depends on the controller hardware and the assumption that at least one scan head is installed. These devices cover basic functions of controller and scan head.

You may want to add further devices and their drivers respectively for the devices connected to the controller. At least one should be added for the laser. The following sections explain how this is done and how the devices are managed. Please keep in mind that almost all device function can be only executed when the controller is connected.

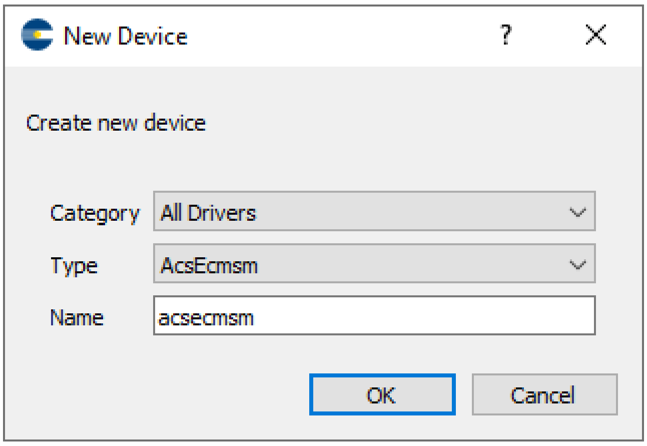
12.2 What Are Device States?

All devices have a state. If a device will be created it has the state ”Deactive”, which means it is not communicating with the underlaying hardware. If we activate a device (see section 12.5 on page 61) its state will change to ”Activating” and the communication with the underlaying hardware will be established. If this succeeds the state will change to ”Active” and the device is ready for use now. If the connection fails, the state will change back to ”Deactive”. If we deactivate an active device (see section 12.9 on page 62) its state will change to ”Deactivating” and the communication with the underlaying hardware will be closed. If this is done, the state will change to ”Deactive”.



12.3 Create A New Device

To create a new device on a certain controller select the controller in the InScript *Navigator* view by simply clicking on it. Expand the controller node, if it isn't. Then open the *Navigator* view context menu of the \Devices node by right clicking on it. Select the menu entry ”Create Device...” to open the ”New Device” dialog.



New Device

Create new device

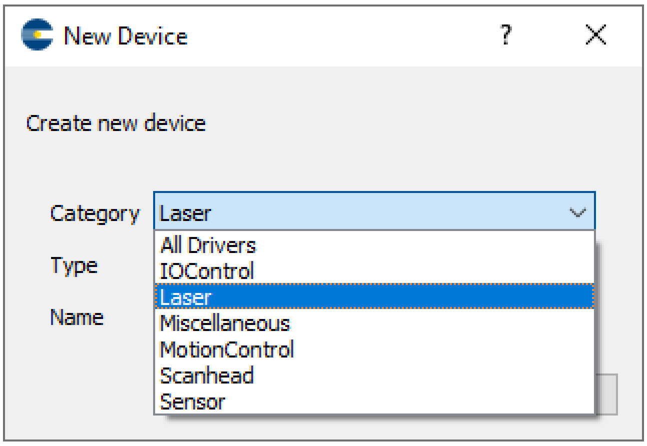
Category: All Drivers

Type: AcsEcmsm

Name: acsecmsm

Figure 12.1: New Device dialog

From the combo box Category select the category the device you want to create belongs to. If you are not sure which category your desired devices belongs to, choose ”All Drivers” and select from the complete list of all available devices. If the controller firmware does not support the categorization of devices, the choice ”All Drivers” is the only one which is available.



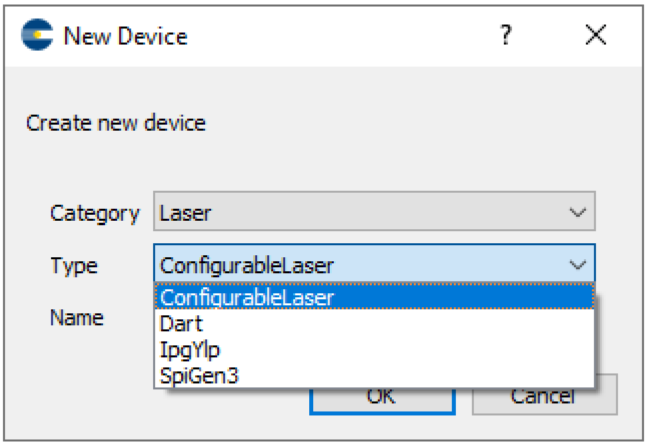
New Device

Create new device

Category Laser

Figure 12.2: New Device dialog - Categories

Then select the type of device you want to create from the Type combo box.



New Device

Create new device

Category Laser

Type ConfigurableLaser

Figure 12.3: New Device dialog - Laser types

By choosing the device type the dialog will suggest a device name which is suitable in most case, but you can change the name, the new device will get by using the Name input box. Please consider that a device name can be used for a certain driver type only once, otherwise the device creation will fail.

Finally click **OK** to create the new device.

12.4 Edit The Device Configuration

To change the device configuration open a Node Properties or an *Inspector* view on the device node you want to configure and change the variables according to your needs.

See section 10.2 on page 47 on how you can create a new *Node Properties* or *Inspector* view 10.6 on page 51 on how to show the device node in an existing *Node Properties* or *Inspector* view.

Please consider that the device configuration can be changed only, if the device is not activated yet. In other words – you can change the device configuration for deactive devices only.

12.5 Activate A Device

To activate a device open the *Navigator* view context menu of the device you want to activate by right clicking on it and select the menu entry ”Activate”. This menu entry is available only, if the device is deactivated.

Please consider that there are a few system devices which are active always, i.e. their device state can’t be changed.

12.6 Reset A Device

To reset a device open the *Navigator* view context menu of the device you want to reset by right clicking on it and select the menu entry ”Reset”. This menu entry is available only, if the device is activated.

12.7 Edit The Device Settings

To change the device settings open a *Node Properties* or an *Inspector* view on the device node you want to configure and change the variables according to your needs.

See section 10.2 on page 47 how to create a new *Node Properties* or *Inspector* view 10.6 on page 51 how to show the device node in an existing *Node Properties* or *Inspector* view.

Please consider that the device settings can be changed only, if the device is in the activated state. In other words – you can change the device settings for active devices only.

12.8 Set Device Parameters While Executing A Job

Procedure

- Use InScript's pen mechanism; see chapter 16 on page 80.

12.9 Deactivate A Device

To deactivate a device open the *Navigator* view context menu of the device you want to deactivate by right clicking on it and select the menu entry ”Deactivate”. This menu entry is available only, if the device is activated.

Please consider that there are a few system devices which are active always, i.e. their device state can't be changed.

12.10 Delete A Device

To delete a device open the *Navigator* view context menu of the device you want to delete by right clicking on it and select the menu entry ”Delete Device”. This menu entry is available only, if the device is deactivated, so possibly you have to deactivate the device you want to delete first.

12.11 Make The Device Configuration Permanent

To make your device configuration permanent (i.e. it will be restored after a controller restart) open the *Navigator* view context menu of the controller your device(s) belongs to by right clicking on it. From the context menu select the menu entry ”Save Configuration”. This stores the configuration of your devices to the flash memory of the controller.

13 Working With Files

13.1 What Can Be Done With Files?

Because of the fact that InScript supports multiple controllers, all file operations refer to the current selected controller in the *Navigator* view. This selection is also called *controller context*.

You can load several file types (jobs, pens, fonts, raster and vector graphics as well as distortions) with the InScript menu function **File** → **Load...**. The file open dialog of the operation system will be opened. Set the file filter to the desired file type, select the desired file and close the dialog with the **OK** button.

When a file will be *loaded* in InScript, in fact it will be read from the local file system and transfered to the memory of the controller so that it is ready for execution.

You can see most of all loaded files of a controller somewhere in its subtree. Please consider the name of the node in the controller memory may differ from the loaded file name, because the node name is stored in the file too.

The context menu of the *Navigator* view offers file functions for several node types too. E.g. the **Pens** node has a **Load Pen...** menu entry and each node below or rather each pen node has a **Save Pen...** menu entry.

The following file functions are available in InScript:

- Load Job
- Save Job
- Load Subtree
- Save Subtree
- Load Pen

- Save Pen
- Load Font
- Load Vectorgraphics
- Save Vectorgraphics
- Load Distortion
- Save Distortion
- Load Layout
- Save Layout

Although InScript runs on different platforms (Windows, Mac) and line endings differ on these platforms, all native InScript file types (jobs, subtrees, pens and distortions) are platform independent too. I.e. you can write each of these file types on one platform and read it on another platform without problem.

13.2 Handling Of Vector Graphics

Vector graphics files can be loaded into RawLines nodes and saved from them. These function can be reached in the *Navigator* view context menu of a RawLines node. Currently InScript supports the following vector graphics file formats:

- ARL – ARGES RawLines (Read/Write)
- SVG – Scalable Vector Graphics (Read/Write)
- DXF – Autodesk Drawing Interchange File Format (Read only)
- HPGL (PLT) – Hewlett-Packard Graphics Language (Plotter) files (Read only)
- IGES – Initial Graphics Exchange Specification files (Read only)

All kind of graphics in the SVG, DXF and IGES vector graphics file which are not primitive (dots and lines) as cirles, ellipse, arcs and curves will be approximated during the read process by a sequence of lines. The maximum deviation of these lines from the original element can be controlled by a parameter called *flatness* which can be set in the *Vector Editor* view preferences; see section 25.3.5 on page 156;

13.2.1 ARL

The ARL format is a very efficient vector graphics file format specified by ARGES. It is the recommended file format for 2D and 3D vector data in InScript. If you want to create ARL files with your own software please contact our service department to get a file format description.

13.2.2 SVG

The SVG file format is a widely used 2D vector graphics file format. The InScript SVG support is based on version 1.1, Second Edition, W3C Recommendation 16 August 2011; see

http://www.w3.org/TR/2011/REC-SVG11-20110816

By importing a SVG file InScript reads the outlines of objects only. Other properties of objects like the stroke width, the stroke color and the fill color will be ignored. The following SVG elements are supported:

- Polyline
- Polygon
- Rect
- Line
- Circle
- Ellipse
- Path
- G (group)
- A (link – the link itself will be ignored, but its subtree will be respected)

Each of these elements may have the *transform* attribute specified. Additional all specified SVG unit lengths are supported, except **%**.

Explicitly not supported are the following SVG elements:

- Text
- Defs
- Use

If your SVG file contains text elements, you should convert these elements to paths with a suitable external SVG editor like Inkscape, because InScript can import path elements very well.

The InScript SVG export always uses a 100x100 mm² symmetric viewport. This way a re-import will be placed at the position it was exported from.

13.2.3 DXF

Autodesk Drawing Interchange File Format also is a widely used vector graphics file format. For more information see

https://en.wikipedia.org/wiki/AutoCAD_DXF

InScript supports the following DXF entities:

- Block
- Arc
- Circle
- Ellipse
- Insert
- Line
- Lwpolyline
- Point
- Polyline
- Spline
- Vertex

Explicitly not supported are all entities of the following type:

- Text
- Dimension
- Faces
- Solids

If your DXF file contains text elements, you should convert these elements to curves with a suitable external DXF editor or CAD application because InScript can import curves very well.

13.2.4 HPGL

The Hewlett-Packard Graphics Language (HP-GL or HPGL) is a printer control language. It was created by Hewlett-Packard as a control language for HP plotters. Because of it's convenience it is used as a data exchange format for 2D vector graphics very often. For more information see

https://en.wikipedia.org/wiki/HP-GL

InScript supports the following HPGL commands respectively instructions:

- SP - Select pen
- PU - Pen up
- PD - Pen down
- PA - Position absolute
- PR - Position relative

Although this is a really very small subset of the language it is sufficient in most cases.

13.2.5 IGES

IGES is a vendor-neutral, widely used 3D vector graphics file format, especially designed for the data exchange between different computer-aided design (CAD) systems. For more information see

https://en.wikipedia.org/wiki/Initial_Graphics_Exchange_Specification

InScript is able to read the most important parts which describe geometry data from an IGES file. The following IGES elements are supported:

- Point
- Line
- Copious data,
  - Form 1-3: Data points,

  - Form 11-13: Linear path
  - Form 63: Simple closed planar curve
- Circular arc
- Conic arc
  - Form 1: Ellipse
  - Form 2: Hyperbola
  - Form 3: Parabola
- B-Spline curve
- Transformation matrix

13.3 Particularities Of InScript XML Job Files

Newer firmware versions (usually fw3) support the InScript XML job file format. This format has the advantage that it may store not only the job, but also its dependencies (pens) in one single file.

This function can be accessed by opening the **Save Job** context menu entry of the *Navigator* view. If the feature is available, the Save File dialog will be extended by a check box widget with the caption *Include dependencies* in the bottom left corner of the dialog. If the check box is activated, the dependencies will be stored together with the job.

13.4 Where Does InScript Store Files?

Generally there are three places InScript reads files from or saves files to - the **program path**, the **user data path** and the **application data path**. InScript stores and loads data to or from all these places basically always in the subdirectory \ARGES\InScript3.

The InScript program path (on Microsoft Windows 7 e.g. `C:\Program Files (x86)\ARGES\InScript3` or `C:\Program Files\ARGES\InScript3`) contains all files required for the program execution. These includes binary files, message files, translation files and additional program icons.

The actual application data of InScript can be stored **user oriented** or **system oriented**. User oriented means that InScript reads all data from the user data path, i.e. InScript uses an own data path for each user who logs on to the operation system. In contrast to that a common data path - the application data path – will be used for all users by the system oriented data storage.

Both the user data path and the application data path are operation system specific, i.e. InScript uses the storage locations that are designated for that by the operation system. On Windows 7 the user data path e.g. is usually `C:\Users\Username\Documents\ARGES\InScript3` and the application data path is `C:\ProgramData\ARGES\InScript3`. These storage locations may differ significantly even inside an operating system family, like Windows XP, Vista and 7. For that reason there is a third kind of data storage in InScript - the **custom defined** data storage. In those mode an arbitrary path can be defined as the root path for all InScript data, which will be used by all users too.

The kind of data storage that InScript uses is defined during the installation of the application (check box **Install for**, selection **All users** or **The logged on user**) or can be set in the file `\Config\GlobalSettings.ini` in the application data path.

In the file `GlobalSettings.ini` the data storage model will be set by the **BasePathMode** key in the `[Preferences]` section which provides the following settings:

Table 13.1: Data Storage

Value	Description
0	User oriented data storage
1	System oriented data storage
2	Custom defined data storage: The name of the path to use will be read from the entry ”CustomBasePath” in the same section. If this path will not be found when InScript starts, the user data path will be used automatically instead of.

By default InScript uses the user oriented data storage model, i.e. all data in InScript will be stored user oriented in the user data path.

To make it easier for the user to realize the confusing variety of storage locations used by different operation systems and to provide an easy and fast access to the InScript data there is the **File → Explore documents...** menu entry in the InScript menu. This menu entry leads always (regardless of the chosen data storage model) to the path where InScript expects its documents. By using user oriented data storage this menu entry points to the user data path, on using system oriented data storage to the application data path and by using custom defined data storage to the path which is set by the **CustomBasePath** entry in the `[Preferences]` section of the `GlobalSettings.ini` file (if it exists, see above).

When the user data path and the application data path are not equal, an additional menu entry **File → Explore application data...** will be shown in the InScript menu. This entry serves primarily for the more comfortable access to the file `GlobalOptions.ini`.

InScript will not touch the file system in any way if the data storage model will be changed in the `GlobalSettings.ini`file (or with the preferences dialog later on). I.e. the directory structure will not be changed and the data inside it will not be moved to the new storage location automatically.

By default InScript uses dedicated directories for all file types (distortion, font, job, layout, pen, raster graphic and vector graphic files). These directories are called *default paths* and will be created by the InScript setup. The latter installs several example files of each file type there too.

Because some users may be accustomed to an operation method where the software remembers the last used path InScript can be set up to work so. For more information see section 25.8 on page 161 (Preferences) and the respective sections in chapter 31 on page 205.

13.5 Auto save

InScript supports a very useful feature called ”Auto save” i.e. job files may be automatically be stored. It can be configured in a broad range by the preferences. For more information see section 25.8 on page 161 (Preferences) and the respective sections in chapter 31 on page 205.

14 Working With Jobs

14.1 Requirement

- You need a controller, that is connected to InScript, to work with jobs; see also sections 11.1 on page 53 and 11.2 on page 53.

14.2 Creating A Job

Procedure

1. In the *Navigator* view, select the controller on which the job shall be created.
2. In the menu, click the symbol.

   –OR–

   In the Navigator view in the controller's *Jobs* subtree, open the context menu and click **Create job**.

   The *New Job* window opens.
3. Type in a name for the new job.
4. Click **OK**.

14.3 Creating Nodes In A Job

Use the *Create* toolbar of the *Navigator* view to create job nodes. This toolbar contains six buttons. Each button stands for a certain group of jobnodes - *Geometrical Operators*, *(Graphical) Tools*, *Organizers*, *Modifiers*, *Dialogs* and *IO's*.

If the icon of the job node type you want to create is the one on the button simply click the button to create the node. Otherwise click on the small black triangle on the right side of the button. The button will drop down a list of available node types in the appropriate group. Click on the desired node type in the list to create the node. This will select the clicked node type as default for the button too.

Usually the new created job node will be selected in the *Navigator* view automatically. This behaviour can be prevented by pressing the **Alt** key and hold it down while clicking on a button on the *Create* toolbar.

> **TIP**
>
> Find the job nodes described in detail in [3].

14.4 Renaming Nodes

Procedure

1. In the *Navigator* view, open the context menu for the node which should be renamed.
2. Click **Rename**.

14.5 Modifying The Order Of Nodes

- In the *Navigator* view in the node's context menu, click **Make first**, **Move up**, **Move down** or **Make last**.

  –OR–

  Drag the node that shall be ordered and drop it at its new position.

  While dragging the node the allowed positions are indicated either by a line – the node will be put between the nodes – or by a box around a node – the node will be put as child of this node.

  > **TIP**
  >
  > If a Navigator filter plugin is loaded then Drag and Drop is disabled.

–OR–

In the node's context menu, click **Cut**, browse to the node where the cut node shall be inserted and **Paste** it the same way there.

In the context menu the *Paste* entry is only enabled at allowed positions. The node will be inserted as last child and has to be ordered manually if required.

14.6 Clipboard Functions

Like many other applications, also InScript offers clipboard functions. They can also be used to modify jobs. All clipboard functions are accessible through the InScript main menu and the *Navigator* view context menu. The functions in the main menu use the widely-used hot keys (**Ctrl+X** for *cut*, **Ctrl+C** for *copy* and **Ctrl+V** for *paste*).

To cut a node and its complete subtree select the node you want to cut by clicking on it in the *Navigator* view. Then press **Ctrl+X** on the keyboard (or use the main menu entry **Edit** → **Cut** or use the context menu function **Cut**)

To copy a node and its complete subtree select the node you want to copy by clicking on it in the *Navigator* view. Then press **Ctrl+C** on the keyboard.

To paste the clipboard contents select the node where you want to insert to in the *Navigator* view. Then press Ctrl+V on the keyboard. The paste function will be enabled only, if the clipboard contains job node or subtree data.

The clipboard functions do work across different controllers too, i.e. you can copy a subtree from one controller and paste is to another one.

Some special tasks do require the name or the path of a node. For that purpose the Navigator view context menu offers the functions ”Nodename To Clipboard” and ”Nodepath To Clipboard”. If you need the node path somewhere open the context menu for this node by right clicking on it and select the ”Nodepath To Clipboard”. Now the clipboard contains the complete path of the node. You can paste it from the clipboard now wherever you need it.

14.7 Saving A Job As A File

Procedure

1. In the *Navigator* view, select the job which shall be saved.
2. In the job's context menu, click **Save job as**.

   The *Save Job File* window opens.
3. Browse where to save the job file and enter a *File name*.
4. Click **Save**.

14.8 Loading A Job

Procedure

1. In the *Navigator* view, select the controller to which the job shall be loaded.
2. In the menu, click **File** → **Load file**.

   –OR–

   In the Navigator view in the controller's *Jobs* subtree, open the context menu and click **Load file**.

   The *Load File* window opens.
3. Browse to the job file that shall be loaded.

   Job files have the file name extension `jobx`.
4. Click **Open**.

   > **TIP**
   >
   > If you load a job with a pen section for a device that is not activated then you will get a warning message and the pen section will have no control over the device even if you create and activate the device later. In this case:
   >
   > a) Create and activate the respective device.

> b) Reload the job.

14.9 Saving A Job As A Subtree

You can create a library of often used and complex tree structures by saving these to so called job subtrees and loading them when needed.

Procedure

1. In the *Navigator* view, select the job which shall be saved as a subtree.

   > **TIP**
   >
   > Only the whole job can be saved as subtree, thus the node has to be selected.

2. In the job's context menu, click **Save subtree**.

   The *Save Subtree File* window opens.

3. Browse where to save the subtree file and enter a *File name*.
4. Click **Save**.

14.10 Loading A Subtree

You can create a library of often used and complex tree structures by saving these to so called job subtrees and loading them when needed.

Procedure

1. In the *Navigator* view, select the job node where the subtree shall be appended.
2. In the job's context menu, click **Load subtree**.

   > **TIP**
   >
   > Only job nodes where it is appropriate to append a subtree show the *Load subtree* entry.

   The *Load Subtree File* window opens.

3. Browse to the subtree file that shall be loaded.

   Subtree files have the file name extension `tre`.

4. Click **Open**.

14.11 Starting / Aborting A Job

Procedure

1. If the *Job Control* view is not already open then open it by clicking **View** → **Job Control** in the menu.

   The *Job Control* view has 2 main buttons for job control that are similar to buttons on a media player. The ”Start job” button displays a triangle and the ”Abort job” button displays a square.

2. In the *Job Control* view, select the job that shall be executed from the *Selected job* list.

   –OR–

   In the *Navigator* view in the job’s context menu, click **Select**.

   In the *Job Control* view the ”Start job” button is active now and in the *Navigator* view in column *State* the node is marked with a ”laser beam” symbol. This indicates that the job can be executed now.

   > **⚠ CAUTION**
   >
   > Visible and / or invisible laser radiation
   >
   > Hazard of eye or skin injury
   >
   > - Wear protective glasses that are appropriate for the laser in use.
   > - Avoid eye or skin exposure to direct or scattered radiation.

3. In the *Job Control* view, click the ”Start job” button.

4. To abort job execution click the ”Abort job” button.

14.12 Deleting Nodes

Procedure

1. In the *Navigator* view, open the context menu for the node which should be deleted.
2. Click **Delete**.

15 Working With Variables

In the normal case variables can be modified in the *Node Properties* view.

The *Node Properties* view shows a reasonable subset of variables that are available in a node. To access all variables of a node use the *Inspector* view on this node.

Find useful reading in conjunction with variables in sections 10.2.3 on page 48 and 10.6 on page 51.

Procedure

Example: You want to modify the variable `usr.job.Job.Rectangle..x2`.

1. Open an *Inspector* view on the *usr.job.Job.Rectangle* node.
2. In the *x2* row, double-click the *Value* column.
3. Edit the variable.

   For most variable *Types* the variable editing works inplace, i.e. directly inside the table cell. Only the variable types *VAR:TEXT* and *VAR:BIN* do not work with inplace editing. If you start editing these variable types then an editor opens. Edit the variable value in this editor and click **OK** to resume its value.

> **TIP**
>
> Variables can also be accessed by scripting; see the *Firmware 3 User Manual* chapter *Job nodes: Organizing* section *Script Functions*.

16 Working With Pens

16.1 What Are Pens?

The InScript pen concept is similar to that well known from drawing programs. In a drawing program e.g. the line color, width and style can be defined for a pen. In InScript e.g. the laser power, frequency and precessing speed can be defined for a pen instead.

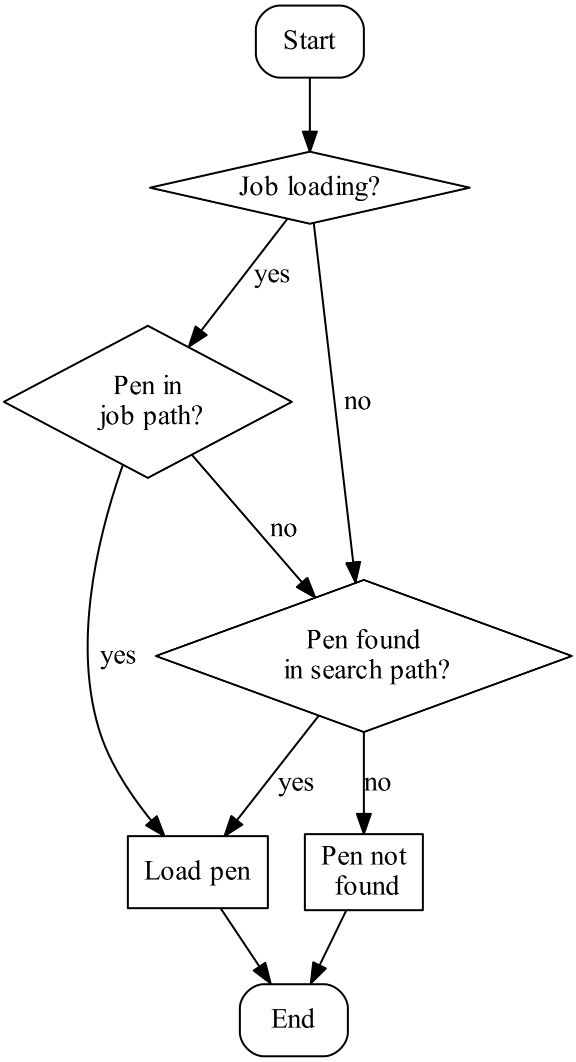
Pens are made up of pen sections. Each pen section contains the settings of the corresponding activated device. A pen section only contains device parameters that can be changed during a job.

Pens are self-contained files located in the *Pens* branch of the tree structure in the *Navigator* view.

16.2 Search Strategy

Where a pen is searched depends on whether the job is currently loading or not. If the job is currently loading then the controller firmware asks InScript which directory the job is loading from and tries to load the pen from there. If the controller firmware does not find the pen in that directory and all other directories given in the search path then the controller firmware will ask the master client again when the job is started. In that case InScript does not know where the job was loaded from and therefor only searches the normal search path. The pen will be taken from the first directory in which it is found; see also figure 16.2 on page 80.

In the *Preferences* a search path can be defined and reordered; see section 25.7 on page 160.



Pen search strategy

16.3 Managing Pens

16.3.1 Creating A Pen

Procedure

1. In the *Navigator* view, open the context menu of *Pens*.
2. Click **Create Pen**.

   The *Create New Pen* window opens.
3. Enter a *Pen name*.

   > **TIP**
   >
   > Pen names are limited to 31 characters. A pen name should be meaningful regarding the parameter(s) set by the pen. Later on when applying a pen within a job this will help to select a specific pen in the case that you created many pens.

4. Click **OK**.

16.3.2 Creating A Pen Section

Requirement

- The pen exists for which the pen section shall be created; see section 16.3.1 on page 81.

Procedure

1. In the *Navigator* view in the *Pens* subtree, open the context menu of the pen for which the pen section shall be added.

   > **TIP**
   >
   > Pen sections cannot be created for the *default* pen. The *default* pen automatically contains pen sections for active devices.

2. Click **Create pen section**.

   The *New Pen Section* window opens.
3. Select the pen section *Type*.

   The offered list depends on which *Devices* are activated.
4. Click **OK**.

Pen sections will be shown alphabetically sorted always in the Navigator.

16.3.3 Adding A Variable To A Pen Section

Device variables can be added to a pen section using the *Inspector* view. Variables that can be added to a pen section can be identified by having the P flag in the *Flags* column of the *Inspector* view.

Sets of variables (type VAR:SET) have the P flag but they cannot be added to a pen. Here the P flag indicates that the set contains at least 1 variable that can be added to a pen section.

Requirements

- The pen already exists; see section 16.3.1 on page 81.
- The pen already has a pen section corresponding to the variable which shall be added to a pen section; see section 16.3.2 on page 82.

Procedure

1. In the *Navigator* view, select the controller.
2. Open the *Inspector* view.
3. In the *Inspector* view, browse to the variable you want to add to the pen section; e.g. `dev.linepar.common.speed_m`.
4. Open the context menu of the variable.
5. Click **Add to pen** → <**pen name**>.

   > **TIP**
   >
   > If the pen section already has that variable then adding it again has no effect.

6. Check if the variable was added to the pen section by browsing to the pen section in the *Navigator* view.

   The variable is either visible at once or contained in a variable set (VAR:SET) in the *Inspector* view.

16.3.4 Saving A Pen

Procedure

1. In the *Navigator* view, select the pen to save.
2. Open its context menu and click **Save pen**.

   The *Save Pen File* window opens.
3. Browse where to save the pen file and enter a *File name*.
4. Click **Save**.

16.3.5 Loading A Pen

Procedure

1. In the *Navigator* view, select the *Pens* subtree.
2. Open its context menu and click **Load pen**.

   The *Load Pen File* window opens.
3. Browse to the pen file that shall be loaded.

   Pen files have the file name extensions `penx` and `pen`.
4. Click **Open**.

   > **TIP**
   >
   > If you load a pen with a pen section for a device that is not activated then you will get a warning message and the pen section will have no control over the device even if you create and activate the device later. In this case:
   >
   > a) Create and activate the respective device.
   >
   > b) Reload the pen.

16.3.6 Using A Pen Within A Job

Procedure

1. In the *Navigator* view, expand the job where you want to use the pen.
2. Add an organizing **Use pen** node to the job.
3. If necessary drag and drop the node to its intended position within the job subtree.

4. In the *Node Properties* view, enter the *Pen name* you want to use.

> **TIP**
>
> Note that in some cases it may take essential time for a device to reach the set parameter value, e.g. when a translation stage moves into position.

Alternative Procedure

1. In the *Navigator* view, expand the job where you want to use the pen.
2. Drag the pen, that you want to use, from the pens subtree to its intended position within the job subtree and drop it there.

   This creates a new **Use pen** node at the drop position and sets the pen name property of this node to the name of the dragged pen.

   If you drag and drop a pen from the pens subtree to an existing **Use pen** node then only the pen name of the drop target node will be set to the name of the dragged pen.

16.3.7 Deleting A Pen Variable

Procedure

1. In the *Navigator* view, browse to the pen section from which the pen variable shall be deleted and select it.
2. In the *Inspector* view, select the pen variable.

   The variable is either visible at once or contained in a variable set (VAR:SET).
3. Open its context menu and click **Delete node**

> **TIP**
>
> When deleting the last variable in a variable set (VAR:SET) then the variable set ”folder” remains.
>
> - Delete this ”folder” in the *Navigator* view.

16.3.8 Deleting A Pen Section

Procedure

1. In the *Navigator*, browse to the pen section which shall be deleted.
2. Open its context menu and click **Delete node**.

16.3.9 Deleting A Pen

Procedure

1. In the *Navigator*, browse to the pen which shall be deleted.
2. Open its context menu and click **Delete node**.

17 Working With Fonts

17.1 Supported File Formats

InScript supports TrueType fonts (TTF) as well as the ARGES-specific font format FDT.

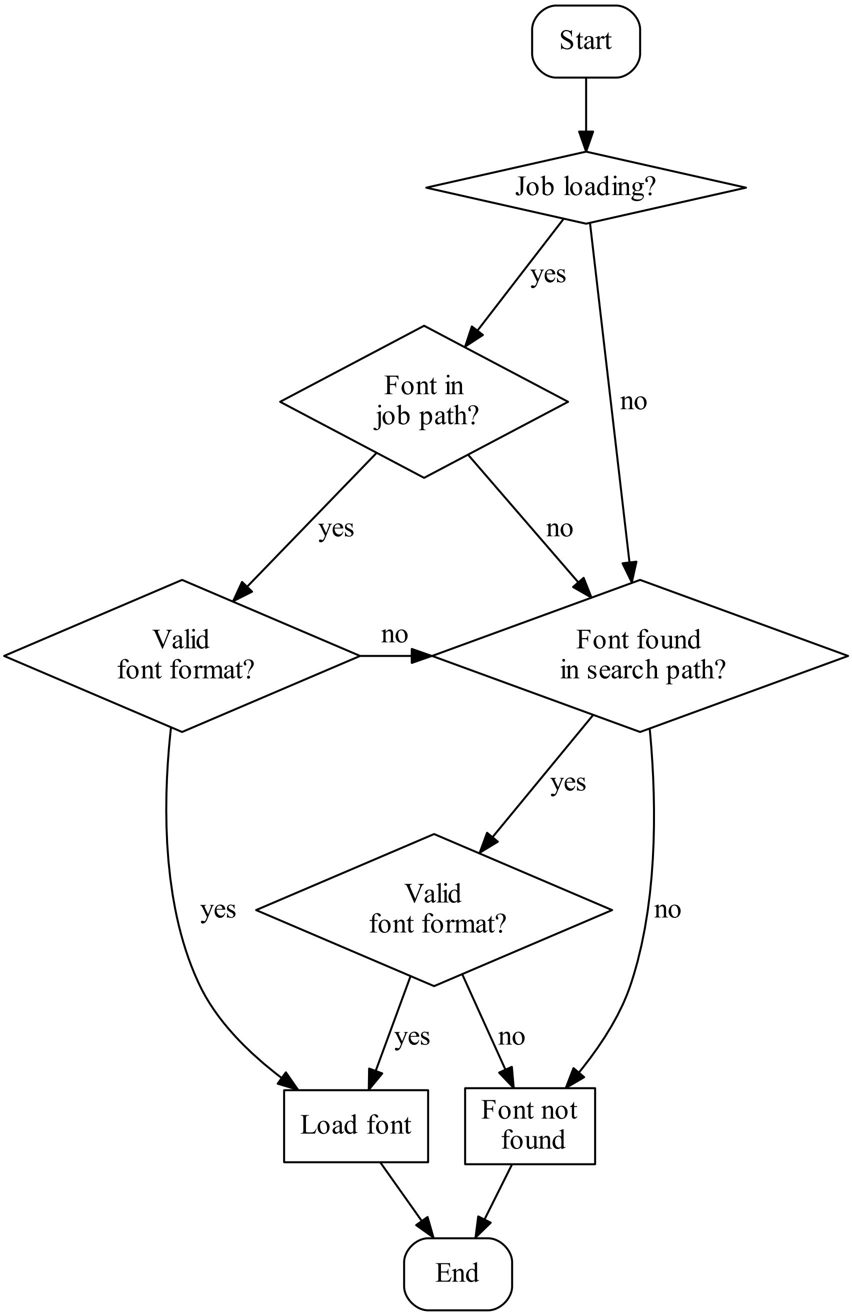
17.2 Search Strategy

> **TIP**
>
> If you install a font during the runtime of InScript then restart InScript. Otherwise InScript will not find the font.

Where a font is searched depends on whether a job is currently loading or not. If the job is currently loading then the controller firmware asks InScript which directory the job is loading from and tries to load the font from there. If the controller firmware does not find the font in that directory and all other directories given in the search path then the controller firmware will ask the master client again when the job is started. In that case InScript does not know where the job was loaded from and therefor only searches the normal search path. The font will be taken from the first directory in which it is found; see also figure 17.2 on page 87.

If the same font is as FDT and TTF in that directory then the FDT font will be loaded.

In the *Preferences* search paths can be defined and put in order; see section 25.6 on page 159.



Font search strategy

18 Working With The Navigator View

The InScript Navigator is one of the most important parts of InScript. It shows a tree of certain objects. On the top level of the tree controllers are shown always. If a controller is connected it shows further subnodes. On the second level of the tree there are the grouping nodes ”Jobs”, ”Pens”, ”Fonts” and ”Devices”.

Several function blocks of the InScript *Navigator* view were already described in chapters:

- 12 on page 58 (Managing Devices)
- 14 on page 72 (Working With Jobs)
- 15 on page 79 (Working With Variables)
- 16 on page 80 (Working With Pens)

18.1 Navigator View Navigation

Use the following keys to navigate inside the *Navigator* view, respectively to change the current selected node inside the tree:

Table 18.1: Navigator Keys

Key	Description

Continued on next page

Table 18.1 – concluded from previous page

Key	Description
Down arrow key	Navigates to the node below the current one (Makes the node below the current one the new current node.)
Up arrow key	Navigates to the node above the current one (Makes the node above the current one the new current node.)
Right arrow key	Expands the current node
Left arrow key	Collapses the current node

18.2 Navigator View Inplace Editing

The *Navigator* view provides the ability to change a node name directly in the tree without opening a dedicated editor window. This is called ”inplace editing”. Inplace editing basically will be started by clicking on an already selected item in InScript.

To start the inplace editing of a node name in the *Navigator* view, select the node you want to change by simply clicking on it or using the arrow keys (see paragraph above). Doing so causes the node to become the current node of the tree. Then click the node again to start inplace editing. If the node you want to start inplace editing for is already the current one, then simply click on it.

18.3 The Navigator View Context Menu

The Navigator provides a context menu to execute several actions on a node or a controller or the controller itself.

18.3.1 View Actions

Table 18.2: View Actions

Action	Description
Show in Vector Editor	Shows the current node in one of the already existing *Vector Editor* views or a new view (Select by sub menu)
Show in Node Properties	Shows the current node in one of the already existing *Node Properties* views or a new view (Select by sub menu)
Show in Inspector	Shows the current node in one of the already existing *Inspector* views or a new view. (Select by sub menu)

See also chapter 10 on page 47.

18.3.2 Controller Actions

Table 18.3: Controller Actions

Action	Description
Add	Adds a new controller to the list
Remove	Removes the selected controller from the list
Connect	Connects the selected controller
Disconnect	Disconnects the selected controller
Manage scan field correction	Shows the scan field correction wizard for the selected controller
View scan field correction	Shows the scan field correction for the selected controller

Continued on next page

Table 18.3 – concluded from previous page

Action	Description
Controllerservices	Shows the controllerservices for a controller in the *Node Properties* view when the controller is selected
Save configuration	Saves the device configuration and the default pen to the controller

18.3.3 Job Node Actions

Table 18.4: Controller Actions

Action	Description
Execution mode	Modifies the ”Execution mode” of a job node directly

18.3.4 Device Actions

Table 18.5: Device Actions

Action	Description
View	Depending on its configuration, a certain controller may offer different views for its devices. The *View* menu entry provides a sub menu with an entry for each available device view. Use this to change the layout of the device nodes according to various aspects

Continued on next page

Table 18.5 – concluded from previous page

Action	Description
Create device	Creates a new device
Activate	Activates the current device
Deactivate	Deactivates the current device
Reset	Resets the current device
Delete	Deletes the current device

See also chapter 12 on page 58.

18.3.5 Node Actions

Table 18.6: Node Actions

Action	Description
Rename	Renames a node. Starts inplace editing of the name column
Select / Deselect	Selects / deselects the current job node. Toggles the selection state of a job node
Delete	Deletes the current node
Create pen	Creates a new pen
Create pen section	Creates a new pen section

18.3.6 File Actions

Table 18.7: File Actions

Action	Description
Load job	Selects a job file and loads it

Continued on next page

Table 18.7 – concluded from previous page

Action	Description
Save job	Saves a job into a file
Load subtree	Selects a subtree file and loads it
Save subtree	Saves a subtree into a file
Load pen	Selects a pen file and loads it
Save pen	Saves a pen into a file
Load font	Selects a font file and loads it

18.3.7 Clipboard Actions

Table 18.8: Clipboard Actions

Action	Description
Cut	Copies the current node and its subtree to the clipboard and deletes them afterwards
Copy	Copies the current node and its subtree to the clipboard
Paste	Pastes the content of the clipboard to the current node if it has an appropriate format
Nodename to clipboard	Copies the name of the current node to the clipboard
Nodepath to clipboard	Copies the path of the current node to the clipboard

18.3.8 Treeview Actions

Table 18.9: Treeview Actions

Action	Description

Continued on next page

Table 18.9 – concluded from previous page

Action	Description
Expand node	Expands the current node
Expand subtree	Expands the current subtree
Collapse node	Collapses the current node
Collapse subtree	Collapses the current subtree

18.3.9 Arrange Actions

Table 18.10: Arrange Actions

Action	Description
Make first	Make the current node the first one
Move up	Move the current node up one step
Move down	Move the current node down one step
Make last	Make the current node the last one

If a certain action is not allowed for the current node it will be disabled or hidden in the context menu.

18.4 Navigator View Styles

The *Navigator* view visualizes the ”Execution mode” of job nodes with different view styles.

Table 18.11: View Styles

Execution Mode	View Style

Continued on next page

Table 18.11 – concluded from previous page

Execution Mode	View Style
Execute	Normal
Skip	Icon and text of the node and its complete subtree are grayed out and will be skipped
Neutral	Icon and text of the node are grayed out
View Only	Icon and text of the node are grayed out

18.5 Navigator Tooltips

The *Navigator* view provides detailed information about the icons shown in the state column in form of tooltips.

To see these tooltips just move the mouse on to the state column of the view and then stay on that position for a while.

18.6 Navigator Hotkeys

The *Navigator* view provides several hotkeys. If a *Navigator* view is the current active view, actions can be performed on the current node in the view by simply pressing this key.

Table 18.12: Navigator Hotkeys

Key	Description
F2	Rename
Ctrl + Return	Select / Deselect
Del	Delete

Continued on next page

Table 18.12 – concluded from previous page

Key	Description
Ctrl + X	Cut
Ctrl + C	Copy
Ctrl + V	Paste

The hotkey actions work just as the corresponding context menu actions, so please look at the paragraph above for a detailed description.

18.7 Navigator Drag&Drop Support

The *Navigator* view supports several kind of drag and drop operations:

- Dragging job nodes inside the navigator moves them (if allowed)
- Dragging job nodes inside the navigator with pressed **Ctrl** key copies them (if allowed)
- Dragging pen nodes inside the navigator to the job subtree creates a new **Use pen** node and sets the pen name property of this node to the name of the dragged pen
- Dragging pen nodes inside the navigator to a **Use pen** node the job subtree sets the pen name property of the drop target node to the name of the dragged pen
- Dragging a node from the navigator to some other InScript **User view** sets the content of the drop target view to the dragged node and locks the view. If it is not possible to drag a node to a certain view, the mouse cursor changes when it is dragged over the view
- On Windows you can load job files (files with extension .job and .jobx) by dragging them from the Windows Explorer to a connected controller or the Jobs subnode of a controller
- On Windows you can load pen files (files with extension .pen and .penx) by dragging them from the Windows Explorer to a connected controller or the Pens subnode of a controller
- On Windows you can load an ARGES RawLines vector file (files with extension .arl) to a RawLines node by dragging it from the Windows Ex-

plorer to the RawLines job node

19 Working With The Messages View

The InScript *Messages* view acts as a central collection point for all kind of system messages. This way modal message boxes can be avoided in most cases which makes InScript very well suitable for production environments.

Messages may be nested, i.e. messages can have submessages.

19.1 Message Types

Messages can be of the following types:

- Success messages
- Informational messages
- Warnings and
- Errors.

19.2 Message Fields

Each message in InScript consists of the following parts respectively fields:

Table 19.1: Message parts

Part	Description

Continued on next page

Table 19.1 – concluded from previous page

Part	Description
Date	The timestamp of the message (date and time).
Severity	The kind of the message (see above).
Name, Id	A unique name (or Id) which identifies the message clearly.
Facility	The software module which caused the message.
Caller	Usually the class and method name where the message comes from.
Sender	Identifies the controller in which context the message was triggered. May be empty if the message does not depend in any way from a controller.
Message, Caption	A short description of the message.
Cause	A more detailed description of the cause of the message. May be empty.
Remedy	A hint what can be done to avoid the message. May be empty.

19.3 View Types

The *Messages* view provides 2 different types of view on messages – a list view and a detail view.

In the list view all current messages are shown in a list in short form, i.e. not all available information about the messages are presented there. If there are messages that have submessages the list will be expanded to a tree and the submessages will be shown in subtrees.

The detail view shows all available information of a single message. Thus it is very well suited to get more information about a message, its cause and possible remedies. If a message has submessages these will be shown indented in the detail view.

Additional the list view provides a tooltip for a certain message which will be shown if the mouse cursor is positioned above this message and is not changed for a short time.

19.4 Message Actions

The *Messages* view may be controlled by the following actions.

Table 19.2: Message actions

Action	Description
Show list	Switches to the list view.
Show details	Switches to the detail view.
Delete current	Deletes the currently selected message.
Delete all	Deletes all messages from the list.
Show previous	Shows the previous message in the list (detail view) or activates it (list view).
Show next	Shows the next message in the list (detail view) or activates it (list view).
Focus last	With this option activated the last message in the list will be focused always. This is very useful for incoming message.

All of these actions can be carried out either by the main menu *Messages* or the *Messages* view context menu.

20 Working With The Job Control View

The *Job Control* view serves as the central control for the job execution on the controller. By the help of this view jobs may be selected for execution and the job execution may be started or aborted immediately. Additionally it provides status information about the job execution.

20.1 Job Control Widgets

Table 20.1: Job Control widgets

Widget	Description
InScript logo	Used to signalize laser activity. The InScript logo's beam will flash while the laser is processing.
Start job push button	Starts to process the selected job.
Abort job push button	Aborts the momentarily executed job immediately.
Controller	Shows the name of the controller whose jobs will be controlled by this view.
Selected job	This combobox widget contains the list of available jobs. The selected job will be executed the next time the Start job button will be clicked.

Continued on next page

Table 20.1 – continued from previous page

Widget	Description
Job duration	Shows the current job duration while the job will be executed and the total job duration when the job execution has been finished.
Blue down arrow	Activates detailed status information
Blue down up	Deactivates detailed status information
Devices	Shows the devices summary status
Job	Shows the job summary status
Devices ready	The *Devices ready* state is set whenever all managed devices are in their highest power state (”ready”) and without errors. When at least one of the managed devices is in a lower power state or in error, the *Devices ready* state is cleared.
Devices failure	This state is set whenever at least one of the managed devices is in failure state. It is cleared when the failure vanishes.
Devices setup	Currently not supported in Firmware 3.
Devices awake	The *Devices awake* state is active whenever all managed devices are in the power state ”ready”.
Job ready	An active *Job ready* state means that a job is selected on the controller. It is not affected by other subsystems like a device status. The system can have a *Job ready* state set even if a *Devices failure* is active. A Job start can always be issued when *Job ready* is set. If you set Job start while *Job ready* and *Devices failure* is set, the system tries to recover from the *Devices failure* and bring the devices to a state of *Devices ready*.

Continued on next page

Table 20.1 – concluded from previous page

Widget	Description
Job active	The *Job active* state means that a job is currently running.
Job completed	The *Job completed* is set after a running job has terminated successfully. At the same time, the *Job active* signal is cleared. During job start, the *Job completed* is cleared when the job is about to be started (at the same time when the *Job active* is set).
Last job failed	The *Last job failed* state is set after a running job has terminated with errors or was terminated by the user (Job abort). At the same time, the *Job active* signal is cleared. During job start, the *Last job failed* is cleared when the job is about to be started (at the same time when the *Job active* is set).

For more information about Device and Job states search for ”PLC state” in the firmware documentation.

20.2 Select Job For Execution

InScript can hold several jobs ready in the *Navigator* view. The root node of these jobs has to be a Job node. To determine, which of these jobs shall be executed, it has to be selected for execution first. Only one job may be selected at a time.

There are different, equally good possibilities to select a job for execution:

- In the *Job Control* view, select a job from the list.
- In the *Navigator* view in the Job node's context menu, click **Select**.
- In the *Navigator* view, select a Job node (e.g. by clicking on it) and then press the key combination *Ctrl* + *Enter*.

The job will be loaded into the controller's line buffer. In the *Navigator* view the selected job will be marked with a gray laser beam icon in the state column. A selected job can be deselected in the same way. The job is deleted from the controller's line buffer then.

20.3 Start Job

1. Select the job you want to execute; see section 20.2 on page 104.

   > **⚠ CAUTION**
   >
   > Visible and / or invisible laser radiation
   >
   > Hazard of eye or skin injury
   >
   > - Wear protective glasses that are appropriate for the laser in use.
   > - Avoid eye or skin exposure to direct or scattered radiation.

2. Click the **Start job** button.

20.4 Abort Job

- Click the **Abort job** button.

21 Working With The Vector Editor View

The InScript *Vector Editor* view is a true 3D editing tool, i.e., internally it uses 3D coordinates (x,y,z) throughout. The *Vector Editor* view provides two different view modes, the 2D and 3D view mode, which can be switched.

The 2D view mode shows the plane z=0. It is the preferred mode for the modification and editing of objects, because this is possible in the 2D view mode only at the time. In this mode in the lower left corner of the *Vector Editor* view a two dimensional black axes cross with a schematic x and y axis will be shown.

The 3D view mode allows to view objects from different points of view, thus it is perfectly suited as a viewer for 3D objects as e.g. ARGES RawLines data with 3D contents.

In this mode, the lower left corner of the *Vector Editor* view shows a 3 dimensional axes cross with schematic x, y and z axes. The x axis will be shown red, the y axis green and the z axis blue, so it can easily be kept in mind: RGB stands for xyz. These axes have the same orientation as the center of the view always. Dependent of the current orientation straight, axis parallel lines like grid lines may be perspective distorted more or less. For that reason these lines may not appear parallel to the axes cross if they are located in the near of it. This is caused by the principle and is no failure.

For applications where the third dimension doesn't play any role, the view can be limited to the 2D view mode. All GUI Elements which has to do with the 3D mode will be hidden respectively inactivated then.

For more Information see section 31.6.1 on page 212, subsection `[VectorEditor]`, `Common\LimitTo2D`.

21.1 Modify The View

Because there is neither common nor prevalent control concept for applications which modify vector graphics, we decided just to create a new one which should be as memorable as possible. That is why the InScript *Vector Editor* view follows a very simple control concept:

The *M*iddle mouse button/wheel *M*odifies the view.

Use one of the following operations to change the *Vector Editor* view according to your needs:

Table 21.1: Operations to to change the Vector Editor view

Action	Description
Zoom	Turn the mouse wheel to zoom in or out.
Pan	Press the middle mouse button and drag the mouse to pan the view. or Press the left and the right mouse button together and drag the mouse to pan the view.
Rotate	Press the right mouse button and drag the mouse to rotate the view. This operation is *not* available in the 2D mode.

If your mouse device has no middle mouse button but a wheel try pressing the wheel. The wheel acts as a middle mouse button on many mouse devices.

21.2 Types Of Interaction

The InScript *Vector Editor* view is able to modify objects in several ways. First, single objects or groups of objects can be moved, scaled or rotated in their entirety. This is called the object interaction mode or *object mode* and this is the default interaction mode of the *Vector Editor* view.

Beyond that, the *Vector Editor* view is able to modify special properties of certain objects, e.g., the vertices of a RawLines node. This is called the edit interaction mode, or short the *edit mode*. Objects which can be modified in this way are called editable objects.

Editable objects are:

- RawLines nodes; see section 21.10 on page 121
- PosList nodes; see later on
- Spline nodes; see later on

Use one of the following operations to change the interaction mode:

Table 21.2: Operations to enter the edit mode

Requirement	**Key / method**
There is only one object in the Vector Editor and this object is editable.	Ctrl + E
There is exactly one selected object in the Vector Editor and the object is editable.	Ctrl + E
There is no object selected in the Vector Editor	Ctrl + Alt + click editable object

Table 21.3: Operations to leave the edit mode

Requirement	**Key / method**
Nothing / no vertex is selected	Esc

Continued on next page

Table 21.3 – concluded from previous page

Requirement	Key / method
	Ctrl + Alt + click Vector Editor background

21.3 Change The Selection

Use one of the following mouse operations to modify the selection of objects in the *object mode* of the InScript *Vector Editor* view respectively to modify the selection of vertices or lines in the RawLines *edit mode*.

Table 21.4: Operations to modify the selection of object

Key	Description
Click unselected object	Select object (and deselect all other)
Alt + Click object	Select the next object at the same position
Shift + Click object	Object mode: Extend the selection by the clicked object Edit mode: Extend the selection by the clicked object contiguously along the path
Ctrl + Click object	Toggle the selection state of the object Deselect a selected object
Doubleclick at the VE background	Deselect all objects
Click at the VE background, drag the mouse and release the mouse at another position	Select all objects in the frame between the mouse press and release position

Continued on next page

Table 21.4 – concluded from previous page

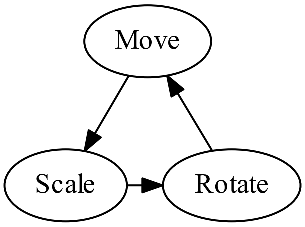
Key	Description
Shift + click at the VE background, drag the mouse and release the mouse at another position	Extend the selection by all objects in the frame between the mouse press and release position
Ctrl + click at the VE background, drag the mouse and release the mouse at another position	Toggle the selection state of all objects in the frame between the mouse press and release position

Alternatively you can use the selection functions in the InScript Edit menu or on the Edit toolbar to modify the selection. Some of these functions do have hot keys you can use e.g. *Ctrl* + *A* to select all objects or *Ctrl* + *D* or *Esc* to deselect all objects.

21.4 Switch The Transformation Mode

The InScript *Vector Editor* view offers three different transformation modes - the Move, Scale and Rotate mode. In the Rotate mode objects can also be sheared/slanted.

Transformation mode switching in the *Vector Editor* view will be performed by clicking the mouse on the selection without moving it. The switch operation always cycles through the available modes:



If no object is selected, the *Vector Editor* view is not in any transformation mode. When the first object will be selected, the move transformation mode will be activated automatically. When the last object will be deselected, the current transformation mode is switched off.

21.5 Move Objects

Perform the following steps to move objects in the *Vector Editor* view:

- Select the objects you want to move
- Set up the Move mode as described above
- Click one of the selected objects and drag the mouse to the desired position

When the *Ctrl* key is pressed while clicking and dragging the selection, the movement can be limited to one axis direction.

21.6 Scale Objects

Perform the following steps to scale objects in the *Vector Editor* view:

- Select the objects you want to scale
- Set up the Scale mode as described above
- Click one of the handles of the selection (the small black squares in 2D mode) and drag the mouse to scale the selection

When the *Ctrl* key is pressed while clicking and dragging the handle, the scale operation will be performed stepwise.

The step width can be set up in the preferences dialog.

When the *Shift* key is pressed while clicking and dragging the handle, the scale operation will be performed centered.

When the key combination *Ctrl* + *Shift* is pressed while clicking and dragging the handle, the scale operation will be performed stepwise and centered.

21.7 Rotate And Slant Objects

Perform the following steps to rotate objects in the *Vector Editor* view:

- Select the objects you want to modify
- Set up the Rotate mode as described above
- Click one of the corner handles of the selection (rounded arrows in 2D mode) and drag the mouse to rotate the selection

When the *Ctrl* key is pressed while clicking and dragging the corner handle, the rotation will be performed stepwise.

The step width can be set up in the preferences dialog.

To modify the rotation center do the following:

- Click the center handle of the selection (target symbol in 2D mode) and drag the mouse to move the rotation center of the selection
- Press the *Ctrl* key while clicking and dragging the rotation center to limit the movement to one axis direction.

To shear/slant objects in the *Vector Editor* view perform the following steps:

- Select the objects you want to modify
- Set up the Rotate mode as described above
- Click one of the middle handles of the selection (straight arrows in 2D mode) and drag the mouse to shear the selection

When the *Ctrl* key is pressed while clicking and dragging the middle handle, the shear operation will be performed stepwise.

The step width can be set up in the preferences dialog.

When the *Shift* key is pressed while clicking and dragging the middle handle, the shear operation will be performed centered.

When the key combination *Ctrl* + *Shift* is pressed while clicking and dragging the middle handle, the shear operation will be performed stepwise and centered.

21.8 Change Visual Appear. Of Objects

Job nodes will be drawn in the *Vector Editor* view in several colors depending on their state:

Table 21.5: Used node colors in the Vector Editor view

ColorName	Description
Standard	This is the default color for all nodes in the *Vector Editor* view, which are not somehow special. This color is mapped from the default pen (the pen with the id = 1) from the *Vector Editor* view pen settings.
PenDefined	Nodes which have a superior pen node will be drawn with the color of this pen or the color of the default pen (black) if the pen could not be found. (continued in next row)

Continued on next page

Table 21.5 – continued from previous page

ColorName	Description
PenDefined	(concluded from previous row) There are some nodes as e.g. the Raw Lines and the Precession Drill node which may contain select pen opcodes in their line data. These nodes act more special. A superior pen node has no influence on the color of these objects. Instead of that the lines of these nodes will be drawn either with the pen colors defined in the *Vector Editor* view preferences; see section 25.3.4 on page 154; or with the colors defined by real pens when the node has a a superior pen set node. Pen set nodes are used to define assignments between pen numbers (used by SelectPen OpCodes) to pens. All vector data after a certain SelectPen OpCode of a RawLines node then will be drawn with the color of the assigned pen. I.e. RawLines nodes with several different SelectPen OpCodes inside may be drawn with different colors. RawLines nodes without any SelectPen OpCodes inside behave like any other node. The color of a pen will be defined by the variable raster.fg_color of the linepar section. If this section and/or variable does not exist in your pen, you have to add the linepar section first and then the pen variable raster.fg_color. See chapter 16 on page 80 for more information about this. The PenDefined color has higher drawing priority than the Standard color.

Continued on next page

Table 21.5 – concluded from previous page

ColorName	Description
Selected	Selected nodes will be drawn with the selection color to distinguish them from unselected ones. This option has higher drawing priority than the PenDefined and Standard color.
Highlighted	Nodes which are selected in the Navigator view tree or beyond a selected node in the Navigator view tree will be drawn with another special color in the *Vector Editor* view. This behavior is called highlighting and can be turned off. This option has the highest drawing priority, i.e. even selected objects will be drawn with the highlighted color.

There is one additional state – ”Inactive”. These objects are not accessible at the moment because they are currently processed (like moving, rotating, etc.) e.g. That’s why they will be drawn with a special color in the *Vector Editor* view too.

The visual appearance of objects and several other aspects regarding the *Vector Editor* view can be controlled in the ”Vector Editor” section of the InScript Preferences dialog.

On the ”view” tab you can e.g. select the colors described above, switch on or off the highlighting of objects and switch on or off if the line start and end points should be visualized especially. If the option ”Show Line Start Points” is checked, a small rectangle will be shown in the appropriate color at the start point of each (multi) line. When the option ”Show Line End Points As Arrows” is checked, a small arrow will be show at the end of each line segment in the appropriate color.

Depending on the graphics capabilities of your system several options of the ”Vector Editor” section in the InScript Preferences dialog may be disabled.

21.9 Working With Layers

21.9.1 Overview

The *Vector Editor* view groups together several graphic information in so called *layers*. All layers can be switched on or off with the help of the *Vector Editor* view context menu to reduce the level of complexity respectively to concentrate on a certain aspect of the graphical representation of the data.

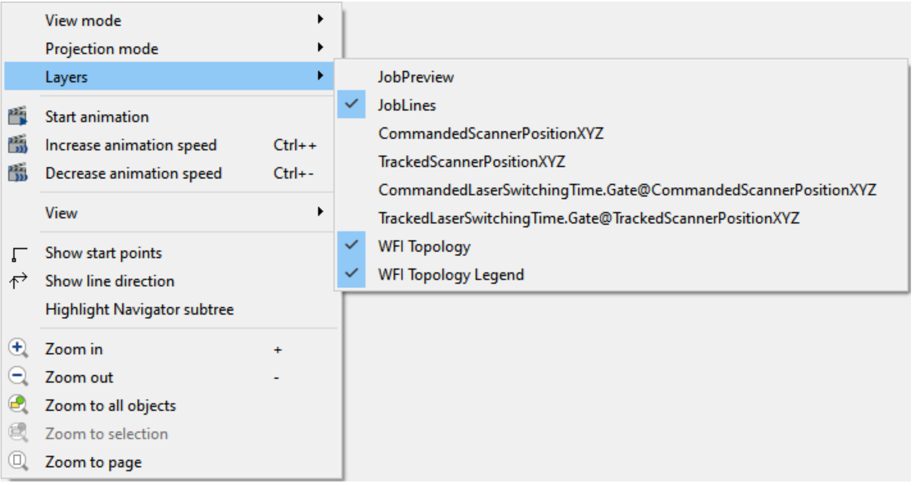
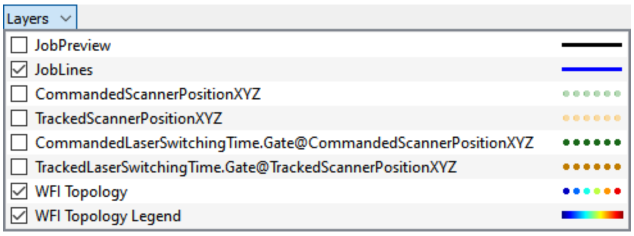

View mode
Projection mode
Layers
Start animation
Increase animation speed Ctrl++
Decrease animation speed Ctrl+-
View
Show start points
Show line direction
Highlight Navigator subtree
Zoom in +
Zoom out -
Zoom to all objects
Zoom to selection
Zoom to page
JobPreview
✓ JobLines
CommandedScannerPositionXYZ
TrackedScannerPositionXYZ
CommandedLaserSwitchingTime.Gate@CommandedScannerPositionXYZ
TrackedLaserSwitchingTime.Gate@TrackedScannerPositionXYZ
✓ WFI Topology
✓ WFI Topology Legend

Figure 21.1: VectorEditor Context Menu

Alternatively they can be switched via the VectorEditor Layers Combobox in the InScript Toolbar.


Layers

☐ JobPreview
☑ JobLines
☐ CommandedScannerPositionXYZ
☐ TrackedScannerPositionXYZ
☐ CommandedLaserSwitchingTime.Gate@CommandedScannerPositionXYZ
☐ TrackedLaserSwitchingTime.Gate@TrackedScannerPositionXYZ
☑ WFI Topology
☑ WFI Topology Legend

Figure 21.2: InScript3 Layers Combobox

The status of the layers (On or Off) is a property of the VectorEditor and will be saved and restored with the layout. This way you can have various Vector-

Editor views with different layers active! Or you can have layout files with different layout presets of the VectorEditor. Additionally some properties can be configured via the InScript ini file for the TSS layers. For details see 27.2 on page 185.

Which layers are available in the *Vector Editor* view will be described in the following.

21.9.2 The Job Lines Layer

This layer contains everything that has to do with the line data of jobs. Interacting with objects in the *object mode* and the *edit mode* takes place on this layer.

The content of the job lines layer follows either the node context or the job context which is the job the node context belongs to (see Getting to know InScript / View / 8.2.4 on page 23 and 8.1 on page 20). Its data is always updated when this context changes. It is also updated when any variable changes that modifies the job lines output.

21.9.3 The Job Preview Layer

The JobPreview layer contains a preview of a job. Preview means all line data is generated the same way as during the job execution. The preview data contains not only the lines which will be visible on the target after the job execution has finished (the main lines) but also all kind of lines where the scan head mirrors moves, but the laser device is switched off. These are the head, jump and tail lines.

Each preview line type will be shown in the *Vector Editor* view with its own color to be able to distinguish them.

These colors are in detail:

Table 21.6: Colors assigned to line types

Line type	Color
Main	Black
Tail	Red
Jump	Yellow
Head	Blue

The preview data has to be updated manually. This can be done with the help of the *Vector Editor* view context menu entry Update Preview. An active preview can be canceled using the context menu entry Cancel Preview. The menu entries Update Preview and Cancel Preview are available only if the preview layer is active.

Another way to update and cancel the preview is the Preview toolbar toggle button on the *Vector Editor* view Mode toolbar.

If the preview is not updating, this button is unlocked. By clicking it the preview will be started. When a preview is currently updating this button will be locked and then shows another icon and tooltip. By clicking the locked button the currently updating preview will be canceled.

At the moment the preview will be generated for the currently selected job always. If no job is selected for execution when an update of the preview is requested an error message will be shown.

21.9.4 The CommandedScannerPosition Layer

The CommandedScannerPosition layer shows the position samples which will be sent (or commanded) by the firmware to the scan head. Thus, it shows the positions where the scan head should move to (target positions). The content of this layer follows the controller context and its data is always updated when the job is executed.

This layer is one of the 27.2.2 on page 186 and will be shown in the VectorEditor only, when the currently used firmware supports it!

Each sample of this layer will be rendered as a small circle, which is oriented to the viewer always (the normal of the circles face is perpendicular to the plane of the screen).

21.9.5 The TrackedScannerPosition Layer

The TrackedScannerPosition layer shows the position samples where scan head has actual moved to. The content of this layer follows the controller context and its data is always updated when the job is executed.

This layer is one of the 27.2.2 on page 186 and will be shown in the VectorEditor only, when the currently used firmware supports it!

Each sample of this layer will be rendered as a small circle, which is oriented to the viewer always (the normal of the circles face is perpendicular to the plane of the screen).

21.9.6 The CommandedLaserSwitchingTime.Gate(at) CommandedScannerPositionXYZ Layer

This layer shows the commanded laser switching events, interpolated to the commanded scanner position. A gate on event will be shown as a small circle at the position, where the event occurred and a gate off event will be shown as a small hatch cross at the position, where the event occurred. The content of this layer follows the controller context and its data is always updated when the job is executed.

This layer is one of the 27.2.3 on page 187 and will be shown in the VectorEditor only, when both TSS, the CommandedScannerPosition and the CommandedLaserSwitchingTime are supported by the firmware which is currently used!

21.9.7 The TrackedLaserSwitchingTime.Gate(at) TrackedScannerPositionXYZ Layer

This layer shows the tracked laser switching events, interpolated to the tracked scanner position. A gate on event will be shown as a small circle at the position, where the event occurred and a gate off event will be shown as a small hatch cross at the position, where the event occurred. The content of this layer follows the controller context and its data is always updated when the job is executed.

This layer is one of the 27.2.3 on page 187 and will be shown in the Vector-Editor only, when both TSS, the TrackedScannerPosition and the Tracked-LaserSwitchingTime are supported by the firmware which is currently used!

21.9.8 The WFI Topology Layer

The ”WFI Topology” layer contains the topology data itself. The content of this layer follows the controller context and its data is always updated when the job is executed.

Each sample of this layer will be rendered as a small circle, which is oriented to the viewer always (the normal of the circles face is perpendicular to the plane of the screen).

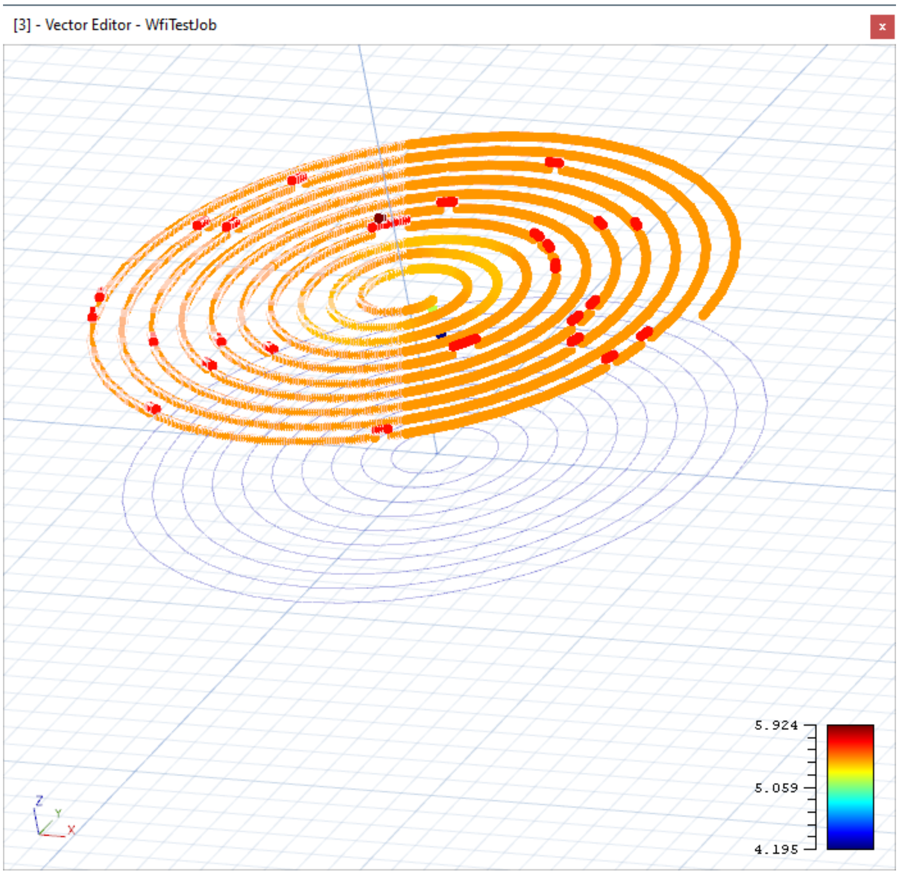


Figure 21.3: 3D View of the VectorEditor WFI Topology Layer

This layer is one of the 27.2.4 on page 188 and will be shown in the VectorEditor only, when the currently used firmware supports it, if the controller with the WFI device is connected in InScript and has an activated WFIBase device.

21.9.9 The WFI Topology Legend Layer

The ”WFI Topology Legend” layer shows the scale (which height value is assigned to a specific color) for the ”WFI Topology” layer.

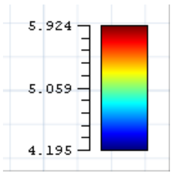


Figure 21.4: The WFI Topology Legend

This layer is one of the 27.2.4 on page 188 and will be shown in the VectorEditor only, when the currently used firmware supports it, if the controller with the WFI device is connected in InScript and has an activated WFIBase device.

21.10 Editing RawLines Nodes

The RawLines Edit mode provides the opportunity to edit single vertices, lines (two vertices which are connected by a line) and complete paths (polylines). All functions of this mode which modify the vertices or their properties, except the selection state, are undo/redo capable.

Especial power gains the RawLines Edit mode in combination with the RawLines Node Properties View. If a single vertex will be selected in the Vector Editor Edit mode, the selected row (cell) of the table in the RawLines Node Properties View will be set to that vertex. And on the other hand - if a certain cell of the table in the RawLines Node Properties View will be selected, the vertex this cell stands for will be selected in the Vector Editor (if it is in the Edit mode and the same RawLines node will be edited there). This makes it very convenient to edit vertices, because you have a graphical feedback now, which vertex you are editing in the Node Properties View.

21.10.1 Select All Vertices

This function refers to all vertices of the node.

21.10.2 Deselect All Vertices

This function refers to all vertices of the node.

21.10.3 Invert Vertex Selection

Invert the current selection of all vertices.

This function refers to all vertices of the node.

21.10.4 Select Next Vertex

Select the next vertex of the current one.

This function refers to the last selected vertex of the node if one ore more vertices are selected. If no vertex is selected yet the first one will be selected.

The shortcut for this function is *Cursor Right*.

21.10.5 Select Previous Vertex

Select the previous vertex of the current one.

This function refers to the first selected vertex of the node if one ore more vertices are selected. If no vertex is selected yet the last one will be selected.

The shortcut for this function is *Cursor Left*.

21.10.6 Select Next Path

Select all vertices of the next path.

This function refers to the last selected vertex of the node if one ore more vertices are selected. If no vertex is selected yet the first path of the node will be selected.

The shortcut for this function is *Ctrl* + *Cursor Right*.

21.10.7 Select Previous Path

Select all vertices of the previous path.

This function refers to the first selected vertex of the node if one ore more vertices are selected. If no vertex is selected yet the last path of the node will be selected.

The shortcut for this function is *Ctrl* + *Cursor Left*.

21.10.8 Select Path

Select all vertices of the path at the current position.

This function refers to the path (any vertex or line of it) at the current position of the mouse pointer.

21.10.9 Deselect Path

Deselect all vertices of the path at the current position. This function refers to the path (any vertex or line of it) at the current position of the mouse pointer.

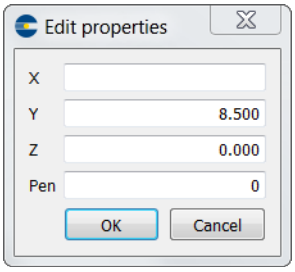
21.10.10 Edit Selection

Edit the position and the pen of all selected vertices.

This function refers to all selected vertices of the node. The shortcut for this function is *Enter*.

For that purpose the ”Edit vertex properties” dialog will be shown.

If one vertex is selected, the edit controls of the dialog show the properties of this vertex. If more vertices are selected and their values of a certain property differ, than the correspondent edit control of the property is shown empty. In other words, a value of a certain property will be shown in the dialog only if it is equal for all selected vertices.



Edit properties

X

Y 8.500

Z 0.000

Pen 0

Figure 21.5: Edit vertex properties Dialog

Entering a value in one of the edit controls of the dialog and closing the dialog with the *OK* button will alter this property for all selected vertices. If you enter a value in one of the coordinate fields (X,Y,Z) with a leading double sign (++ or –) will add the entered value as an offset to this property for all selected vertices.

Setting the pen will work properly for complete paths only at the moment.

21.10.11 Cut

Copies all selected vertices to the clipboard and then deletes them.

The shortcut for this function is *Ctrl + X*.

21.10.12 Copy

Copy all selected vertices to the clipboard.

The shortcut for this function is *Ctrl + C*.

21.10.13 Paste

Paste the content of the clipboard (if it contains RawLines vertices) to the node. The shortcut for this function is *Ctrl + V*.

21.10.14 Delete Selection

Delete all selected vertices.

This function refers to all vertices of the node. The shortcut for this function is *Del*.

21.10.15 Add Dot

Add a dot at the current position. This function refers to the current position of the mouse pointer.

21.10.16 Insert Vertex

Insert a vertex at the current path.

This function refers to the current position of the mouse pointer which must be on a path.

21.10.17 Delete Vertex

Delete the vertex at the current position.

This function refers to the current position of the mouse pointer which must be on a vertex.

21.10.18 New Line

Create a new line at the current position. This function refers to the current position of the mouse pointer.

The new line will be a horizontal line, start at the current mouse position and have a length of 10 mm.

21.10.19 Prepend Line

Prepend a line from the current position to the selected vertex. This function refers to the current position of the mouse pointer.

This function requires a single selected vertex which has to be the start point of a path.

21.10.20 Append Line

Append a line from the selected vertex to the current position. This function refers to the current position of the mouse pointer.

This function requires a single selected vertex which has to be the end point of a path.

21.10.21 Split

Split a single selected vertex. If no vertex is selected but there is a single vertex at the current position of the mouse pointer, this one will be split.

This function requires a single vertex which has to be a mid point of a path (no start point and no end point).

21.10.22 Join

Join two vertices.

This function requires two selected vertices, one of them has to be the end point of one path and the other one has to be the start point of another path.

21.10.23 Open

Remove the line between two vertices.

This function requires a single selected mid line of a path (not the fist line and not the last line of the path).

21.10.24 Connect

Connect two vertices with a line.

This function requires two selected vertices, one of them has to be the end point of one path and the other one has to be the start point of another path.

21.10.25 Make Path First

Move all vertices of the current path to the beginning of the vertex list.

This function refers to the path at the current position of the mouse pointer.

21.10.26 Move Path Up

Move all vertices of the current path in front of the position of the first vertex of the previous path. This function refers to the path at the current position of the mouse pointer.

21.10.27 Move Path Down

Move all vertices of the current path in front of the position of the first vertex of the path after next. This function refers to the path at the current position of the mouse pointer.

21.10.28 Make Path Last

Move all vertices of the current path to the end of the vertex list. This function refers to the path at the current position of the mouse pointer.

21.10.29 Reverse Path

Reverse the direction of the current path. This function refers to the path at the current position of the mouse pointer.

21.10.30 Close Path

Close the current path. This function refers to the path at the current position of the mouse pointer.

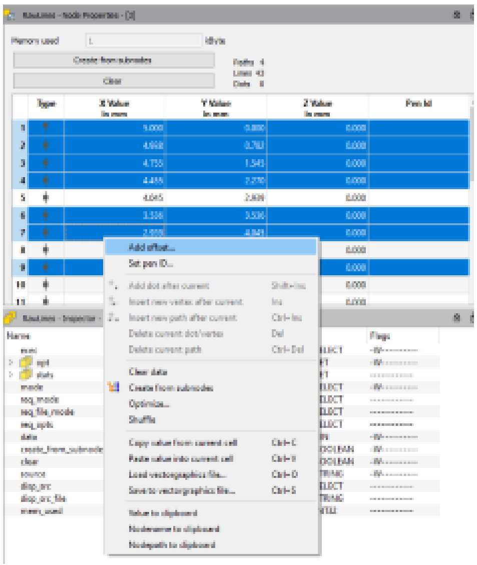
21.10.31 Set As Start Point

Defines the vertex at the current position as new start point for the path. This function refers to the path at the current position of the mouse pointer.

Additional the current position of the mouse pointer must be on a vertex and the path has to be a closed path (the position of the is first and the last point are equal).

21.10.32 Add Offset

Adds a offset to the rows selected in the Rawlines-Editor. X,Y and Z coordinates are set.

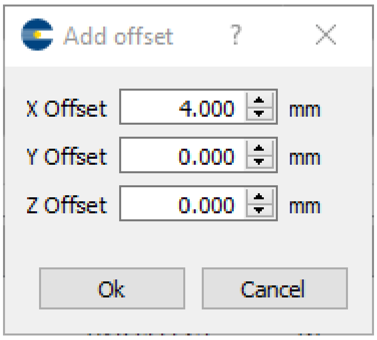

RawLines - Node Properties - [3]
Memory used
kByte
Create from subnodes
Clear
Paths 4
Lines 43
Dots 0
Add offset...
Set pen ID...
Add dot after current Shift+Ins
Insert new vertex after current Ins
Insert new path after current Ctrl+Ins
Delete current dot/vertex Del
Delete current path Ctrl+Del
Clear data
Create from subnodes
Optimize...
Shuffle
Copy value from current cell Ctrl+C
Paste value into current cell Ctrl+V
Load vectorgraphics file... Ctrl+O
Save to vectorgraphics file... Ctrl+S
Value to clipboard
Nodename to clipboard
Nodepath to clipboard
RawLines - Inspector
Name
exec
opt
stats
mode
seq_mode
seq_file_mode
seq_opts
data
create_from_subnodes
clear
source
disp_src
disp_src_file
mem_used
Flags

	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1		5.000	0.000	0.000	
2		4.938	0.782	0.000	
3		4.755	1.545	0.000	
4		4.455	2.270	0.000	
5		4.045	2.939	0.000	
6		3.536	3.536	0.000	
7		2.939	4.045	0.000	
8				0.000	
9				0.000	
10				0.000	
11				0.000	

Figure 21.6: Select multiple rows in the rawlines editor

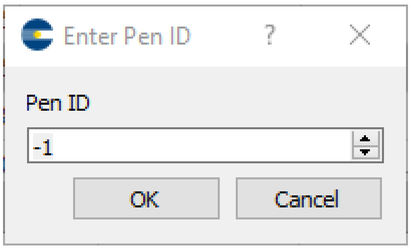
21.10.33 Set Pen ID

Sets a pen ID to the selected rows in the Rawlines-Editor.



X Offset 4.000 mm
Y Offset 0.000 mm
Z Offset 0.000 mm

Figure 21.7: Add offset to selected rawlines



Pen ID
-1

Figure 21.8: Set pen ID to selected rawlines

21.10.34 Create Rawlines From Subnodes

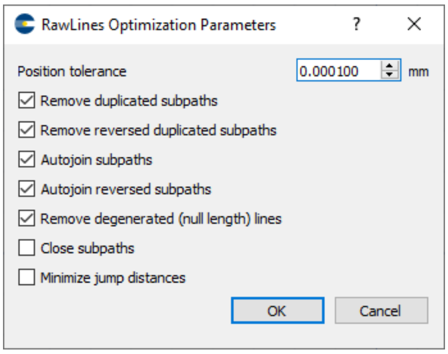
Create geometry data (vertices which may represent lines and or dots) from the subnodes of this node.

21.10.35 Optimize Rawlines Data

Optimizes the data of the node in several ways. Before this function will be exceuted, the RawLines Optimization Parameters dialog will be shown. This way the user can chose which optimization steps shall be performed:

21.10.35.1 Position tolerance

The position tolerance value for auto join and close subpaths (see beyond).



RawLines Optimization Parameters

Position tolerance 0.000100 mm

☑ Remove duplicated subpaths

☑ Remove reversed duplicated subpaths

☑ Autojoin subpaths

☑ Autojoin reversed subpaths

☑ Remove degenerated (null length) lines

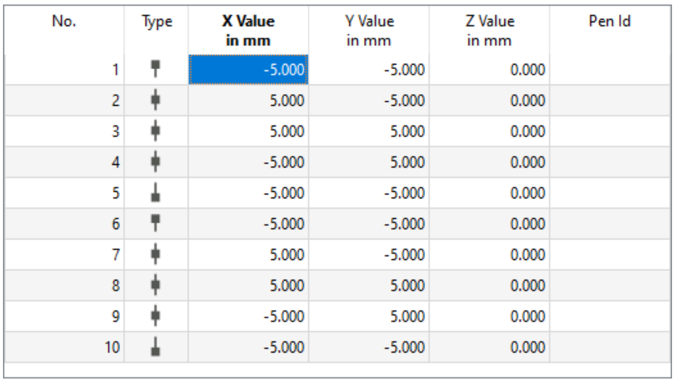
☐ Close subpaths

☐ Minimize jump distances

Figure 21.9: RawLines Optimization Parameters Dialog

21.10.35.2 Remove duplicated subpaths

This function compares all pairs of paths and deletes one of them, if both are equal. In the following example the second path (points 6..10) is a duplicate of the first one (points 1..5). It would be removed by this optimization step.

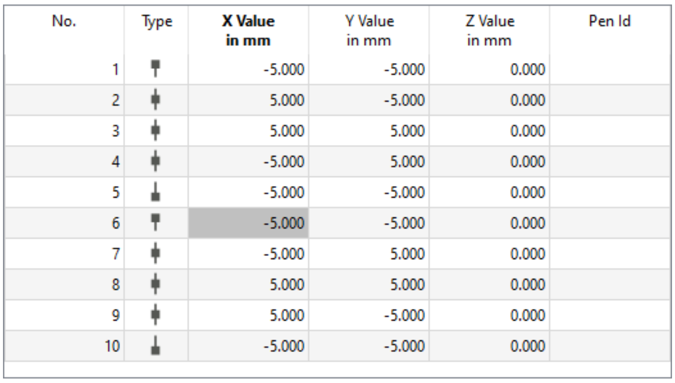


No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	┬	-5.000	-5.000	0.000	
2	┼	5.000	-5.000	0.000	
3	┼	5.000	5.000	0.000	
4	┼	-5.000	5.000	0.000	
5	┴	-5.000	-5.000	0.000	
6	┬	-5.000	-5.000	0.000	
7	┼	5.000	-5.000	0.000	
8	┼	5.000	5.000	0.000	
9	┼	-5.000	5.000	0.000	
10	┴	-5.000	-5.000	0.000	

Figure 21.10: Example for duplicated paths

21.10.35.3 Remove reversed duplicated subpaths

This function compares all pairs of paths and reversed paths and deletes one of them, if both are equal. In the following example the second path (points 6..10) is a reversed duplicate of the first one (points 1..5). It would be removed by this optimization step.



No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	┬	-5.000	-5.000	0.000	
2	┼	5.000	-5.000	0.000	
3	┼	5.000	5.000	0.000	
4	┼	-5.000	5.000	0.000	
5	┴	-5.000	-5.000	0.000	
6	┬	-5.000	-5.000	0.000	
7	┼	-5.000	5.000	0.000	
8	┼	5.000	5.000	0.000	
9	┼	5.000	-5.000	0.000	
10	┴	-5.000	-5.000	0.000	

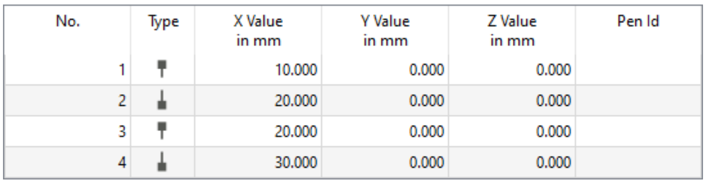
Figure 21.11: Example for duplicated reversed paths

21.10.35.4 Autojoin subpaths

Compares the start and end points of all pairs of paths and joins them, if one of the points are the same (or within the position tolerance).

Example 1:

The end point of the first (2) and the start point of the second path (3) match. They will be joined together. This way the second path will be appended to the first one.



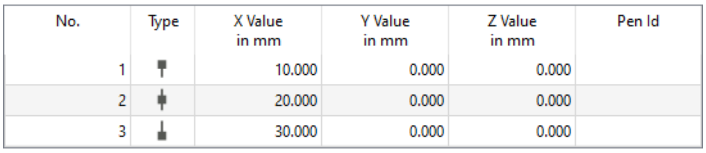
No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	┬	10.000	0.000	0.000	
2	┴	20.000	0.000	0.000	
3	┬	20.000	0.000	0.000	
4	┴	30.000	0.000	0.000	

Figure 21.12: Autojoin subpaths example 1

The point list will look like this after the optimization:

Example 2:

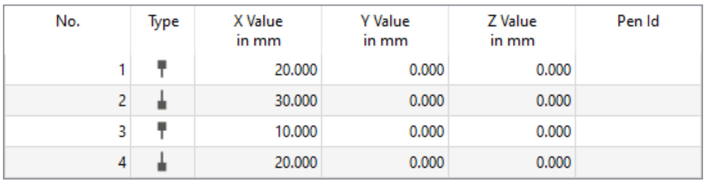
The start point of the first (1) and the end point of the second path (4) match. The second path will be moved up in order and then the matching points will be joined together. This way the second path will be prepended to the first



No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	╤	10.000	0.000	0.000	
2	┼	20.000	0.000	0.000	
3	┴	30.000	0.000	0.000	

Figure 21.13: Autojoin result example 1 and following

one.



No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	╤	20.000	0.000	0.000	
2	┴	30.000	0.000	0.000	
3	╤	10.000	0.000	0.000	
4	┴	20.000	0.000	0.000	

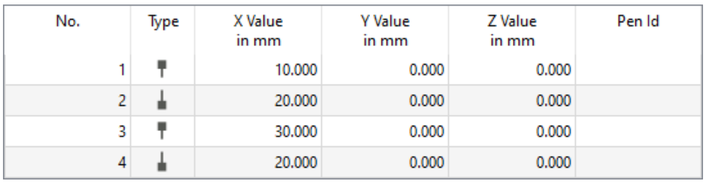
Figure 21.14: Autojoin subpaths example 2

21.10.35.5 Autojoin reversed subpaths

Compares the start and end points of all pairs of paths and joins them, if one of the points are the same (or within the position tolerance).

Example 1:

The end points of the first (2) and the second path (4) match. The second path will be reversed and then the matching points will be joined together. This way the reversed second path will be appended to the first one.

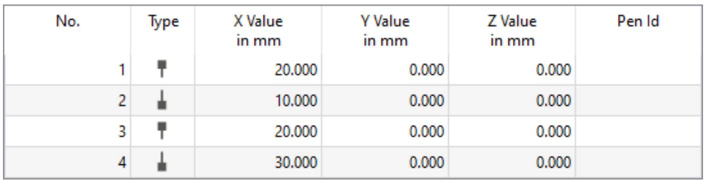


No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	╤	10.000	0.000	0.000	
2	┴	20.000	0.000	0.000	
3	╤	30.000	0.000	0.000	
4	┴	20.000	0.000	0.000	

Figure 21.15: Autojoin reversed subpaths example 1

Example 2:

The start points of the first (1) and the second path (3) match. The first path will be reversed and then the matching points will be joined together. This way the second path will be appended to the reversed first one.



No.	Type	X Value in mm	Y Value in mm	Z Value in mm	Pen Id
1	⊤	20.000	0.000	0.000	
2	⊥	10.000	0.000	0.000	
3	⊤	20.000	0.000	0.000	
4	⊥	30.000	0.000	0.000	

Figure 21.16: Autojoin reversed subpaths example 2

21.10.35.6 Remove degenerated (null length) lines

Removes all lines which are degenerated, i.e. their length is null

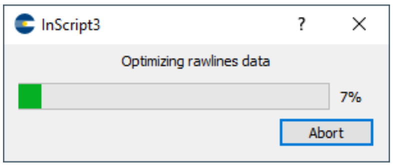
21.10.35.7 Close subpaths

Sets the end point of all paths to the start point if start and end points are within the position tolerance.

21.10.35.8 Minimize jump distances

Minimizes the jump distance between all paths and dots.

During the optimization another dialog will be shown which shows the currently executed task and the progress of the optimization. Additionally the currently running optimization can be aborted by the help of the Abort button in this dialog.



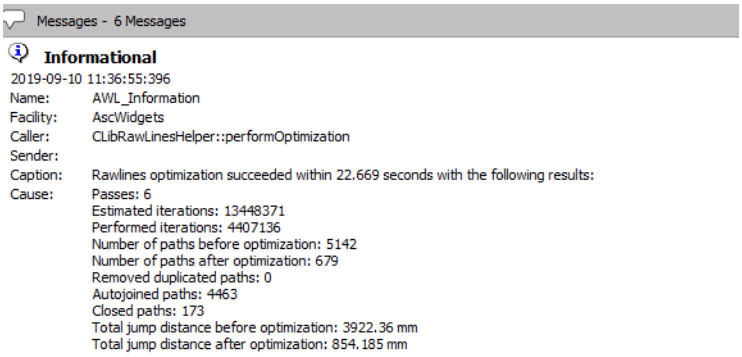
InScript3

Optimizing rawlines data

7%

Figure 21.17: Rawlines optimization progress

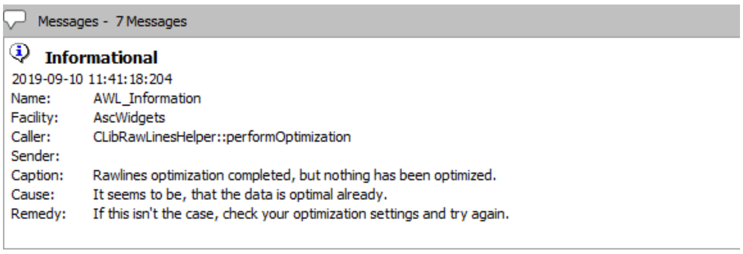
If the optimization has finished, an informational message with the optimization results will be shown in the MessageView.



```
Messages - 6 Messages
Informational
2019-09-10 11:36:55:396
Name:       AWL_Information
Facility:   AscWidgets
Caller:     CLibRawLinesHelper::performOptimization
Sender:
Caption:    Rawlines optimization succeeded within 22.669 seconds with the following results:
Cause:      Passes: 6
            Estimated iterations: 13448371
            Performed iterations: 4407136
            Number of paths before optimization: 5142
            Number of paths after optimization: 679
            Removed duplicated paths: 0
            Autojoined paths: 4463
            Closed paths: 173
            Total jump distance before optimization: 3922.36 mm
            Total jump distance after optimization: 854.185 mm
```

Figure 21.18: Informational message when rawlines optimization succeeded

If nothing has been optimized for certain reason, an informational message will be shown too.



```
Messages - 7 Messages
Informational
2019-09-10 11:41:18:204
Name:       AWL_Information
Facility:   AscWidgets
Caller:     CLibRawLinesHelper::performOptimization
Sender:
Caption:    Rawlines optimization completed, but nothing has been optimized.
Cause:      It seems to be, that the data is optimal already.
Remedy:     If this isn't the case, check your optimization settings and try again.
```

Figure 21.19: Informational message when rawlines optimization finished without changing anything

21.10.36 Shuffle Rawlines Data

Arrange all paths and dots of the node in random order

21.11 Line Order Animation

The InScript *Vector Editor* view is able to visualize the order in which the lines of the objects in the view are executed respectively marked by the controller and the used laser. This feature is called *line order animation*.

All line order animation function can be executed by the help of the *Vector Editor* view context menu. Additionally the line order animation can be started and stopped by a toolbar button. As a third opportunity the *Vector Editor* view provides hotkeys for all line order animation function.

The following functions are available to control the line order animation:

Table 21.7: Line Order Animation Functions

Function	Description
Start animation	Starts the line order animation
Stop animation	Terminates the line order animation
Increase animation speed	Increases the drawing speed of the line order animation
Decrease animation speed	Decreases the drawing speed of the line order animation

21.12 Vector Editor View Hotkeys

The *Vector Editor* view provides several hotkeys. If an *Vector Editor* view is the current active view, actions can be performed on the current node in the view by simply pressing this hotkey.

Table 21.8: Vector Editor View Object Mode Hotkeys

Key	Description

Continued on next page

Table 21.8 – concluded from previous page

Key	Description
Cursor Left	Select Previous Object
Cursor Right	Select Next Object

Table 21.9: Vector Editor View RawLines Edit Mode Hotkeys

Key	Description
Cursor Left	Select Previous Vertex
Cursor Right	Select Next Vertex
Ctrl + Cursor Left	Select Previous Path
Ctrl + Cursor Right	Select Next Path
Shift + Cursor left, or Shift + Cursor up	Extend selection backward (along the path)
Shift + Cursor right, or Shift + Cursor down	Extend selection forward (along the path)
F2	Edit Selection
Ctrl + X	Cut To Clipboard
Ctrl + C	Copy To Clipboard
Ctrl + V	Paste From Clipboard
Del	Delete Selection

Table 21.10: Vector Editor View Common Hotkeys

Key	Description
Ctrl + A	Select All
Ctrl + D	Deselect All
Esc	Deselect All
Plus	Zoom In

Continued on next page

Table 21.10 – concluded from previous page

Key	Description
Minus	Zoom Out
F5	Redraw
F6	Refresh Job Lines
F7	Zoom To All Objects
F8	Zoom To Page
Space	Start / stop line order animation
Ctrl + +	Increase animation speed
Ctrl + -	Decrease animation speed

The hotkey actions work just as the corresponding context menu actions. Please look at the paragraph above for a detailed description.

22 Working With The Node Properties View

22.1 General Information

The *Node Properties* view provides a convenient way to edit the most important variable values (properties) of a node. For that a predefined user interface (UI) for each node will be requested from the controller and shown. If no UI is available for a certain node this view remains empty. In that case or if a certain variable is needed which is not shown in the predefined UI the Inspector view can be used to modify these variable values.

The InScript *Node Properties* view is the replacement for the InScript 2 node editors. Because the UI is provided by the controller (and it's firmware) now, InScript never runs into situations where the firmware variables and the associated GUI mismatch.

The UI's presented in the *Node Properties* view are optimized for a design with two columns in almost every case. Furthermore these two columns are designed to have an equal minimum size and to stretch in horizontal direction for a certain amount up to a predefined maximum size.

22.2 Node Properties View Navigation

The navigation in the *Node Properties* view is the same as the navigation in any other InScript dialog, as e.g. the preferences dialog. You can go to the next UI element by pressing the *TAB* key or go to the previous UI element by pressing the key combination *Shift* +/nbsp;*TAB*, alter values of comboboxes, spin boxes and float spin boxes by pressing the *Up* or *Down* key if the widget has the focus and so on.

23 Working With The Inspector View

The InScript *Inspector* view can be used to inspect, rename or edit all subnodes of a node, including its variables. For that purpose it shows a variable tree of a certain node on the controller, this node might be a job node, a pen or a device e.g. The tree provides five columns – ”Name”, ”Value”, ”Unit”, ”Type” and ”Flags” whereat the ”Name” column not only shows the name but carries the branch decoration too.

23.1 Inspector View Navigation

Use the following keys to navigate inside the *Inspector* view, respectively to change the current selected node inside the tree:

Table 23.1: Inspector Keys

Key	Description
Down arrow key	Navigate to the node below the current one. (Make the node below the current one the new current node.)
Up arrow key	Navigate to the node above the current one. (Make the node above the current one the new current node.)

Continued on next page

Table 23.1 – concluded from previous page

Key	Description
Right arrow key	Expand the current node.
Left arrow key	Collapse the current node.

23.2 The Inspector View Inplace Editing

The *Inspector* view provides the ability to change a node name or value directly in the tree without opening a dedicated editor window. This is called ”inplace editing”. Inplace editing basically will be started by clicking on an already selected item in InScript.

To start the inplace editing of a node name or value in the *Inspector* view, select the node you want to change by simply clicking on it or using the arrow keys (see paragraph above). Doing so causes the node to become the current node of the tree. Then click the node again to start inplace editing. If you want to change the name of the node click on the ”Name” column and if you want to change a variable value click on the ”Value” column.

Changing the value of a variable is very easy:

- if the node is already selected:
  - left-click on the value
  - start typing
  - press the **ENTER** key
  - press the key combination **CTRL** + **UP** or **DOWN** (to increase or decrease the value by 1)
- if the node is not already selected:
  - double-left-click on the value

23.3 The Inspector View Context Menu

The Inspector provides a context menu to execute several actions on a node.

Table 23.2: Inspector Keys

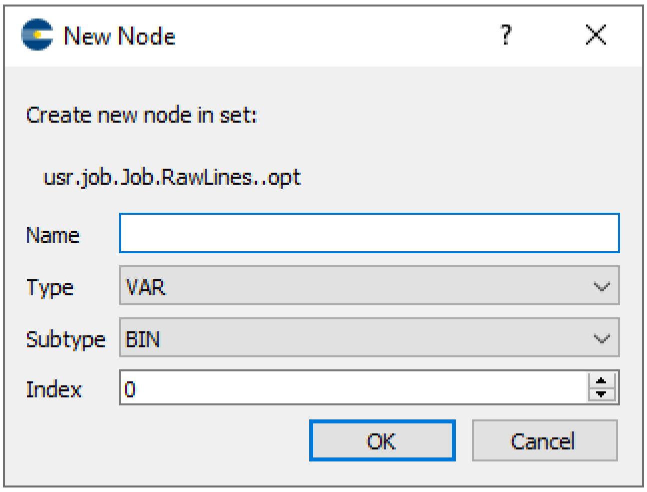
Action	Description
Create node...	Creates a new node. For that purpose the New Node Dialog will be shown. For more details see section 23.4 on page 141.
Rename	Rename a node Starts inplace editing of the name column.
Edit Variable Value	Edit the value of a variable Starts inplace editing of the value column.
Select / Deselect	Select / Deselect a job node Toggles the selection state of a job node.
Delete	Deletes a node
Value to clipboard	Copies the value of the current node to the clipboard.
Nodename to clipboard	Copies the name of the current node to the clipboard.
Nodepath to clipboard	Copies the path of the current node to the clipboard.
Add to pen	See section 16.3.3 on page 83.

If a certain action is not allowed for the current node it will be disabled or hidden in the context menu.

23.4 The New Node Dialog

The New Node Dialog will be shown if the context menu function ”Create Node...” is selected.

To create a new node first you have to enter the desired name of the new node in the ”Name” input box. Then select the ”Type” and the ”Subtype” of the new node from the combo boxes of the same name. Finally you may decide at which position in the parent node the new node will be created by choosing the ”Index”.



New Node

Create new node in set:

usr.job.Job.RawLines..opt

Name:

Type: VAR

Subtype: BIN

Index: 0

Figure 23.1: New Node Dialog

23.5 Inspector Hotkeys

The *Inspector* view provides several hotkeys. If an *Inspector* view is the current active view, actions can be performed on the current node in the view by simply pressing this key.

Table 23.3: Inspector Hotkeys

Key	Description
F2	Rename node
Return	Edit variable value
Ctrl + Return	Select / Deselect node
Del	Delete node
Ctrl + Cursor up	Increase the variable value (Works for INT32/64 and REAL32/64)

Continued on next page

Table 23.3 – concluded from previous page

Key	Description
Ctrl + Cursor down	Decrease the variable value (Works for INT32/64 and REAL32/64)

The hotkey actions work just as the corresponding context menu actions, so please look at the paragraph above for a detailed description.

24 Working With The UI File Container View

24.1 General Information

In some situations it may be necessary to extend InScript to achieve special aims or to make InScript even more comfortable for special requirements. One means of choice for that may be a custom user interface.

InScript is build with the Qt framework which comes with a tool for designing and building graphical user interfaces (GUIs) – the Qt Designer. This tool makes it possible to compose and test dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner. Usually almost every kind of widget can be used in these dialogs. For the InScript development a special plugin for the Qt Designer was created which contains the so called ARGES widgets. These widgets are able to connect with a certain variable of a controller. They integrate seamless into the Qt Designer, so it becomes very easy to design custom specific user interfaces (UI) not only for InScript but for any conceivable purposes.

The *UI File Container* view provides the possibility to load and show user interface files – or forms – designed with the Qt Designer using ARGES widgets. This way it is possible to show and / or to edit a bunch of variables of the InScript context node or of one or more certain nodes of a controller. Thus these UI files can act as node or device editors as well as an user interface for all variables of a job which shall be changeable by the operator.

24.2 Creating Custom UI Files

If you need customized UIs, there are 2 options to get them.

First, you may contact the ARGES marketing and request an offer. This is the easiest way for you, because you do not need to know anything about the Qt Designer and the ARGES widgets.

The second way will be available with InScript version 3.1. For this method you will need a special software package which contains the Qt Designer and the ARGES widgets as well as the documentation of the latter. Additional you need the Qt Designer manual which can be found in the internet.

We recommend the following readings:

- Qt Designer Manual[1]
- A Quick Start to Qt Designer[2]
- Getting to Know Qt Designer[3]

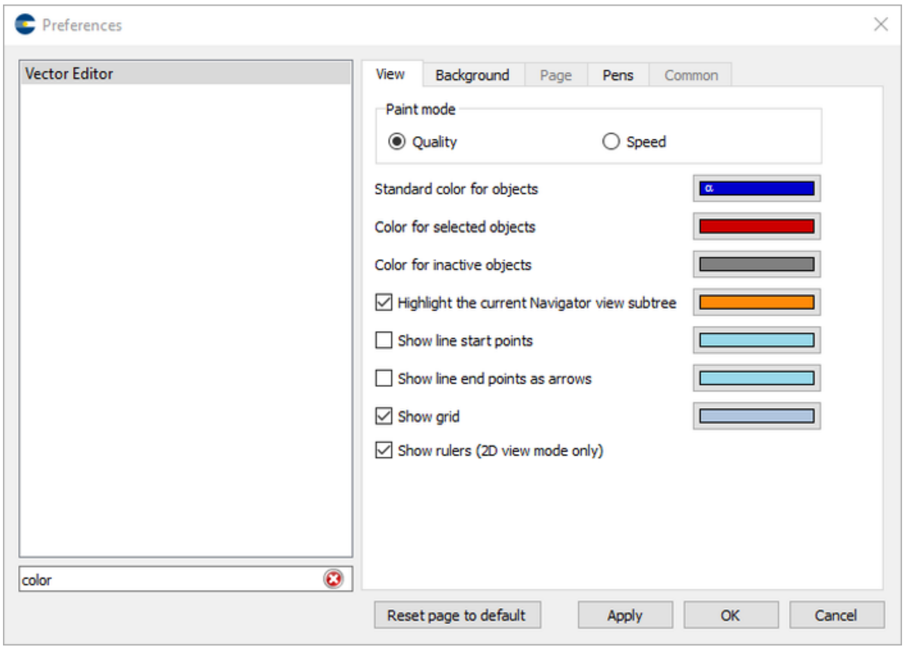
[1] http://doc.qt.io/qt-5/qtdesigner-manual.html
[2] http://doc.qt.io/qt-5/designer-quick-start.html
[3] http://doc.qt.io/qt-5/designer-to-know.html

25 Managing Preferences

25.1 Preferences Window

Open the *Preferences* window by clicking **File** → **Preferences** in the main menu.

This section describes the GUI (Graphical User Interface) elements common to all following preferences dialogs.



Paint mode

● Quality ○ Speed

Standard color for objects

Color for selected objects

Color for inactive objects

☑ Highlight the current Navigator view subtree

□ Show line start points

□ Show line end points as arrows

☑ Show grid

☑ Show rulers (2D view mode only)

color

Figure 25.1: Preferences window - example filtered for the term *color (bottom left)*

Description of figure 25.1 on page 146:

Category list on the left side groups preferences in categories.

Tabs on the right side group settings for the respective category.

Filter text box below the categories list filters the categories and respective tabs for a term given here.

Categories not matching the term will be hidden in the category list and tabs not matching the term will be disabled. Find an example in the figure above.

To remove the filter again click the red ⊗ symbol in the text box.

Reset page to default resets the changes made to default in the active tab.

Apply accepts changes made without closing the *Preferences* window.

OK accepts changes made and closes the *Preferences* window.

Cancel dismisses changes made and closes the *Preferences* window.

Please find the remaining GUI elements in the respective section.

25.2 InScript Category

In this category customize InScript's behavior on startup and shutdown and basic properties of the GUI (Graphical User Interface).

25.2.1 Common Tab

On this tab customize InScript's behavior on startup and exit.

Description of figure 25.2 on page 148:

Controller connections defines the connections of controllers in InScript.

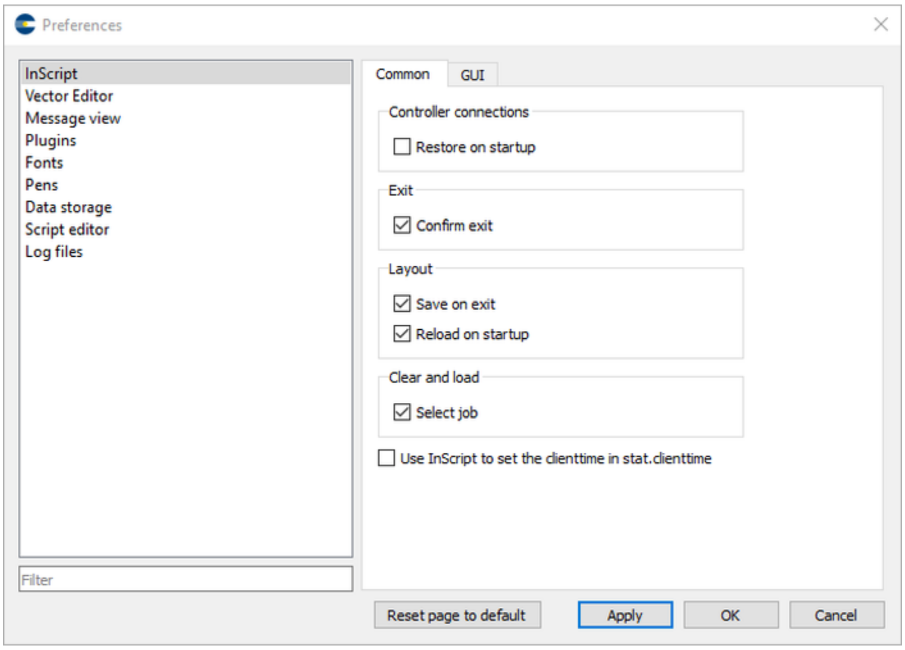
Restore on startup If checked controllers which were connected on the last exit of InScript are connected again.

Exit defines the behavior of InScript on exit.

Confirm exit requests, if checked, confirmation on exit.

Layout defines the handling of the layout.

Save on exit saves, if checked, the arrangement of all InScript views



Controller connections

☐ Restore on startup

Exit

☑ Confirm exit

Layout

☑ Save on exit

☑ Reload on startup

Clear and load

☑ Select job

☐ Use InScript to set the clienttime in stat.clienttime

Figure 25.2: Preferences window - InScript category - Common tab

to file `standard.lay3` in the `Config` subdirectory of the InScript `Documents` directory.

Reload on startup loads, if checked, the layout form file `standard.lay3` on startup.

Clear and load defines the behavior after a clear job an load job action that can be initialized e.g. by clicking *File* > *Clear and load* in the main menu.

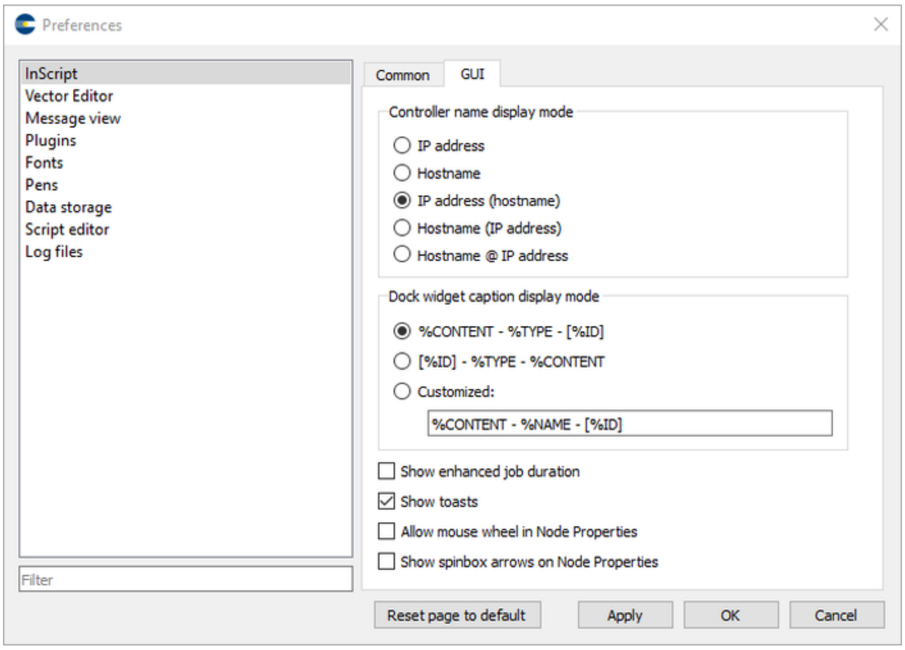
Select job selects automatically the job for execution, that has been loaded by a clear and load job action, if activated.

Use InScript to set the clienttime in stat.clienttime [tbd]

Please find the remaining GUI elements in section 25.1 on page 146.

25.2.2 GUI Tab

On this tab customize InScript's basic properties of the GUI (Graphical User Interface).



Controller name display mode

○ IP address
○ Hostname
● IP address (hostname)
○ Hostname (IP address)
○ Hostname @ IP address

Dock widget caption display mode

● %CONTENT - %TYPE - [%ID]
○ [%ID] - %TYPE - %CONTENT
○ Customized:
%CONTENT - %NAME - [%ID]

□ Show enhanced job duration
☑ Show toasts
□ Allow mouse wheel in Node Properties
□ Show spinbox arrows on Node Properties

Figure 25.3: Preferences window - InScript category - GUI tab

Description of figure 25.3 on page 149:

Controller name display mode selects how the name and IP-address of a controller is displayed. For a preview see the tool tip of each entry on screen:

IP address

Hostname

IP address (hostname)

Hostname (IP address)

Hostname @ IP address

Dock widget caption display mode selects how captions in the dock widgets are formatted. `%CONTENT`, `%TYPE`, `%NAME` and `%ID` are placeholders and replaced as follows: `%CONTENT` the name of the node or controller context show in the widget, `%TYPE` the type of the dock widget, `%NAME` the name of the controller and `%ID` the id of the dock widget.

The group contains 2 predefined settings and 1 customized setting:

%CONTENT - %TYPE - [%ID] e.g. shown as `Transform - Inspector - [2]`

[%ID] - %TYPE - %CONTENT e.g. shown as `[2] - Inspector - Transform`

Customized allows, if selected, to enter a customized caption definition with at least one of the placeholders mentioned above.

Show enhanced job duration shows, if checked, the job duration in an enhanced form.

Show toasts shows toasts, if checked. A toast is a non modal, unobtrusive window element used to display brief, auto-expiring windows of information to a user.

Allow mouse wheel in Node Properties allows, if checked, using the mouse wheel in the Node Properties window.

Show spinbox arrows on Node Properties shows, if checked, spinbox arrows on numerical fields in the Node Properties window.

Please find the remaining GUI elements in section 25.1 on page 146.

25.3 Vector Editor Category

In this category customize various settings for the *Vector Editor*.

> **TIP**
>
> Depending on the graphical capabilities of the computer system, InScript is running on, some tabs or settings may be disabled in this category.

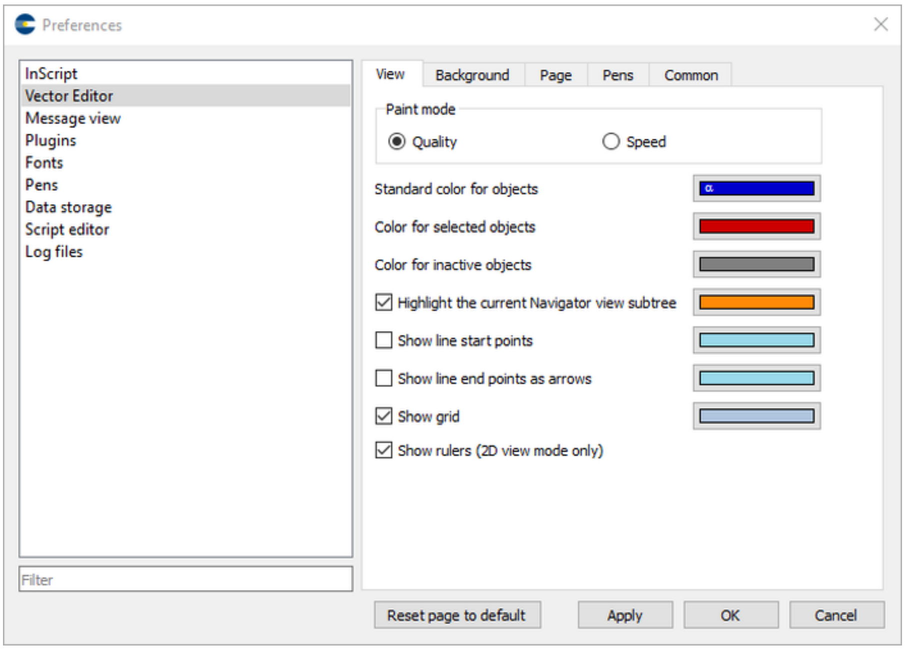
25.3.1 View Tab

On this tab customize settings that effect the visual appearance of objects in the *Vector Editor*.

On a related topic please see also the section 21.8 on page 113.

Description of figure 25.4 on page 151:

Paint mode selects the drawing routines for the *Vector Editor*.



Paint mode

● Quality ○ Speed

Standard color for objects 🟦

Color for selected objects 🟥

Color for inactive objects ⬛

☑ Highlight the current Navigator view subtree 🟧

☐ Show line start points 🟦

☐ Show line end points as arrows 🟦

☑ Show grid 🟦

☑ Show rulers (2D view mode only)

Figure 25.4: Preferences window - Vector Editor category - View tab

Quality enhances the quality of drawing in the *Vector Editor* and is the preferred option.

Speed enhances the speed of drawing in the *Vector Editor*. If you encounter performance problems with the *Vector Editor* then select this option.

Standard color for objects selects the standard color for nodes in the *Vector Editor*. This color value maps the color of pen No. 1 from the *Vector Editor* – *Pens* tab.

Color for selected objects selects the color for selected objects.

Color for inactive objects selects the color for all objects that are not accessible at the moment because they are currently being processed, i.e. moved, rotated, etc.

Highlight the current Navigator view subtree highlights, if checked, objects from the current *Navigator* subtree in the *Vector Editor* and selects the color for highlighting. The *Vector Editor* is the graphical representation of a job in the *Navigator*. Highlighting describes the ability of the

Vector Editor to draw nodes, which are selected in the *Navigator* tree or beyond a selected node in the *Navigator* tree, with a special color. If a job contains many objects highlighting may help to identify single objects or certain subtrees in the *Vector Editor*.

Show line start points shows, if checked, a small rectangle at the start point of each (multi-)line and selects a color for this rectangle.

Show line end points as arrows shows, if checked, a small arrow head at the end of each line segment and selects a color for this arrow head.

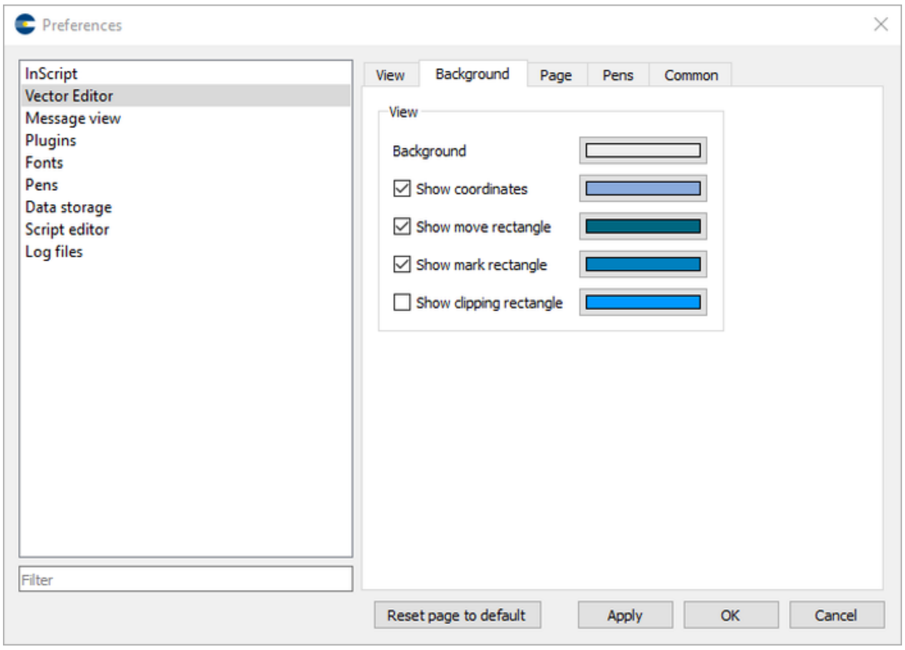
Show grid shows, if checked, a grid in the *Vector Editor*.

Show rulers (2D view mode only) shows, if checked, rulers at the left and upper edge of the *Vector Editor*when it is 2D view mode.

Please find the remaining GUI elements in section 25.1 on page 146.

25.3.2 Background Tab

On this tab customize the background of the *Vector Editor*.



View

Background

☑ Show coordinates

☑ Show move rectangle

☑ Show mark rectangle

□ Show clipping rectangle

Figure 25.5: Preferences window - Vector Editor category - Background tab

Description of figure 25.5 on page 152:

View defines elements that shall be shown in the *Vector Editor* background.

Background selects the background color.

Show page coordinates shows, if checked, the coordinate plane and the axes legend and selects the color for these.

Show move rectangle shows, if checked, the rectangle, which encloses the area where the scanner mirrors can move to and selects the color for this rectangle.

Show mark rectangle shows, if checked, the rectangle, which encloses the area where the laser beam is allowed to be switched on and selects the color for this rectangle.

Show clipping rectangle shows, if checked, the rectangle, which encloses the laser output area and selects the color for this rectangle.

Please find the remaining GUI elements in section 25.1 on page 146.

25.3.3 Page Tab

On this tab customize the page of the *Vector Editor*.

Description of figure 25.6 on page 154:

Page Size determines how the page is shown in the *Vector Editor*.

User defined sets, if unchecked, the page size to the settings in the *dist_xy* device of the controller the root node of the current view belongs to. In that case the page size will be equivalent to the whole scan field. *User defined* sets, if checked, the page size by the following settings in distances respective to the scan field origin:

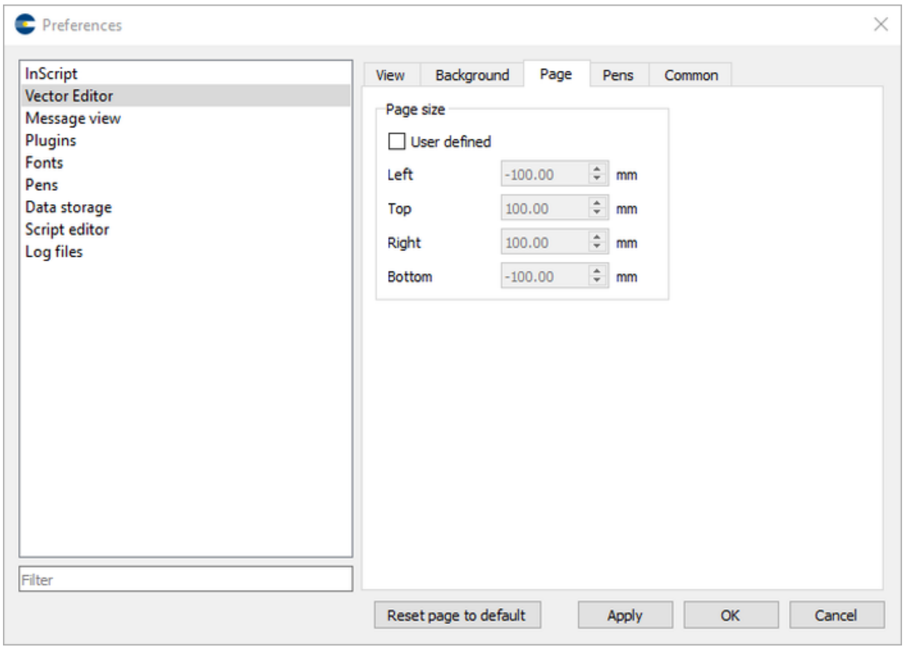
Left

Top

Right

Bottom

Please find the remaining GUI elements in section 25.1 on page 146.



Page size

□ User defined

Left -100.00 mm

Top 100.00 mm

Right 100.00 mm

Bottom -100.00 mm

Figure 25.6: Preferences window - VectorEditor Page tab

25.3.4 Pens Tab

There are some nodes, as e.g. the *Precession Drill* and the *Raw Lines* node, which may contain *select pen* opcodes in their line data.

For the *Precession Drill* node these *pen select* opcodes have no other function than to visualize the different spiro states by different colors in the *Vector Editor*.

In case of the *Raw Lines* node a *pen select* opcode defines the pen, a certain line segment shall be laser output with. A superordinate *Pen Map* node is required to map the *pen select* opcodes to pens.

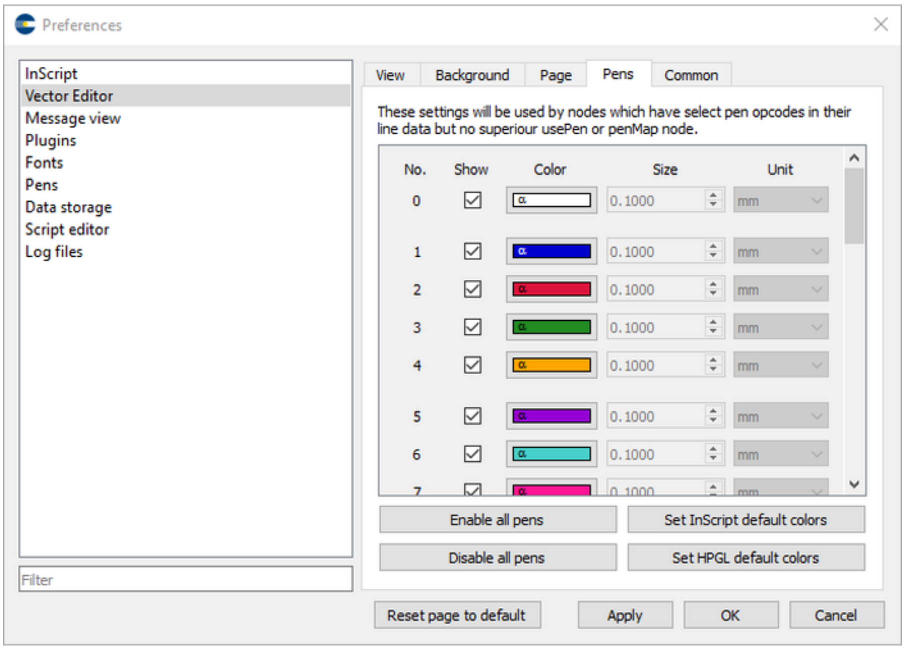
In order to make different pens distinguishable even without a superordinate *Pen Map* node, there is a set of standard settings for all pens.

On this tab customize these pen settings.

InScript supports up to 32 different pen numbers and therefore pens. For compatibility with HPGL/PLT data there is one additional pen for the background. This background pen has the number 0 and is virtually never used.

All other pens have numbers from 1 to 32. Pen number 1 is the default pen, i.e. the settings of this pen are used for laser output whenever no other pen is specified.

On a related topic please see also the section 21.8 on page 113.



These settings will be used by nodes which have select pen opcodes in their line data but no superiour usePen or penMap node.

No.	Show	Color	Size	Unit
0	☑	⬜ α	0.1000	mm
1	☑	🟦 α	0.1000	mm
2	☑	🟥 α	0.1000	mm
3	☑	🟩 α	0.1000	mm
4	☑	🟧 α	0.1000	mm
5	☑	🟪 α	0.1000	mm
6	☑	🟦 α	0.1000	mm
7	☑	🟥 α	0.1000	mm

Figure 25.7: Preferences window - Vector Editor category - Pens tab

Description of figure 25.7 on page 155:

Table The properties for all pens are laid out as a table with the following columns:

No. shows pen number.

Show show, if checked, the lines drawn with this pen in the Vector Editor.

If cleared then all lines drawn with this pen are hidden in the Vector Editor.

Color selects the color for the pen ID is shown in.

Size sets the width of a line (prepared for later use).

Unit sets the unit of the size (prepared for later use).

Enable all pens checks all checkboxes at once in column *Show*.

Disable all pens clears all checkboxes at once in column *Show*.

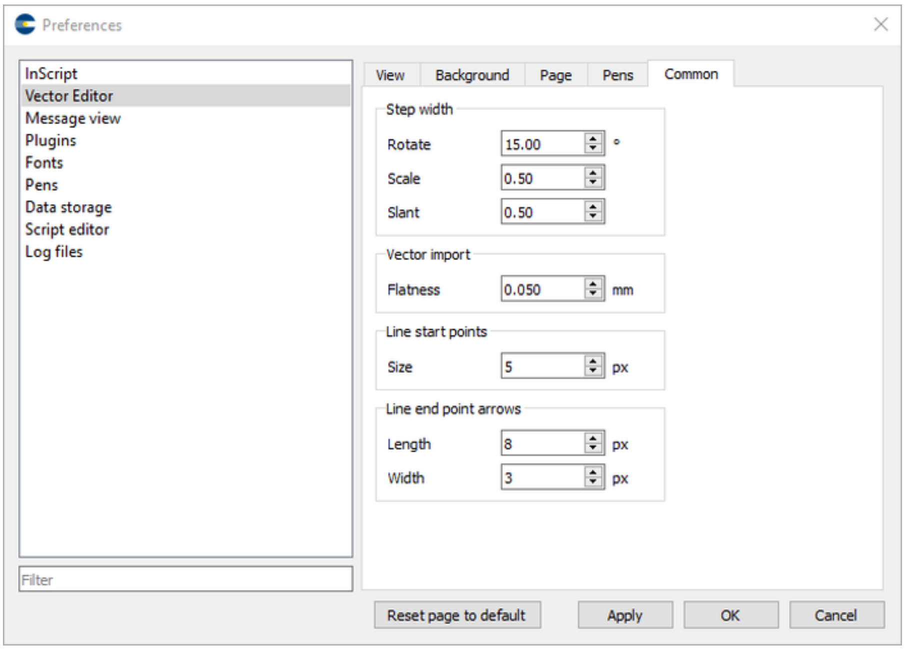
Set InScript default colors sets colors to InScript color scheme.

Set HPGL default colors sets colors to HPGL color scheme.

Please find the remaining GUI elements in section 25.1 on page 146.

25.3.5 Common Tab

On this tab customize basic properties of the *Vector Editor*.



Step width

Rotate 15.00 °

Scale 0.50

Slant 0.50

Vector import

Flatness 0.050 mm

Line start points

Size 5 px

Line end point arrows

Length 8 px

Width 3 px

Figure 25.8: Preferences window - Vector Editor category - Common tab

Description of figure 25.8 on page 156:

Step width Some operation in the *Vector Editor* can be performed stepwise if a certain key is pressed when the operation started; see sections 21.6 on page 111 and 21.7 on page 112. Customize the step width for these operations here:

Rotate sets the rotation angle step width used when objects are rotated stepwise.

Scale sets the scale step width used when objects are scaled stepwise.

Slant sets the slant step width used when objects are slanted stepwise.

Vector import Customize the results of the import of vector graphics files here:

Flatness sets the maximum deviation for the conversion of cirles, ellipse, arcs and curves to lines. Smaller values result in higher accuracy but lower processing speed, higher values result in lower accuracy but higher processing speed.

Line start points Customize the appearance of the line start points here. Line start points are the small rectangles at the start point of each (multi-)line (see also the section 25.3.1 on page 150):

Size sets the size of the line start points in pixel.

Line end point arrows Customize the appearance of the small arrow heads at the end of each line segment (see also the section 25.3.1 on page 150):

Length sets the length of the arrow head at the line end point in pixel.

Width sets the width of the arrow head at the line end point in pixel.

Please find the remaining GUI elements in section 25.1 on page 146.

25.4 Messages View Category

In this category manage the behavior of the *Messages* view.

Description of figure 25.9 on page 158:

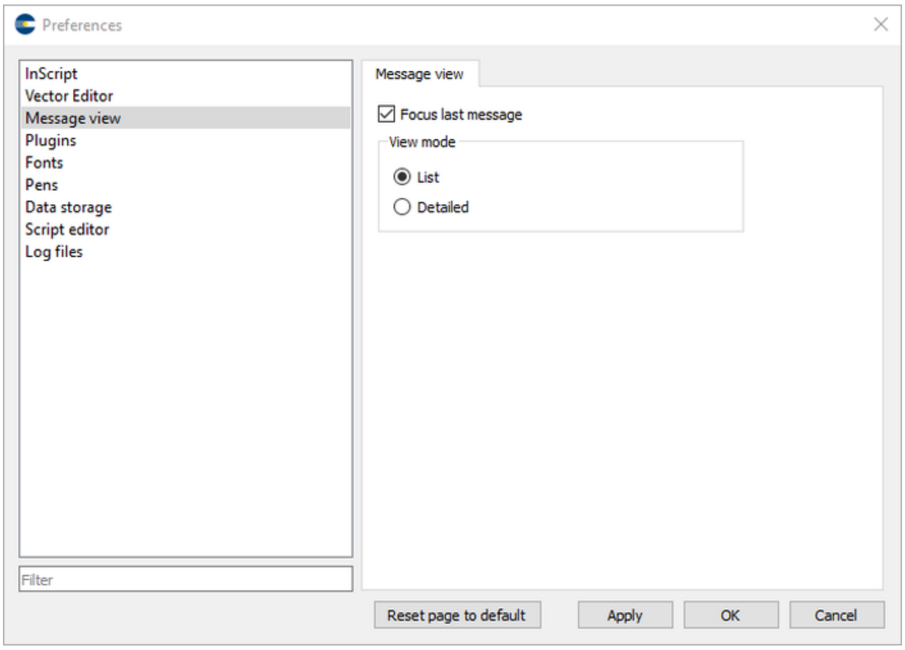
Focus last message always focuses, if checked, on the last message.

View mode selects the appearance of messages in the *Messages* view.

List shows all available messages in a list.

Detailed shows a detailed view of the currently active message.

Please find the remaining GUI elements in section 25.1 on page 146.



Message view

☑ Focus last message

View mode

● List

○ Detailed

Figure 25.9: Preferences window - Messages View category

25.5 Plugins Category

In this category manage InScript plugin search paths in a priority list.

Description of figure 25.10 on page 159:

The plugin search paths are listed on the right side. After a fresh InScript installation the list consist of the following entry:

- The `Plugins` subdirectory in the InScript `Documents` directory

Organize the **Search path** list with help of the following buttons:

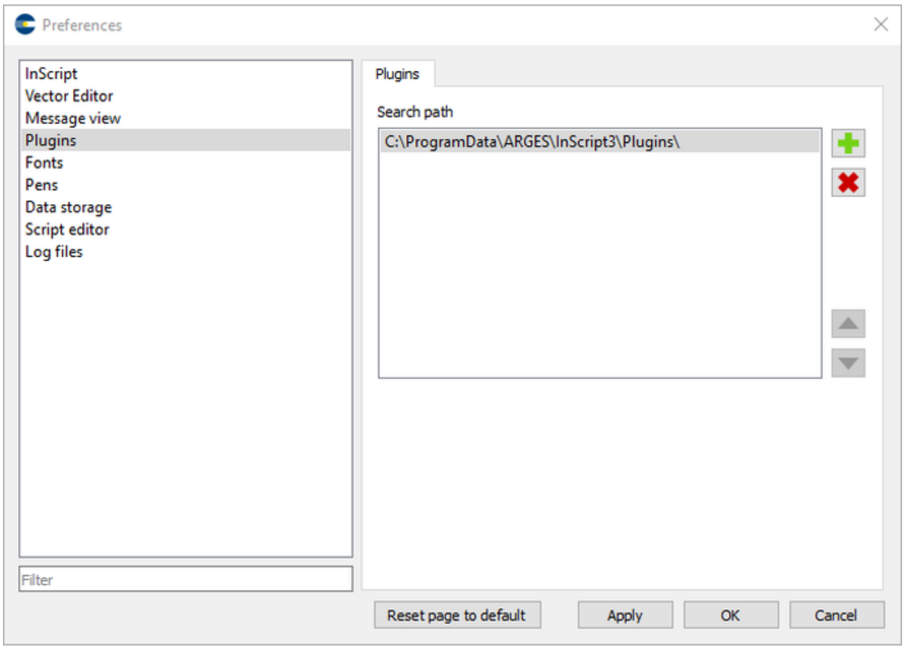
➕ adds a directory to the list.

✖ removes the selected directory from the list.

▲ moves the selected directory 1 entry up in the list, i.e. increases priority.

▼ moves the selected directory 1 entry down in the list, i.e. decreases priority.

Please find the remaining GUI elements in section 25.1 on page 146.



Search path

C:\ProgramData\ARGES\InScript3\Plugins\

Figure 25.10: Preferences window - Plugins category

25.6 Fonts Category

In this category manage InScript font search paths in a priority list.

Description of figure 25.11 on page 160:

The font search paths are listed on the right side. After a fresh InScript installation the list consist of following 2 entries:

- The `Fonts` subdirectory of the InScript `Documents` directory
- The `Fonts` subdirectory of the operating system InScript is running on e.g. `Windows\Fonts` on most Windows systems

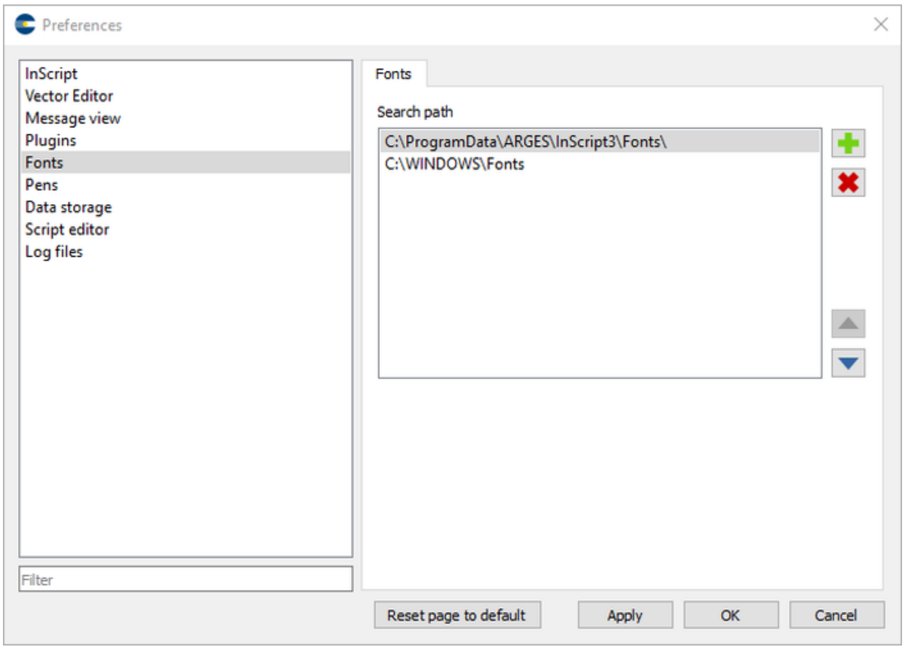
Organize the **Search path** list with help of the following buttons:

➕ adds a directory to the list.

✖ removes the selected directory from the list.

▲ moves the selected directory 1 entry up in the list, i.e. increases priority.

▼ moves the selected directory 1 entry down in the list, i.e. decreases prior-



Search path

C:\ProgramData\ARGES\InScript3\Fonts\
C:\WINDOWS\Fonts

Figure 25.11: Preferences window - Fonts category

ity.

Please find the remaining GUI elements in section 25.1 on page 146.

25.7 Pens Category

In this category manage InScript pen search paths in a priority list.

Description of figure 25.12 on page 161:

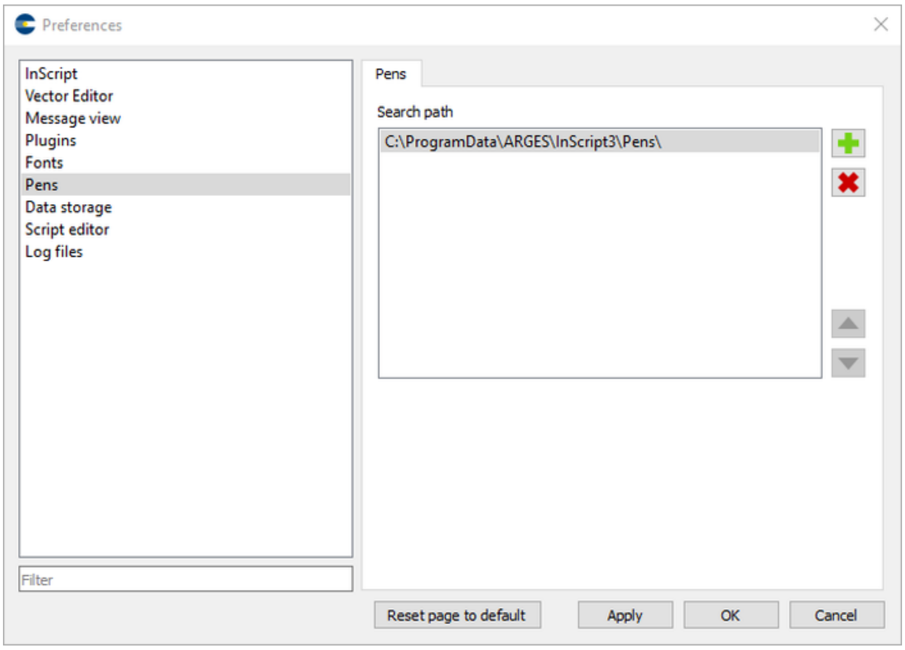
The pen search paths are listed on the right side. After a fresh InScript installation the list consist of the following entry:

- The `Pens` subdirectory in the InScript `Documents` directory

Please keep in mind that InScript tries to load pens first from the path where the job is loaded from. For more details see section 16.2 on page 80.

Organize the **Search path** list with help of the following buttons:

✚ adds a directory to the list.



Search path

C:\ProgramData\ARGES\InScript3\Pens\

Figure 25.12: Preferences window - Pens category

- ✖ removes the selected directory from the list.
- ▲ moves the selected directory 1 entry up in the list, i.e. increases priority.
- ▼ moves the selected directory 1 entry down in the list, i.e. decreases priority.

Please find the remaining GUI elements in section 25.1 on page 146.

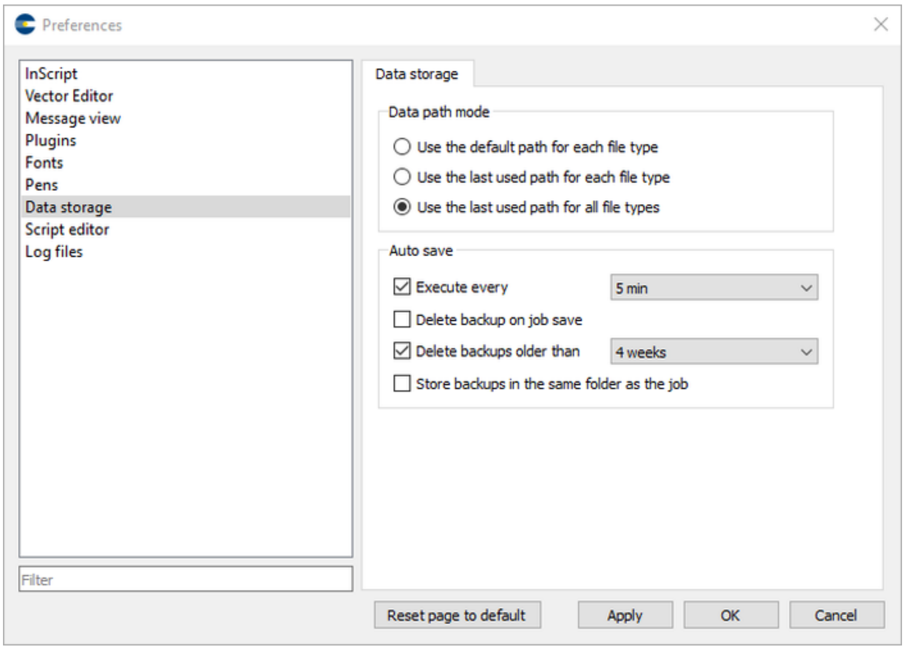
25.8 Data Storage Category

In this category customize how data paths for distortion, font, job, layout, pen, raster graphic and vector graphic files are handled.

Description of figure 25.13 on page 162:

Data path mode selects InScript's handling of the data path.

Use the default path for each file type always uses, if selected, the default paths for all file types.



Data storage

Data path mode

○ Use the default path for each file type

○ Use the last used path for each file type

● Use the last used path for all file types

Auto save

☑ Execute every 5 min

□ Delete backup on job save

☑ Delete backups older than 4 weeks

□ Store backups in the same folder as the job

Figure 25.13: Preferences window - Data Storage Category

The default paths are dedicated directories for each file type which are used to load data from or to store data to. These directories never change during the InScript runtime in this mode. In most cases the default paths are located in the InScript `Documents` directory.

Use the last used path for each file type uses, if selected, the last used path for each file type until the session ends and uses it for the next file load/save operation of the same file type. On InScript startup the last used paths for all file types will be initialized with the default paths (see above).

Use the last used path for all file types uses, if selected, the last used path of all file load/save operations until the session ends and uses it for the next file operation, regardless of the file type.

Auto save defines if and how auto save for job files will be used.

Execute every When active, all changed jobs of a controller will be saved automatically after the given period of time.

Delete backup on job save When active, the job backup file will be deleted, when the job will be saved.

Delete backups older than When active, all job backup files will be deleted on InScript start, which are older than the given age.

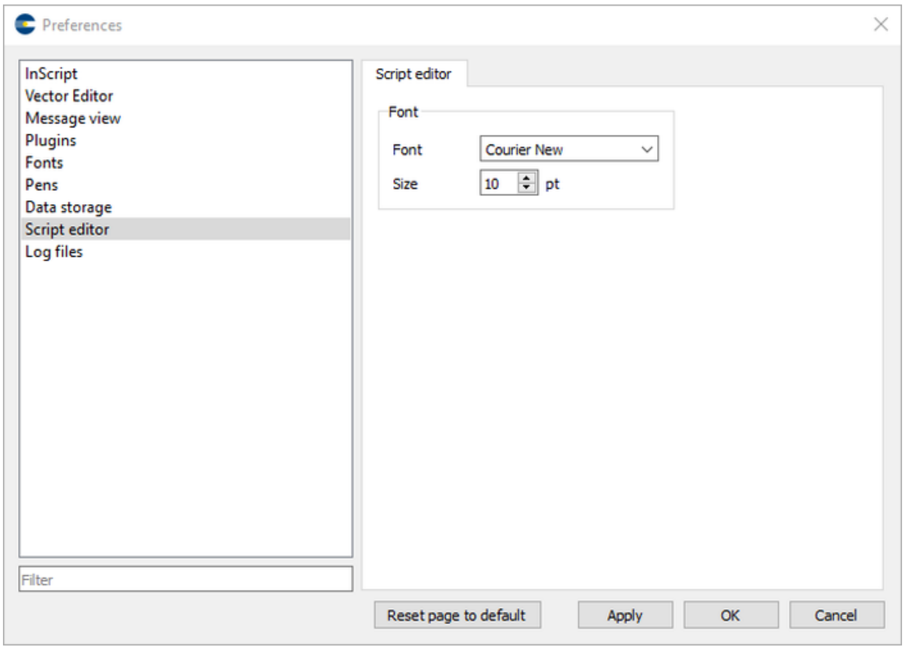
Store backups in the same folder as the job When active, all job backup files will be stored in the same folder as the job.

Otherwise the AutoSave subdirectory of the application data path will be used.

Please find the remaining GUI elements in section 25.1 on page 146.

25.9 Script Editor Category

This category defines the appearance of the script editor.



Font: Courier New

Size: 10 pt

Figure 25.14: Preferences window - Script Editor category

Description of figure 25.14 on page 163:

Font defines the font and its size that is being used in the script editor.

Font selects the font.

Size selects the font size.

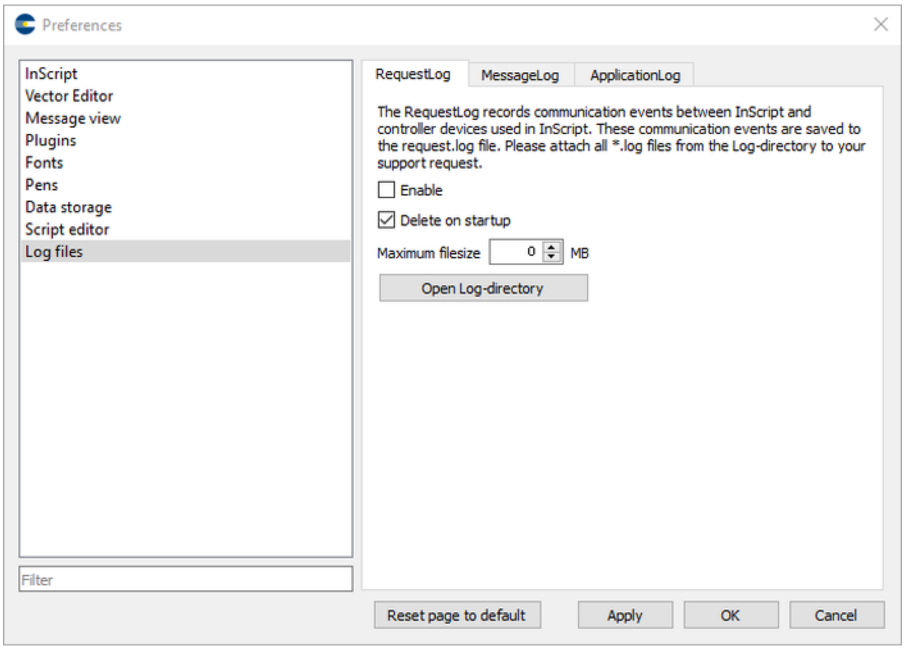
Please find the remaining GUI elements in section 25.1 on page 146.

25.10 Log Files Category

InScript collects information on communication events, messages and InScripts behavior at runtime in log files. When requesting support these files are usually very useful. InScript saves all log files in the `Log` subdirectory of the InScript `Documents` directory.

25.10.1 RequestLog Tab

The *RequestLog* records communication events between InScript and controller devices used in InScript. These communication events are saved to the `request.log` file. Please attach this file when requesting support at ARGES.



The RequestLog records communication events between InScript and controller devices used in InScript. These communication events are saved to the request.log file. Please attach all *.log files from the Log-directory to your support request.

□ Enable

☑ Delete on startup

Maximum filesize 0 MB

Figure 25.15: Preferences window - Log Files category - RequestLog tab

Description of figure 25.15 on page 164:

Enable enables, if checked, recording the `request.log` file.

Delete on startup deletes, if checked, the `request.log` file on every InScript start up.

We recommend to select this option as the `request.log` file can get very large.

Maximum filesize limits the filesize of the `request.log` file to the given value.

A value of `0` disables the limitation.

Open Log-directory opens the `Log`-directory.

Please attach all ∗.log files from the Log-directory to your support request.

Please find the remaining GUI elements in section 25.1 on page 146.

25.10.2 MessageLog Tab

The *MessageLog* records all messages that occur during InScript runtime. All messages shown in the *Messages* view are saved to the `message.log` file.

Description of figure 25.16 on page 166:

Delete on startup if checked then this deletes the `message.log` file on every InScript start up.

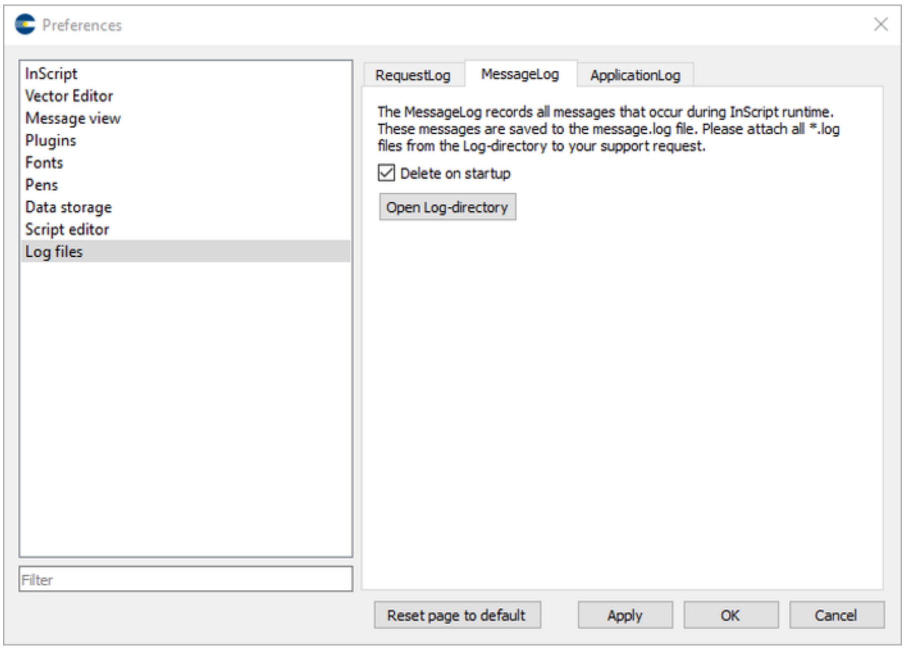
Open Log-directory opens the `Log`-directory.

Please attach all ∗.log files from the Log-directory to your support request.

Please find the remaining GUI elements in section 25.1 on page 146.

25.10.3 ApplicationLog Tab

The *ApplicationLog* collects information about the InScript application behavior at runtime. This includes information about the InScript version, the storage location where configuration files, plugins, help files, messages and



The MessageLog records all messages that occur during InScript runtime. These messages are saved to the message.log file. Please attach all *.log files from the Log-directory to your support request.

☑ Delete on startup

Figure 25.16: Preferences window - Log Files category - MessageLog tab

translations were loaded from, command line options, basic screen properties and information about the OpenGL capabilities of the computer system InScript is running on.

The *ApplicationLog* is saves this information to the `application.log` file.

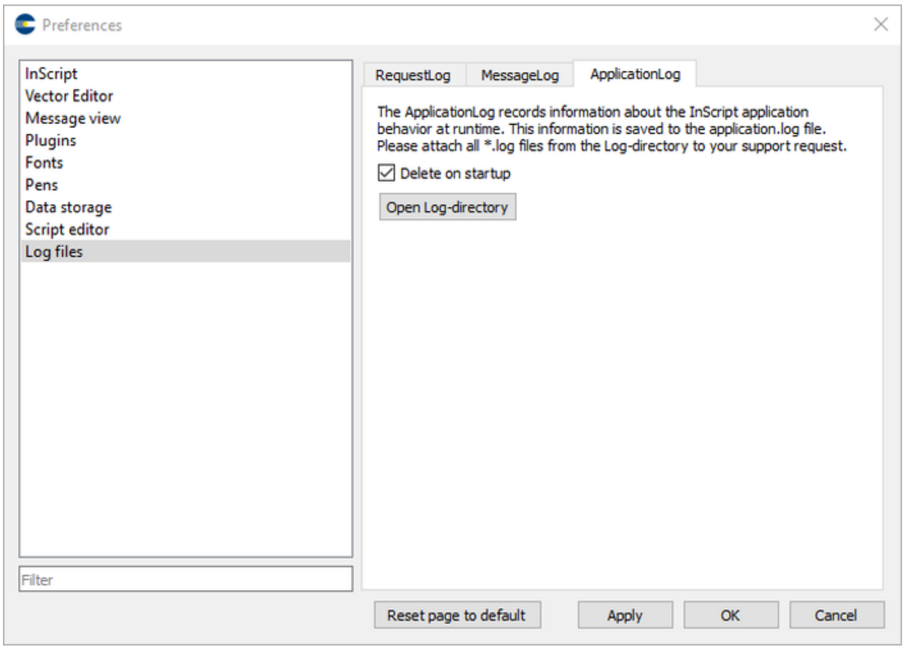
Description of figure 25.17 on page 167:

Delete on startup if checked then this deletes the `application.log` file on every InScript startup.

Open Log-directory opens the `Log`-directory.

Please attach all ∗.log files from the Log-directory to your support request.

Please find the remaining GUI elements in section 25.1 on page 146.



The ApplicationLog records information about the InScript application behavior at runtime. This information is saved to the application.log file. Please attach all *.log files from the Log-directory to your support request.

☑ Delete on startup

Figure 25.17: Preferences window - Log Files category - ApplicationLog tab

26 Controllerservices

The controllerservices are functions that manage controller dependent activities. This means, you can get information for our support, install a new firmware, shut down or reboot the controller and more.

26.1 Connecting/Disconnecting The Controllerservices

Procedure

1. In the *Navigator* view, click the controller's IP address.
2. In the *Node Properties* view, click **Connect Controllerservices** or **Disconnect Controllerservices** respectively.

   The Controllerservices connect to the controller and the dialog is enabled; see figure 26.1 on page 169.

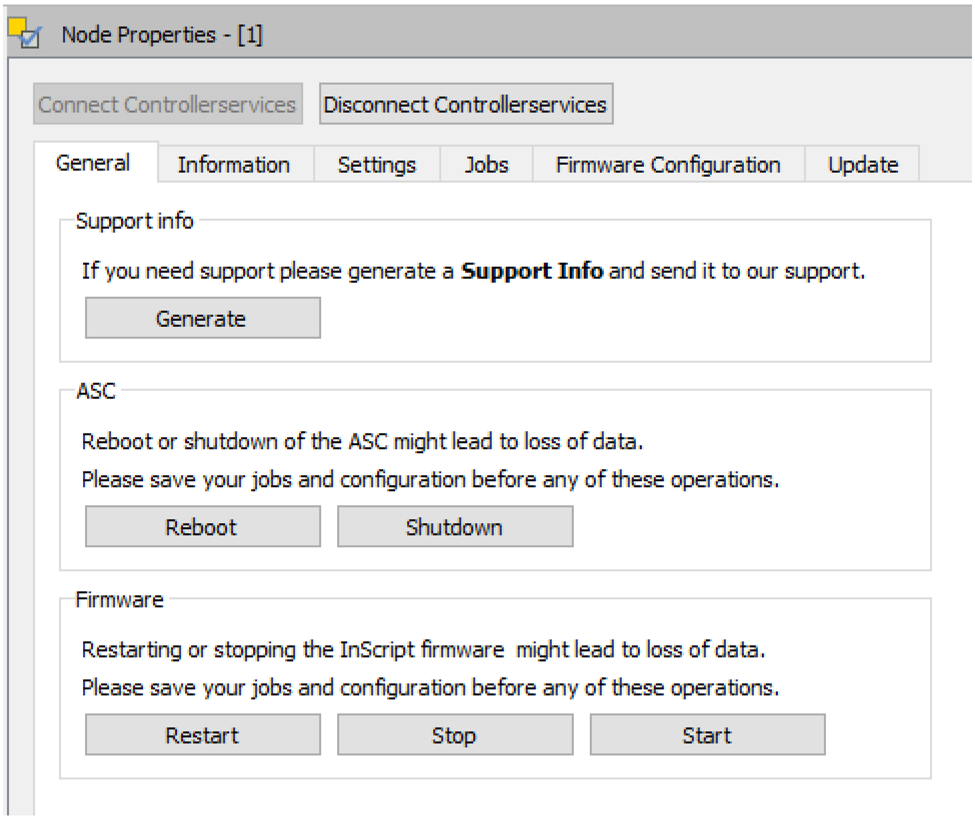
26.2 Gathering Information For Support

For support purposes you can retrieve information from the controller and save this information into a file. This information is more comprehensive than that gathered on the *Information* tab in the *System information* group.

> **TIP**
>
> When asking for support always attach this file, please.

Procedure

1. On the *General* tab, in the *Firmware* group, click **Stop** to stop the firmware.



If you need support please generate a **Support Info** and send it to our support.

Reboot or shutdown of the ASC might lead to loss of data.
Please save your jobs and configuration before any of these operations.

Restarting or stopping the InScript firmware might lead to loss of data.
Please save your jobs and configuration before any of these operations.

Figure 26.1: Controller services window - General tab

2. In the *Support info* group, click **Generate**.

   You will be informed, that the task may take some time and asked if you want to continue.

3. Click **Yes**.

   The support information will be retrieved. Depending on your system this might take a few minutes.

4. Select the location where to save the support info file and click **Save**.
5. After the file has been saved, click **Start** in the *Firmware* group.

26.3 Rebooting The Controller

> **TIP**
>
> Note, that this procedure breaks the connection to the controller and unsaved data might be lost.

Procedure

- On the *General* tab in the *ASC* group, click **Reboot**.

26.4 Shutting Down The Controller

If you want to switch off the controller then shut it down first.

> **TIP**
>
> Note, that this procedure breaks the connection to the controller and unsaved data might be lost.

Procedure

- On the *General* tab in the *ASC* group, click **Shutdown**.

26.5 Restarting The Firmware

> **TIP**
>
> Note, that this procedure breaks the connection to the firmware and unsaved data might be lost.

Procedure

- On the *General* tab in the *Firmware* group, click **Restart**.

26.6 Stopping The Firmware

If you want to delete or upload the controller configuration `flash.xml` then stop the firmware first. Otherwise the firmware might overwrite the `flash.xml` again.

> **TIP**
>
> Note, that this procedure breaks the connection to the firmware and unsaved data might be lost.

Procedure

- On the *General* tab in the *Firmware* group, click **Stop**.

26.7 Starting The Firmware

If you have uploaded or deleted the controller configuration `flash.xml`, or if the firmware has crashed and does not start by itself, then start the firmware.

Procedure

- On the *General* tab in the *Firmware* group, click **Start**.

26.8 Viewing The Temperature

The *Information* tab shows the current temperature of board and CPU.

Procedure

- On the *Information* tab in the *Current temperatures* group, read the temperatures of *Board* and *CPU*.

26.9 Cleaning Up The Filesystem

This procedure cleans up space on the ASC controller. This removes temporary files from folders `/tmp` and `/var/core` on the controller. Note, that this might do nothing, if no unnecessary files are on the controller.

Procedure

- On the *Information* tab in the *Filesystem* group, click **Cleanup**.

  The available space will update automatically

26.10 Viewing The System Information

- On the *Information* tab in the *System information* group, read the gathered information.

26.11 Setting Network Information

Setting the network configuration to integrate the controller in your local setup.

ETH0 is the port which the controller connects to your network.

ETH1 is the port the WFI-OCT connects to your network, if you have WFI-OCT attached.

Procedure

1. On the *Settings* tab on the *Network* subtab, enter the IP-address, Netmask and Gateway.
2. Click **Set**.
3. Reboot the controller; see section 26.2 on page 169.

26.12 Setting The Hostname

Procedure

1. On the *Settings* tab on the *Network* subtab, enter the hostname of the controller and click **Set hostname**.
2. Reboot the controller; see section 26.2 on page 169.

26.13 Setting The Date To Current Date

This procedure sets date and time on the controller to the current date and time on the PC from where you started the controllerservices.

If date or time are wrong on the controller then set them to the correct values. Date and time should be correct as this is most certainly helpful when a support case occurs.

Procedure

1. On the *Settings* tab on the *Date* subtab, click **Set current date**.

   Note, that the controller date will not be automatically updated in the text field.

2. Click **Update date** to view the date on the controller.

26.14 Autoload And Autoselect A Job

Autoload jobs are jobs, which are loaded when the ASC or the firmware is started. These jobs are stored on the ASC.

One of these *autoload jobs* can be automatically selected for job execution after it is loaded.

The *Autoload Jobs* list lists these jobs.

Procedure

1. On the *Jobs* tab, add a job from the local file system by clicking **Add**.
2. In *Autoselect*, select the job, that shall be autoselected for job execution on start.

Additional actions on the *Jobs* tab are:

- Jobs can be downloaded from the ASC by clicking **Download**
- Jobs can be removed by clicking **Remove**.

> **TIP**
>
> Files are deleted from the ASC's file system permanently. Download a job before removing it.

26.15 Uploading The Controller Configuration

The `flash.xml` file is used to save the configuration of the InScript firmware. You can download and upload `flash.xml` files to save the configuration and restore it later or to copy configurations between machines.

Note, that when uploading a configuration file to the controller, it will be automatically renamed to `flash.xml`.

Procedure

1. Stop the firmware; see 26.5 on page 170.

   Otherwise the firmware might overwrite the `flash.xml` again.

2. On the *Firmware Configuration* tab, click the ... button.
3. Select the file that shall be uploaded to the controller.

   > **NOTICE**
   >
   > Wrong file content
   >
   > damages the configuration of the firmware; e.g. the configuration of the devices.
   >
   > - Upload only files which have been downloaded from the controller or have been provided by Novanta for this purpose.

4. Click **Upload Flash.xml**.
5. Wait until the upload is complete.
6. Start the firmware; see 26.6 on page 171.

26.16 Downloading The Controller Configuration

The `flash.xml` file is used to save the configuration of the InScript firmware. You can download and upload `flash.xml` files to save the configuration and and restore it later or to copy configurations between machines.

Procedure

1. On the *Firmware Configuration* tab, click **Download Flash.xml**.
2. Select a location where to save the file.
3. Click **Save**.

26.17 Deleting The Controller Configuration

The `flash.xml` file on the controller is the file where the configuration of the firmware is saved, e.g. the configuration of the devices.

Procedure

1. Stop the firmware; see 26.5 on page 170.

   Otherwise the firmware might overwrite the `flash.xml` again.
2. On the *Firmware Configuration* tab, click **Delete Flash.xml**.
3. Wait until the deletion is complete.
4. Start the firmware; see 26.6 on page 171.

26.18 Updating Controller Or Scan Head

26.18.1 Downloading The Update File(s)

Requirements

- The controller to be updated is an ARGES System Controller, ARGNET Series (aka Gen4).
- The controller is already installed as described in the *ARGES System Controller, ARGNET Series – User Manual*.
- The InScript software is already installed as described in the *InScript Software – User Manual*.
- The scan head is already installed as described in the *Scan Head – Manual*.

Procedure

1. Visit *novantaphotonics.com*.
2. Click **Products**.
3. In section *Controllers & Software* click **Software**.
4. Click **InScript**.
5. Scroll down and click **View InScript Manuals**.
6. Enter your contact details and click **Submit**.
7. If you want to update the controller, then:
   a) From section *InScript Packages* download the *InScript software*.
   b) From section *Controller Update Package* download one of the packages according to your ASC hardware. Find information about your ASC hardware on the ASC's type label.
8. If you want to update the scan head, then download the file from section *Scanhead FPGA*.

26.18.2 Updating The Controller

Requirements

- You have downloaded the necessary update file(s) as described in 26.18 on page 175.
- The controller and the InScript PC are switched on.

Procedure

1. Install the InScript software as described in the *InScript Software - User Manual*.
2. Start the InScript software.
3. In the InScript software in the *Navigator* view, click the IP-address of the controller that you want to update.
4. In the *Node Properties* view, click **Connect Controllerservices**.
5. On the *Update* tab, click the **System** subtab.
6. Click the ... button and browse to the directory where you saved

the downloaded file(s).

7. Select the ∗**.ZIP** file and click **OK**.

   Which file you have selected is displayed in *Update file*.

   > **NOTICE**
   >
   > Power loss during the update
   >
   > results in the controller becoming inoperable and having to be sent in for repair.
   >
   > - Do not shut down the controller or remove the external power from the controller during the process.
   >
   >   The update will take up to 30 minutes depending on how much components are going to be updated.

8. Click **Update**.

   Once the update is completed, you will receive the message that *The system update was successful* or that *The system update failed*.

9. Click **OK**.

   You will be asked if the update protocol should be saved.

10. If the system update failed, then it is mandatory to save the update protocol.

    The update protocol helps Novanta support to determine why the update failed.

    – OR –

    If the system update was successful, then it is optional to save the update protocol, but we recommend saving the update protocol anyway.

11. This step of rebooting the controller, and finishing the update process that way, depends on the type of controller you have.

    a) If you have an ASC-1 ARGNET series controller then switch off the controller, disconnect the power cable, wait for 5 seconds, connect the power cable and switch on the controller again.

– OR –

If you have another type of controller (ASC-2 or ASC-6 ARGNET series) reboot the controller via the Controllerservices: On the *General* tab in the *ASC* group, click **Reboot**.

26.18.3 Updating The Scan Head

Requirements

- You have downloaded the necessary update file as described in 26.18 on page 175.
- The controller and the InScript PC are switched on.
- The scan head is connected to the controller and switched on.

Procedure

1. Start the InScript software.
2. In the InScript software in the *Navigator* view, click the IP-address of the controller to which the scan head you want to update is connected.
3. In the *Node Properties* view, click **Connect Controllerservices**.
4. On the *Update* tab, click the **Firmware/FPGA** tab.
5. Click the ... button and browse to the directory where you saved the downloaded file(s).
6. Select the ∗**.SWU** file and click **OK**.

   Which file you have selected is displayed in *Firmware / FPGA Updatefile*.

> **NOTICE**
>
> Power loss during the update
>
> results in the scan head becoming inoperable and having to be sent in for repair.
>
> - Do not shut down the scan head and the controller or remove

> the external power from the scan head and the controller during the process.
>
> The update will take about 20 minutes.

7. Click **Install**.

   Once the update is completed, you will receive the message that *The system update was successful* or that *The system update failed*.

8. This step of rebooting the scan head, and finishing the update process that way, depends on the scan head's power supply.

   If the scan head is supplied with power via the controller, then reboot the controller via the Controllerservices: On the *General* tab in the *ASC* group, click **Reboot**.

   – OR –

   If the scan head is supplied with power via an external power supply, then switch off the scan head, wait for 5 seconds, switch on the scan head again and restart the InScript firmware via the Controllerservices: On the *General* tab in the *Firmware* group, click **Restart**.

27 Timed Signal Streams

27.1 Overview

27.1.1 What are Timed Signal Streams?

The concept of sending calculated or measured data with timing information from the firmware to clients (as e.g. InScript) is called Timed Signal Stream, or in short - TSS.

Examples: calculated line preview, sample data, events like plc state changes, sysmessages

- Transported data may be either isochronous (i.e. with a given fixed sample rate) or asynchronous (i.e. single events but every event with a timestamp).
- Data source is somewhere in the firmware always.
- Data clients (consumers) may be in the firmware or externally (ControllerLib, InScript)
- Each data source may have 0..n clients.
- Each client can connect to an arbitrary number of sources

27.1.2 Properties

- TimedSignalStreams (TSS) were distinguished by a unique name, the so called channelType
- Each channelType has a coordinate system (CS)
- The coordinate system defines the encoding of the data in the stream
- Each coordinate system has 1..n Axes

- Clients can request a list of available channelTypes and their properties
- Clients can register/unregister to one or more certain channelTypes
- Each registered connection will get a unique channel connection ID

27.1.3 Available channel types and coordinate systems

Which channel types (CT) and coordinate systems (CS) are available depends on the fw implementation. Thus, two controllers with different firmware versions may support different CT's and CS's.

To see what CT's and CS's a controller support perform the following steps:

1. Open a *Vector Editor* view or click on a *Vector Editor* view that already exists, to make it the current view.
2. Press the key combination **Ctrl** + **T**. This will generate some TSS information in the InScript `application.log`.
3. In the main menu, click **File** > **Explore documents**.

   This opens the InScript documents path in a *Windows Explorer* window.
4. Double click the `log` subdirectory to open it.
5. Open the file `application.log` with a text editor.

You will find the generated information at the end of the file, e.g.:

```
2020-09-30 15:43:26  (26636.953s)
Info: The controller ASC#172.29.227.235:1610 supports the following timed
   signal streams:
   Channeltype 1
      Name:                    TrackedScannerPositionXYZ
      GetCoordinateSystemName: mmCartesianSample
         Axis 0: X
         Axis 1: Y
         Axis 2: Z
```

```
Channeltype 2

   Name:                    TrackedLaserSwitchingTime

   GetCoordinateSystemName: digital16bit

      Axis 0: Gate

      Axis 1: Unused

      Axis 2: Unused

      Axis 3: Unused

      Axis 4: Unused

      Axis 5: Unused

      Axis 6: Unused

      Axis 7: Unused

      Axis 8: Unused

      Axis 9: Unused

      Axis 10: Unused

      Axis 11: Unused

      Axis 12: Unused

      Axis 13: Unused

      Axis 14: Unused

      Axis 15: Unused

Channeltype 3

   Name:                    CommandedScannerPositionXYZ

   GetCoordinateSystemName: mmCartesianSample

      Axis 0: X

      Axis 1: Y

      Axis 2: Z

Channeltype 4
```

```
      Name:                     CommandedLaserSwitchingTime

      GetCoordinateSystemName: digital16bit

         Axis 0: Gate

         Axis 1: Unused

         Axis 2: Unused

         Axis 3: Unused

         Axis 4: Unused

         Axis 5: Unused

         Axis 6: Unused

         Axis 7: Unused

         Axis 8: Unused

         Axis 9: Unused

         Axis 10: Unused

         Axis 11: Unused

         Axis 12: Unused

         Axis 13: Unused

         Axis 14: Unused

         Axis 15: Unused

   Channeltype 5

      Name:                     LineDataCartesianXY

      GetCoordinateSystemName: mmCartesianLine

         Axis 0: X

         Axis 1: Y

   Channeltype 6

      Name:                     SyncMessages

      GetCoordinateSystemName: smPayload
```

```
         No Axisnames found

   Channeltype 7

      Name:                    SysMessages

      GetCoordinateSystemName: text

         No Axisnames found
```

27.1.4 Data Rates

TimedSignalStreams create quite high data volumes. Usually TSS data on the controller will be sampled with 200 KHz (ASC4). That means that 200000 samples per second are generated by the controller and transmitted to the InScript client. This data rate still has to be be multiplied by the sample size.

For example the Cartesian sample layers have a sample size of 3 float values, thus $3 * 4 = 12$ bytes. According to this, these layers generate the following amount of data per time:

```
      200 000 sample/s * 12 bytes/sample

=   2 400 000 bytes/s

= 144 000 000 bytes/min

=     140 625 kB/min

=         137 MB/min
```

The good news is, for now there is only one coordinate system with a higher sample size, which is wfiPayload.

The bad news is, there may be lots of TSS channelTypes, especially for the mmCartesianSample coordinate system. Thus the data rates shown above has to be multiplied by the number of the used channels!

27.2 Timed Signal Streams in InScript

For now TSS are used in InScript in the VectorEditor view. Several channel-Types or combinations of them are mapped to layers in the VectorEditor.

When working with these VectorEditor layers always keep the high data rates in mind! The most important factor in this regard is the memory and the performance of your graphics card. The older and more low-end your used graphics card is, the higher the probability of problems occurring! If you notice that your graphics card is slowing down when drawing TSS layers, consider upgrading this. If this is not possible, reduce the number of used TSS channels / layers.

Because at the moment many users still use older graphics hardware, we have decided to limit the sample count for the time being. For almost all TSS layers the maximum sample count is 24 million. The only exception is the WFI topology layer, where the limit is somewhat higher, namely 33.5 million (2^25). If you now relate this value to the sample rate, you will see that this limitation will lead to a maximum recording duration of

```
24 million samples / 200000 samples / s = 120 s
```

If this number is reached, the stored values will be overridden from the start of the buffer where it is stored in (ringbuffer principle). So if the amount of data of your TSS layer is higher than 24 million samples you will se the last 24 million samples always, or in temporal terms, the last two minutes of this data.

If this limitation restricts you too much, there is an alternative way of working with TimedSignalStreams. In the InScript software family there is a tool called TssRecorder which is made for recording TimedSignalStream in files. The file format is well documented for each channel (coordinate system). All you have to do is create your own software to evaluate the recorded streams.

27.2.1 LineDataCartesianXY layers

The ChannelType LineDataCartesianXY is used by the JobPreview of the VectorEditor. For more information about that layer see 21.9.3 on page 117

27.2.2 Cartesian sample layers

All channels with the coordinate system ”mmCartesianSample” will be mapped to VE layers automatically. In newer firmware versions usually there are the channel types *CommandedScannerPosition* and *TrackedScannerPosition* available with the coordinate system mentioned above.

Cartesian sample layers can be configured via the file InScript.ini as following:

```
[StreamName]

sampleColor=\#RRGGBBAA

sampleSize=integer
```

Example:

```
[CommandedScannerPositionXYZ]

sampleColor=\#228b2254

sampleSize=6
```

#RRGGBBAA denotes a hexadecimal RGBA value. In the example the alpha value is 54 hexadecimal which is 84 decimal. Thus the alpha value is 255 : 84 = 33 %.

As the name says the sampleSize can be used to adjust the visible size of a single sample point. The sampleSize value should be choosen very careful, because higher values may slow down the render speed of the VectorEditor.

For more information about that layers see 21.9.4 on page 118 and 21.9.5 on page 119.

27.2.3 Combination layers

The VectorEditor is able to show combined information from certain channel types, which is called combination layers.

Posible combination are for now:

- One axis of a channel with digital16bit CS with one channel with mmCartesianSample CS. This combination is usually used to visualize the position, where laser switching events took place.
- One channel with degreeCelsius CS with another channel with mmCartesianSample CS. This combination is usually used to visualize the temperature at certain positions as false-colour display. The minimum and maximum temperature as well as the colors belonging to them can be configured via the InScript.ini file.

The channel combinations

- CommandedLaserSwitchingTime.Gate@CommandedScannerPositionXYZ and
- TrackedLaserSwitchingTime.Gate@TrackedScannerPositionXYZ

are predefined, i.e. if all involved channel types and coordinate systems are supported, the combination layer will be shown in the VectorEditors layer list.

For more information about that layers see 21.9.6 on page 119 and 21.9.7 on page 119

Other channel combinations are configurable via InScript.ini, [VectorEditor], Layers\TssCombinations

Examples:

```
Layers\TssCombinations=PyrometerTemperature.temperature\@TrackedScannerPositionXYZ
```

Multiple entries have to be separated with a comma, follwed by space, e.g. ”Layers\\TssCombinations=entry1, entry2, entry3”

For combinations of isochronous and asynchronous timing types the asynchronous values will be (linear) interpolated to match the isochronous ones.

The LaserSwitchingTime.Gate@ScannerPositionXYZCartesian layers can be configured via the file InScript.ini as following:

```
[CombinationLayerName]

symbolColor=\#RRGGBBAA

symbolRadius=integer
```

Example:

```
[CommandedLaserSwitchingTime.Gate%40CommandedScannerPositionXYZ]

symbolColor=\#196819ff

symbolRadius=7
```

#RRGGBBAA denotes a hexadecimal RGBA value.

The symbolRadius setting can be used to adjust the radius of the symbol of the event (a small circle for gate on and a small hatch cross for gate off events). The symbolRadius value should be choosen very careful, because higher values may slow down the render speed of the VectorEditor.

27.2.4 WFI topology layers

For all controllers with the WFI PCIe hardware extension there are addional layers available.

The combination of the measured height value from the WFI with the position data from the scan head is called Topology Data. For that, there is no dedicated view available. Instead of this, there are additional layers in the InScript3 VectorEditor, the ”WFI Topology” layer, which contains the topology data itself and the ”WFI Topology Legend” layer which shows the scale (which height value is assigned to a specific color).

Please keep in mind, that these layers are visible only when the controller has an activated WFIBase device.

From the TSS point of view the ”WfiData” stream with the coordinate system ”wfiPayload” is used for that feature. Although the WFI Views Plugin is not needed for these layers it may provide additional functionality. In the current version of the WFI Views Plugin there is a menu function which enables the export of the data of the ”WFI Topology” layer as CSV file.

The WFI topology layers can be configured via the file InScript.ini as following:

```
[WFI_Topology]

sampleSize=integer

manualScaling=true or false

manualZMinValue=float

manualZMaxValue=float
```

Example:

```
[WFI_Topology]

sampleSize=6

manualScaling=false

manualZMinValue=0

manualZMaxValue=20
```

Just as with the Cartesian sample layers, the sampleSize can be used to adjust the visible size of a single sample point. The sampleSize value should be choosen very careful, because higher values may slow down the render speed of the VectorEditor.

If manualScaling is true, the legend will use the manualZMinValue and manualZMaxValue as range, otherwise the minimum and maxmimum Z value from the data.

For more information about that layers see 21.9.8 on page 120 and 21.9.9 on page 121.

28 Timed Signal Stream Recorder

28.1 Overview

28.1.1 What is the TSS Recorder View?

The TimedSignalStream (TSS) Recorder is a view int the InScript software. As its name suggests, it is designed for recording timed signal streams. Timed signal streams are streams of various data which can be requested from ARGES ASC controllers. This tool provides a very easy way to capture these streams and store them in files (binary or as CSV).

28.1.2 What is a TimedSignalStream

The concept of sending calculated or measured data with timing information from an ARGES system controller to clients is called **TimedSignalStream**.

Examples: calculated line preview, sample data, events like plc state changes, sysmessages

Transported data may be either isochronous (i.e. with a given fixed sample rate) or asynchronous (i.e. single events but every event with a timestamp).

28.1.3 Properties of TimedSignalStream

- TimedSignalStreams are distinguished by the channelType (a unique name)
- Each channelType has a coordinate system (CS)
- The coordinate system defines the encoding of the data in the stream
- Each coordinate system has 1..n Axes

28.1.4 How does the TSS Recorder work?

When the recording is startet, the TssRecorder registers as a client for each selected channel type. It receives the data sent by the controller firmware for these channels and writes it to data files with the extension *.bin* (when binary data should be written) or *.csv*, when CSV data should be written. The file names follow the scheme

```
<ChannelType>_<FileNumber>.bin
```

e.g.

```
CommandedScannerPositionXYZ_000001.bin
```

For now the file number uses 6 digits, i.e. it starts with 0 and may run up to 999999.

Because the received data may have a significant large size (e.g. all channels with the mmCartesianSample coordinate system will receive about 137.3 MB per minute at 200 khZ!) the file size of the data file is limited. The default maximum file size is 256 MB.

The TssRecorder checks for each new data packet it receives, if it fits into the defined maximum file size. If this is the case, the packet will be appended to the currently used data file. Otherwise (if writing the new packet to the currently used file would exceed the defined maximum file size) the currently used data file will be closed, the file counter increased and a new data file (with a new filename with increased fileNumber) will be opened.

28.2 Supported Streams

28.2.1 CommandedLaserSwitchingTime

The CommandedLaserSwitchingTime stream (and its corresponding files) contain information when the laser should be switched on. A sample in this stream is a transition from LASER ON to LASER OFF or the other way around.

28.2.2 TrackedLaserSwitchingTime

The TrackedLaserSwitchingTime stream (and its corresponding files) contain information when the laser was switched on. A sample in this stream is a transition from LASER ON to LASER OFF or the other way around.

28.2.3 CommandedScannerPositonXYZ

The CommandedScannerPositionXYZ stream (and its corresponding files) contain information about the commanded cartesian position of the scanner. A sample in this file consists of float values of the x, y and z position. The timestamp of each sample can be calculated from the information in the header of a file.

28.2.4 TrackedScannerPositionXYZ

The CommandedScannerPositionXYZ stream (and its corresponding files) contain information about the commanded cartesian position of the scanner. A sample in this file consists of float values of the x, y and z position. The timestamp of each sample can be calculated from the information in the header of a file.

28.2.5 DG_BB_Instriumentation

The DG_BB instrumentation stream contains additional information coming from the scan head. This information can be useful when you need support, but it is encrypted. You cannot evaluate this information.

28.2.6 SyncMessages

The SyncMessages stream (and its corresponding files) contain synchronization messages used internally by the controller fw.

28.2.7 pageUserDocStreamPyrometerTemp

Temperaturedata in °C from a connected pyrometer.

28.2.8 SysMessages

The SysMessages stream (and its corresponding files) contain all the system messages which occurred on the system. Each sample has a timestamp and the message itself.

28.2.9 WfiData

The WfiData stream (and its corresponding files) contain all wfi payload data from a WFIBase device of the system. Each sample consists of a timestamp, the cartesian position of the scanner where the measurement took place (xy) and the measured distance (z). Additionally is contains the WedgePos which stands for the measurement range and the raw value of the measurement.

28.3 Using the TssRecorder

- In the Navigator choose the controller for which you want to record Timed Signal Streams
- Open the TssRecorder View
- In the TssRecorder View choose the outputdirectory (storage path)
- Choose, whether you want to write binary data or CSV-data
- In the TssRecorder View select the streams you want to record
- Press **Start recording**
- Start the job you want record Timed Signal Streams for
- Wait for the job to finishe
- Press **Stop recording**

Now all data of the recording is stored and can be copied and/or processed further.

ATTENTION:

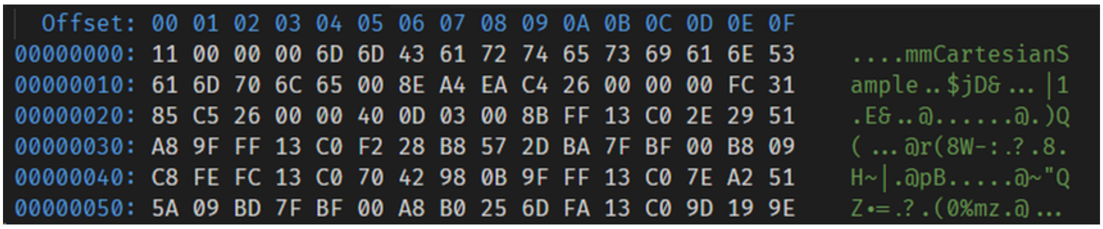
Do not store the data of more than one job execution in one data set. This might complicate the data analysis unnecessary! Instead of that press ”Stop Recording” after each job execution and then copy and/or use the data from the storage path. After that start a new recording. Do so, because it is possible, that data will be written until the stop recording event.

28.4 Stream Data File Formats (Binary)

28.4.1 Overview

The files generated by the Timed Signal Stream Recorder are binary files. They consist of the name of the stream, an underscore a number (starting with 000000) and the extension ”.bin”. For example CommandedLaserSwitching-Time_000000.bin

Each file consists of an header and a various number of samples. The header has information about the used coordinatesystem, the start-timestamp of the first sample and the end-timestamp of the last sample. A detailed description of the header can be found in the description of every stream. Here is an example of the beginning of a CommandedScannerPositionXYZ-File:



```
  Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 11 00 00 00 6D 6D 43 61 72 74 65 73 69 61 6E 53    ....mmCartesianS
00000010: 61 6D 70 6C 65 00 8E A4 EA C4 26 00 00 00 FC 31    ample..$jD&...|1
00000020: 85 C5 26 00 00 40 0D 03 00 8B FF 13 C0 2E 29 51    .E&..@......@.)Q
00000030: A8 9F FF 13 C0 F2 28 B8 57 2D BA 7F BF 00 B8 09    (...@r(8W-:.?.8.
00000040: C8 FE FC 13 C0 70 42 98 0B 9F FF 13 C0 7E A2 51    H~|.@pB.....@~"Q
00000050: 5A 09 BD 7F BF 00 A8 B0 25 6D FA 13 C0 9D 19 9E    Z•=.?.(0%mz.@...
```

Figure 28.1: Hexdump of a CommandedScannerPositionXYZ file (partial)

28.4.2 CommandedLaserSwitchingTime

CommandedLaserSwitchingTime files contain the data of the stream with the same name.

Because all samples have the same size, the number of samples in the file can be calculated by (fileSize - headerSize) / sampleSize.

Table 28.1: CommandedLaserSwitchingTime - Header

Datatype	Data
Uint32_t	Length of the stream name (in this case ”12”)
N ∗ character	Coordinate system (in this case ”digital16bit”)
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file
Uint32_t	Frequency of the samples (in this case ”0”)

Table 28.2: CommandedLaserSwitchingTime - Sample

Datatype	Data
Uint64_t	Timestamp
Uint16_t	1 = Laser is switched on, 0 = Laser is switched off

28.4.3 TrackedLaserSwitchingTime

TrackedLaserSwitchingTime files contain the data of the stream with the same name.

Because all samples have the same size, the number of samples in the file can be calculated by (fileSize - headerSize) / sampleSize.

Table 28.3: TrackedLaserSwitchingTime - Header

Datatype	Data

Continued on next page

Table 28.3 – concluded from previous page

Datatype	**Data**
Uint32_t	Length of the stream name (in this case ”12”)
N ∗ character	Coordinate system (in this case ”digital16bit”)
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file
Uint32_t	Frequency of the samples (in this case ”0”)

Table 28.4: TrackedLaserSwitchingTime - Sample

Datatype	**Data**
Uint64_t	Timestamp
Uint16_t	1 = Laser is switched on, 0 = Laser is switched off

28.4.4 CommandedScannerPositonXYZ

CommandedScannerPositonXYZ files contain the data of the stream with the same name.

Because all samples have the same size, the number of samples in the file can be calculated by (fileSize - headerSize) / sampleSize.

Table 28.5: CommandedScannerPositonXYZ - Header

Datatype	**Data**
Uint32_t	Length of the stream name (in this case ”17”)
N ∗ character	Coordinate system (in this case ”mmCartesianSample”)

Continued on next page

Table 28.5 – concluded from previous page

Datatype	Data
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file
Uint32_t	Frequency of the samples (in this case ”200000”)

Table 28.6: CommandedScannerPositonXYZ - Sample

Datatype	Data
Float	X Position (cartesian)
Float	Y Position (cartesian)
Float	Z Position (cartesian)

28.4.5 TrackedScannerPositionXYZ

TrackedScannerPositionXYZ files contain the data of the stream with the same name.

Because all samples have the same size, the number of samples in the file can be calculated by (fileSize - headerSize) / sampleSize.

Table 28.7: TrackedScannerPositionXYZ - Header

Datatype	Data
Uint32_t	Length of the stream name (in this case ”17”)
N ∗ character	Coordinate system (in this case ”mmCartesianSample”)
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file

Continued on next page

Table 28.7 – concluded from previous page

Datatype	Data
Uint32_t	Frequency of the samples (in this case ”200000”)

Table 28.8: TrackedScannerPositionXYZ - Sample

Datatype	Data
Float	X Position (cartesian), backtransformed
Float	Y Position (cartesian), backtransformed
Float	Z Position (cartesian), backtransformed

28.4.6 SyncMessages

SyncMessages files contain the data of the stream with the same name.

Because all samples have the same size, the number of samples in the file can be calculated by (fileSize - headerSize) / sampleSize.

Table 28.9: SyncMessages - Header

Datatype	Data
Uint32_t	Length of the stream name (in this case ”9”)
N $*$ character	Coordinate system (in this case ”smPayload”)
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file
Uint32_t	Frequency of the samples (in this case ”0”)

Table 28.10: SyncMessages - Sample

Datatype	Data

Continued on next page

Table 28.10 – concluded from previous page

Datatype	Data
Uint64_t	Timestamp
Uint32_t	ID
Uint32_t	Encoder Postion Counter Timestamp
Uint32_t	Encoder Position Counter
Uint32_t	Encoder Velocity

28.4.7 SysMessages

CommandedLaserSwitchingTime files contain the data of the stream with the same name.

Because the size of the samples differs here, there is no way to calculate the number of samples in the file.

Table 28.11: SysMessages - Header

Datatype	Data
Uint32_t	Length of the stream name (in this case ”4”)
N ∗ character	Coordinate system (in this case ”text”)
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file
Uint32_t	Frequency of the samples (in this case ”0”)

Table 28.12: SysMessages - Sample

Datatype	Data
Uint64_t	Timestamp

Continued on next page

Table 28.12 – concluded from previous page

Datatype	Data
Uint32_t	Length of the sysmessage
N ∗ Char	Messagestring

28.4.8 WfiData

WfiData files contain the data of the stream with the same name.

Because all samples have the same size, the number of samples in the file can be calculated by (fileSize - headerSize) / sampleSize.

Table 28.13: WfiData - Header

Datatype	Data
Uint32_t	Length of the stream name (in this case ”10”)
N ∗ character	Coordinate system (in this case ”wfiPayload”)
Uint64_t	Starttimestamp of the first sample in this file
Uint64_t	Endtimestamp of the last sample in this file
Uint32_t	Frequency of the samples (in this case ”0”)

Table 28.14: WfiData - Sample

Datatype	Data
Uint64_t	Timestamp
Float	X Position (cartesian)
Float	Y Position (cartesian)
Float	Z Position (cartesian) = measured distance
Uint16_t	WedgePos

Continued on next page

Table 28.14 – concluded from previous page

Datatype	Data
Uint32_t	The RawValue of the measurement

28.5 Stream Data File Formats (CSV)

The CSV files are human readable. The separator if the semicolon (";"). The first line in each file is the header which describes the content of each column. The mmCartesianSample-files also consist of a line with additional information as comment (like Starttimestamp, Endtimestamp, Samples per second). This line is commented out by starting with a hashtag ("#").

29 Updating

Audience And Qualification

This chapter addresses *qualified personnel with administrator rights* who are assigned with updating a machine where an ASC system controller is incorporated, but not necessarily using the InScript software directly.

Requirements

> **NOTICE**
>
> Incompatibility
>
> causes loss of functionality.
>
> - Do not update a customized with a standard InScript software. Instead contact Novanta for support.
> - Do not mix files of one distribution with files of another distribution.

Procedure

1. Install InScript as described in chapter 6 on page 15.

   > **TIP**
   >
   > During installation you can decide whether to install the InScript software for all users or the current user only. If this decision is the same as it was for the already installed version then commissioning InScript software (step 3 of this procedure) is not necessary. In this case the settings of the older installation will be used for the update.
   >
   > How to determine how you decided last time:
   >
   > - In the menu of the already installed InScript software, click **File**.

> If the drop down menu contains the entry *Explore documents* then InScript software is installed for all users. If the drop down menu contains the entries *Explore documents* and *Explore application data* then InScript software is installed for the current user only.

2. Start InScript as described in chapter 7 on page 16.
3. Commission InScript as described in chapter 9 on page 28.

30 Uninstalling

Audience And Qualification

This chapter addresses *qualified personnel with administrator rights* who are assigned with uninstalling a machine where an ASC system controller is incorporated, but not necessarily using the InScript software directly.

Procedure

1. In the Windows *Start* menu, click the **gear icon** (Settings).

   The *Windows Settings* open.

2. Click **Apps**.
3. In *Apps and Features*, scroll down, find and click **InScript version** < **version**>.
4. Click **Uninstall**.

31 Appendix

31.1 List of InScript files

Table 31.1: InScript Files

Extension	Description
.arl	Arges RawLines files. Recommended file format for 2D and 3D vector data in InScript.
.dst	Distortions files.
.fdt	InScript font files.
.ini	Initialization files. These files contain several settings Examples: `InScript.ini` stores the settings of the InScript application; see section 31.6.1 on page 212. `InputOptions.ini` stores hotkeys used by InScript; see section 31.6.2 on page 219. `Globalsettings.ini` stores all InScript settings which shall be unique for all users; see section 31.6.3 on page 221.

Continued on next page

Table 31.1 – continued from previous page

Extension	Description
.job	InScript 2 job files.
.jobx	InScript job files.
.lay3	InScript layout files.
.log	Log files. Usually this file type is used to log several information or events. These files usually are very useful for the InScript support. Examples: request.log – Traces InScript communication events message.log – Records all messages that occur during InScript runtime application.log – Collects information about the InScript application behaviour at runtime. OpenGL.log – Records debug messages from the graphics driver.

Continued on next page

Table 31.1 – concluded from previous page

Extension	Description
.pen	InScript 2 pen files.
.penx	InScript pen files.
.ui	Qt User interface files (form files). The *UI File Container* view is able to load and show these files.

31.2 List of InScript folders

InScript files usually are stored in three different locations – the *program files path* , the *application data path* and the *user data path* . The program files path contains all files which are necessary to run InScript and don't need to be changed anytime, the application data path covers all data which is of interest for all users and in the user data path data is stored which is relevant only for the logged on user. Additionally there may be a fourth important path, an *user defined data path* .

Dependend of your settings InScript reads and writes its documents from/to the user data path, the application data path or the user defined data path; see section 13.4 on page 69. To make life easier for the user, InScript uses the term ”documents path” to identify this path.

31.2.1 Folders in the InScript program files path

Table 31.2: InScript Files

Directory	Description
\.	Binary files

Continued on next page

Table 31.2 – concluded from previous page

Directory	Description
Icons	Icon Files
Messages	Message files
Translations	Translation files
designer	Additional binary files
iconengines	Additional binary files
imageformats	Additional binary files
platforms	Additional binary files
printsupport	Additional binary files
sqldrivers	Additional binary files

31.2.2 Folders in the InScript application data path

Table 31.3: InScript Application Data Folders

Directory	Description
Config	Contains all global settings (valid for all users)
Documentation	Contains the InScript documentation

Additionally this folder is the InScript documents path, when the InScript option BasePathMode is set to SYSTEM

31.2.3 Folders in the InScript user data path

This folder is the InScript documents path, when the InScript option BasePathMode is set to USER

31.2.4 Folders in the InScript user defined data path

This folder is the InScript documents path, when the InScript option BasePathMode is set to CUSTOM

31.2.5 Folders in the InScript documents path

The InScript documents path contains the following subfolders:

Table 31.4: InScript Documents Folders

Directory	Description
Config	Settings files
Distortions	Distortion files
Firmware	Firmware files
Fonts	Font files
Jobs	Job files
Layouts	Layout files
Log	Log files
Pens	Pen files
Plugins	Plugin files (Binaries which extend InScript by additional functionality)
Plugins/Translations	Translation files for InScript Plugins
Rastergraphics	Rastergraphics files (bitmaps)
Vectorgraphics	Vectorgraphics files

See also section 13.4 on page 69.

31.3 How to create translation files for additional languages

InScript is prepared to support multiple languages for the GUI and system messages.

There are several translation source (∗.ts) files in the subfolders Messages and Translations of the program files path; see section 31.2 on page 207.

These translation source (∗.ts) files can be loaded in a special translator tool, the Qt Linguist. For more information about that tool and the translation process see Qt Linguist Manual[1].

[1] http://qt-project.org/doc/qt-5/qtlinguist-index.html

List of files which have to be translated:

- MessageFiles\arg_cal_messages_en.ts
- MessageFiles\arg_common_messages_en.ts
- MessageFiles\arg_controllerlib_messages_en.ts
- MessageFiles\Firmware_Messages_en.ts
- MessageFiles\RenderServer_Messages_en.ts
- MessageFiles\InScript_Messages_en.ts
- Translations\ARG_CAL_en.ts
- Translations\ARG_Common_en.ts
- Translations\AscWidgets_en.ts
- Translations\InScript_en.ts

TODO Distribute xxx_en.ts files to the Translations folder

TODO Distribute all available Qt translation files (qt_xx.qm)

31.4 How to extend InScript by plug-ins

See the InScript Plugins Documentation to learn how InScript can be extended by additional functionality with the help of plugins.

31.5 InScript troubleshooting

31.5.1 Hotkeys do not work as described

If hotkeys do not work as described open the File menu and select ”Explore documents”. Then open the ”log” subdirectory. If there is a file named ”application.log” delete this and restart InScript. Now there should be a new application.log file. Open this and search for an entry of the form ”Warning: Attention -- the InScript InputOptions.ini

file contains non-default values.” If there is one, delete the file ”InputOptions.ini” from the Config subpath of the InScript documents folder and restart InScript again. This should solve the problem.

31.5.2 Display problems in the Vector Editor

If you encounter strange display problems in the Vector Editor or you get the LowGraphicsCapabilities message on InScript startup see if there is a driver update available for your graphics card. If there is one, install it. May be you can get help on this by the system administrator of your company.

31.5.3 InScript does not store preferences

For example: InScript was started and the language set to en. Then the user logged off and on again. After that the language was de again.

If you run in problems like this, perform the following steps:

- Save your files (jobs, pens, etc.).
- Uninstall InScript
- Delete the InScript **program path**, the **user data path** and the **application data path**. See section 13.4 on page 69 for more information about this. On Windows 7 e.g. the mentioned directories can be found in
  - C:\Users\YourUsername\Documents\ARGES
  - C:\ProgramData\ARGES
  - C:\Program Files (x86)\ARGES
- Install InScript again, using administrator privileges.

31.5.4 Setup does not install sample files

See section 31.5.3 on page 211.

31.5.5 Other problems

Please contact our service. Help us to detect the problem as following:

- Describe the steps you execute to reproduce the problem
- Describe what behaviour you expect
- Describe what behaviour really occurs

If it isn't possible to reproduce the problem please describe it as close as possible.

In any case mail us all available InScript log files together with your support request:

- application.log
- message.log
- OpenGL.log (if available)
- request.log (if available)

You will find these files in the "log" subdirectory of the InScript documents folder. (Open the File menu and select "Explore documents" to get there.)

31.6 File format descriptions

31.6.1 InScript.ini

The file InScript.ini is used to store the settings of the InScript application. It will always be read from the Config subfolder of the InScript documents path. Most settings can be changed by the InScript preferences dialog, so changing this file directly will be needed very rarely.

The InScript.ini contains the following sections and entries:

31.6.1.1 [CMainWindow]

```
pos=@Point(200 200)

size=@Size(1048 564)
```

31.6.1.2 [CNavigatorViewWidget]

```
pos=@Point(0 0)
size=@Size(256 401)
```

31.6.1.3 [MessageView]

```
FocusLastMessage=true
ViewMode=0
```

31.6.1.4 [IP%20connection%20dialog]

```
address%20part%201=192
address%20part%202=168
address%20part%203=1
address%20part%204=216
port=1610
```

31.6.1.5 [Preferences]

```
Language=en
ConfirmInScriptShutdown=true
SaveLayout=true
ReloadLayout=true
SelectJobOnClearAndLoad=true
DataPathMode=0
```

31.6.1.6 [Plugins]

```
Path=<Comma separated list of paths>
```

Example:

```
Path=C:\Users\Max Mustermann\Documents\ARGES\InScript3\Plugins\
```

31.6.1.7 [Fonts]

```
Path=<Comma separated list of paths>
```

31.6.1.8 [Pens]

```
Path=<Comma separated list of paths>
```

31.6.1.9 [RequestLog]

```
DeleteFirst=true
MaxSizeInMB=0
Enabled=false
```

31.6.1.10 [ApplicationLog]

```
DeleteFirst=true
```

31.6.1.11 [MessageLog]

```
DeleteFirst=true
```

31.6.1.12 [MostRecentlyUsedFileNames]

```
size=10
0=<job file name>
1=<job file name>
2=<job file name>
3=<job file name>
4=<job file name>
```

```
5=<job file name>
6=<job file name>
7=<job file name>
8=<job file name>
9=<job file name>
```

31.6.1.13 [Help]

```
Bookmarks=<Comma separated list of bookmark entries>
```

31.6.1.14 [AutoSave]

```
ExecuteOnTimerActive=true
ExecuteOnTimerPeriod=2
DeleteOnSaveJob=false
DeleteOnStartActive=true
DeleteOnStartAgeLimit=2
UseJobPath=true
```

31.6.1.15 [JobCreation]

```
JobNodeInsertPosition=-1
```

31.6.1.16 [GUI]

```
ShowDockWidgetIcons=true
ControllerNameDisplayMode=2
WidgetCaptionDisplayMode=0
CustomWidgetCaption=CONTENT - NAME - [ID]
ShowAscCpuLoad=true
```

```
UsedStyle=0
UsedStyleSheet=
```

31.6.1.17 [ScriptEditor]

```
Fontsize=10
Font=@Variant(\0\0\0\\0\0\0\x16\0\x43\0o\0u\0r\0i\0\x65\0r\0 \0N\0\x65\0w\”
\0\0\0\0\0\0\xff\xff\xff\xff\x2\x1\0\x32\x10)
```

31.6.1.18 [VectorEditor]

```
Animations\LineOrder\Mode=1
Animations\LineOrder\Pause=1500
Animations\LineOrder\TimePerVertex=10
Common\PaintMode=1
Common\LimitTo2D=false
Common\ShowRulers=true
Common\Stereo\Mode=0
Common\Stereo\EyeSeparation=0.3
Grid\Show=true
Grid\Color=#b0c5deff
Import\Flatness=0.01
Objects\Selected\Color=#cc0000ff
Objects\Highlighted\Show=true
Objects\Highlighted\Color=#ff8a08ff
Objects\Inactive\Color=#808080ff
Objects\LineStartPoints\Show=false
Objects\LineStartPoints\Color=#17eb17ff
Objects\LineStartPoints\Size=5
```

```
Objects\LineEndPointArrows\Show=false
Objects\LineEndPointArrows\Color=#ebcc1bff
Objects\LineEndPointArrows\Length=8
Objects\LineEndPointArrows\Width=3
Overlays\BoundingBox\Show=true
Overlays\BoundingBox\Color=#008000ff
Page\Mode=1
Page\SizeLeft=-100
Page\SizeTop=100
Page\SizeRight=100
Page\SizeBottom=-100
Page\PageColor=#ffffffff
Page\BackColor=#f0f0f0ff
Page\OverlapColor=#e0ffff3f
Page\CoordShow=true
Page\CoordColor=#8aabdcff
Page\MoveRectShow=true
Page\MoveRectColor=#006680ff
Page\MarkRectShow=true
Page\MarkRectColor=#0080bfff
Page\ClipRectShow=true
Page\ClipRectColor=#0099ffff
Preview\TailLines\Color=#e60000ff
Preview\HeadLines\Color=#0000e6ff
Preview\JumpLines\Color=#e6e600ff
Preview\MainLines\Color=#000000ff
StepWidth\Rotate=15
```

```
StepWidth\Scale=0.5
StepWidth\Slant=0.5
```

31.6.1.19 [StandardPenColors]

```
0=#ffffffff
1=#1215cdff
...
32=#000000ff
```

31.6.1.20 [StandardPenSizes]

```
0=0.1
1=0.1
...
32=0.1
```

31.6.1.21 [StandardPenUnits]

```
0=0
1=0
...
32=0
```

31.6.1.22 [StandardPenShowFlags]

```
0=1
1=1
...
32=1
```

31.6.1.23 [systemControllerList]

```
size=3
0=ASC#192.168.1.214:1610
1=ASC#192.168.1.215:1610
2=ASC#192.168.1.216:1610
```

31.6.1.24 [ASC%23192.168.1.214%3A1610]

```
connected=false
```

31.6.1.25 [ASC%23192.168.1.215%3A1610]

```
connected=false
```

31.6.1.26 [ASC%23192.168.1.216%3A1610]

```
connected=true
```

31.6.2 InputOptions.ini

The file InputOptions.ini is used to store some hotkeys respectively key combinations used by the InScript application. It will always be read from the Config subfolder of the InScript documents path.

The InputOptions.ini contains the following sections and entries:

31.6.2.1 [InScript]

```
TriggerContextMenu\modifiers=0
TriggerContextMenu\button=2
```

31.6.2.2 [Navigator]

```
SkipFollowingNewNodes\modifiers=134217728
```

31.6.2.3 [VectorEditor]

```
Perform2DOperationCentered\modifiers=33554432
Perform2DOperationStepwise\modifiers=67108864
Perform2DOperationOnSingleAxis\modifiers=67108864
TriggerSelect\modifiers=0
TriggerSelect\buttons=1
TriggerExtendSelection\modifiers=33554432
TriggerExtendSelection\buttons=1
TriggerToggleSelection\modifiers=67108864
TriggerToggleSelection\buttons=1
TriggerSelectNextAtSamePos\modifiers=134217728
TriggerSelectNextAtSamePos\buttons=1
TriggerDeselectAll\modifiers=0
TriggerDeselectAll\buttons=1
TriggerSelectionFrame\modifiers=0
TriggerSelectionFrame\buttons=1
TriggerExtendSelectionFrame\modifiers=33554432
TriggerExtendSelectionFrame\buttons=1
TriggerToggleSelectionFrame\modifiers=67108864
TriggerToggleSelectionFrame\buttons=1
TriggerEditMode\modifiers=201326592
TriggerEditMode\buttons=1
TriggerPan\modifiers=0
TriggerPan\buttons=4
```

TriggerPanAlternative\modifiers=0

TriggerPanAlternative\buttons=3

TriggerRotate\modifiers=67108864

TriggerRotate\buttons=4

31.6.3 GlobalSettings.ini

The file GlobalSettings.ini is used define some InScript settings, which shall be unique for all users. It will always be read from the Config subfolder of the InScript application data path.

On some platforms (Linux and Mac) the application data path may not exist. If you want to change some settings in this file first you may have to create this path with appropriate file access rights or change the file access rights of the existing path. If the file GlobalSettings.ini does not exist in this path, you have to create it itself just as well.

You can use the application.log file in the Log subfolder of the InScript documents path to determine where InScript expects the GlobalSettings.ini to be on the platform it is running on. This file contains an entry like the following examples (taken from a Window 7 and a Linux Mint installation):

```
Info: Loading global options from

    C:\ProgramData\ARGES\InScript3\Config\GlobalSettings.ini

Info: Loading global options from

    /usr/share/ARGES/InScript3/Config/GlobalSettings.ini
```

The GlobalSettings.ini contains the following sections and entries:

31.6.3.1 [Preferences]

BasePathMode=[0..2]

0 – user oriented data storage

1 – system oriented data storage

2 – user defined data storage

Standard = 0

For more information see section 13.4 on page 69.

CustomBasePath=C:\\Users\\Username\\Documents\\ARGES\\InScript3

Path which is used for the user defined data storage.

On the Windows platform this path name may be an UNC name too. I.e. to access a Windows share this entry might look like the following examples:

CustomBasePath=\\\\SERVERNAME\\ShareName\\InScript3.NetData

CustomBasePath=//SERVERNAME/ShareName/InScript3.NetData

The path name in the example above (InScript3.NetData) is optional.

On Linux systems you have to mount the network share to access it. This can be done by editing the /etc/fstab or (for testing purposes) directly in the Linux shell:

```
sudo mkdir /mount/ShareName

sudo mount -t cifs //SERVERNAME/ShareName /mount/ShareName -o

    user=username,password=mypassword,file_mode=0777,dir_mode=0777
```

Then you can set the CustomBasePath property to the mounted share name or a directory beyond this:

```
CustomBasePath=/mount/ShareName/InScript3.NetData
```

On system running on Mac OS X you have to connect the network share first too. Open a Finder window and select ”Go to”, ”Connect with server” from the menu. Then select the Server you want to connect to and enter the user name and the password. Then you can access the network share as a subdirectory of the /Volumes folder. Once again, you may access a directory beyond the share name too, e.g:

```
CustomBasePath=/Volumes/ShareName/InScript3.NetData
```

For more information see section 13.4 on page 69 and the documentation of your operating system.

31.6.4 Stylesheet files

The InScript application is written with the Qt application framework. Applications written with this framework automatically supports the styling of almost every GUI element by stylesheets. It is possible to load a stylesheet at InScript program start with the -stylesheet option on the command line (see section 7.6 on page 18).

You can find more information about Qt style sheets like the syntax, a complete reference and examples in the Qt Style Sheets [2] documentation.

31.7 Glossary

31.7.1 Context

Each operation in the InScript menu or on one of the toolbars applies to a certain node, controller or view. This relationship we called a context. According to the type of object an action will be applied to we distinguish three different types of contexts – a node context, a controller context and a view context. The node and the controller context will be set by the selection of the *Navigator* view, the view context is defined by the current view. Additionally the job context terms the job the current node context belongs to.

31.7.2 Controller

A controller is a hardware entity specifically designed for laser beam steering and subsystem controlling. Today usually it is an autonomous unit as the ARGES ASC. In InScript controllers are represented by controller nodes in the *Navigator* view.

[2] http://doc.qt.io/qt-5/stylesheet.html

31.7.3 Device

Devices are the hardware connected to the controller or present on the controller. The controller uses device specific drivers to control them. Devices and their drivers can be accessed via the Devices subtree of each controller in the *Navigator* view.

31.7.4 Fixed Views

Fixed views are views which can't be undocked or moved inside the application window. They are called ***fixed views*** because of their fixed position. Most of them can't be closed, i.e. are visible always. The *Job Control*, *Navigator* and *Messages* view are fixed views.

31.7.5 Jobs

A Job is a customized hierarchical tree structure of Job nodes that is used to perform respectively execute a certain task. Jobs can be created and modified in the *Navigator* view. The visual output of a job can be viewed and/or modified in the InScript *Vector Editor* view.

When a job will be executed the Job nodes tree will be processed from his root node to the last node in the last branch. Depending on each job node type the controller hardware renders the data for real-time execution on the connected hardware (e.g. scan heads), gives interfaces to external IO etc..

31.7.6 Jobnode

Jobnodes are special nodes which serve as execution units

31.7.7 Node

Nodes are elements for a certain execution, steering or processing function. They can be treated as groups of variables for a specific purpose.

31.7.8 Pen

Pens are nodes used to control parameters of devices respectively device drivers. In InScript e.g. the laser power, frequency and precessing speed can be defined by a pen.

31.7.9 Pensection

A pen section contains the settings of the corresponding active device driver. It contains device driver parameters that can be changed during a job only.

31.7.10 Penvariable

A pen variable represents a device variable, which value will be overridden by the pen.

31.7.11 Unique HardwareId

A Unique HardwareId is a unique identifier for a controller It always consists of a combination of three upper case letters that designate the controller type (e.g. ”ASC” or ”NCC”) followed by a hash character (”#”) and a specification of the controller. For ASC controllers the specification is ”ASC#IP-Address:port”, with port = 1610.

So a valid Unique HardwareID for an ASC controller might be e.g.

”ASC#192.168.1.216:1610”.

31.7.12 User Views

All kind of views which can be created by the user on demand are called ***user views***. User views can be docked together in the main window of the application, undocked (made floating) or stacked. The *Inspector*, *Node Properties*, *UI File Container* and *Vector Editor* view are user views.

31.7.13 Variable

Variables are the fundamental configuration components of all ARGES controllers. They are arranged hierarchically in a tree structure that is stored on the controller. All components like devices, jobnodes or jobs consist basically of a collection of these variables. Usually a user will only deal with *nodes* that integrate these variables.

Bibliography

[1] *ARGES System Controller – Operator Manual*. (ASC_controller_manual_en.pdf).

[2] *ControllerLib – Interface Description*. (ControllerLib_en.pdf).

[3] *Firmware 3 – User Manual*. (Firmware_<version>_manual_en.pdf).

[4] *InScript 3 – Plug-ins Description*. (InScript_<version>_plug-ins_en.pdf).

Novanta Corporation

125 Middlesex Turnpike
Bedford, MA 01730, USA

Phone: +1-781-266-5800
Email: Photonics@Novanta.com
Website: www.NovantaPhotonics.com

InScript Software
User Manual
Version 5.0.1.65