VisualDCT User's Manual

Project:	VisualDCT
Classification:	User's Manual
Identification:	CSL-MAN-02-xxxxxx

Document History

Revision	Date	Author	Section	Modification
1.0	2002-10-19	Matej Sekoranja	all	Created.
1.1	2002-10-22	Matej Sekoranja	**Released**	

Scope

This document is a users manual of how to use VisualDCT and also contains some tips and ticks.

Confidentiality

This document is classified as a **public document**. Redistribution and use, with or without modification, are permitted provided that:

1. the copyright notice is retained
2. a reference to the original document is made in case of modifications
3. this document may not be distributed for profit except as explicitly permitted by valid licenses.

Audience

The audience of this document are all users of VisualDCT.

References

ID	Author	Reference	Revision	Date	Publisher
1	Matej Sekoranja	VisualDCT Project		2002	Cosylab, Ltd.
2	Matej Sekoranja	VisualDCT latest build	2.3.1237	2002	Cosylab, Ltd.
3	Matej Sekoranja	Java Installation and Build Process of VisualDCT		2002	Cosylab, Ltd.
4	Matej Sekoranja	VisualDCT EPICS Databases Hierarchy Support		2002	Cosylab, Ltd.
5	Sunil Sah	VisualDCT Plugins		2002	Cosylab, Ltd.

Table of Contents

How to Read this Document

This document's meta-information (authors, revision history, table of contents, ...) can be found above. What follows below is the body of the document. The body is composed of several sections, which may be further composed of subsections.

Typographical styles are used to denote entities of different kinds. For a full list of entities and their respective typographic conventions, please refer to the Styles section of the XML Documentation document.

When viewing the document in a non-printed form, it is possible to submit comments regarding a given section to the document's owner. This is achieved by clicking the mail icon next to the section title. For this to work, your mail must be configured to properly handle the `mailto` URLs.

1. Introduction

Visual Database Configuration Tool (VisualDCT) is an EPICS database configuration tool completely written in Java and therefore supported in various systems. It was developed to provide features missing in existing configuration tools as Capfast and Graphical Database Configuration Tool (GDCT). Visually VisualDCT resembles GDCT; records can be created, moved and linked, fields and links can be easily modified. But VisualDCT offers more: using hierarchical design od DBs and groups, records can be grouped together in a logical block. Additionally indication of data flow direction using arrows makes the design easier to understand. VisualDCT has a powerful DB parser, which allows importing existing DB and DBD files. Output file is also DB file, all comments and record order is preserved and visual data saved as comment, which allows DBs to be edited in other tools or manually.
This manual describes the VisualDCT version 2.3 build 1237.

2. Basic principles

VisualDCT is designed to create and maintain EPICS record instance database (.db) files. In order for VisualDCT to execute properly, a database definition (.dbd) file has to be provided which contains the specifications for the various record and device types that they intend to reference in any record instance database (.db) file to be created by VisualDCT. Once a database definition (.dbd) file has been specified, records can be created, copied, renamed, etc. using the various facilities provided by the VisualDCT. As the user interacts with the various VisualDCT windows, selections, and data entry fields, the results of these interactions are displayed on the screen. Revisions and data entry updates of record instance data displayed on the screen do not replace previously stored record instance data until the user saves currently modified record instance database (.db) file. As VisualDCT executes, it attempts to trap and display the most common situations that might lead to diminishing the integrity of the user supplied information.

3. Running VisualDCT

In order to run VisualDCT, Java Runtime Environment 1.4 is required. VisualDCT is distributed as a Java ARchive package (.jar file), so there is only one file in the binary distribution. This file has to be added to the java **`classpath`** variable (search path for application classes and resources) to help JVM find **`com.cosylab.vdct.VisualDCT`** class, which is the main class of the VisualDCT.
Usage of VisualDCT:

```
java -cp VisualDCT.jar com.cosylab.vdct.VisualDCT [<DBD>* or <DB>*]
```

Listing 1: Basic run command.

VisualDCT Java ARchive package (.jar file) is so called executable JAR file, which means it can be run as:

```
java -jar VisualDCT.jar [<DBD>* or <DB>*]
```

Listing 2: Running executable JAR.

or if you GUI has this feature double-click on VisualDCT.jar will also do it. VisualDCT accepts two parameters which are not obligatory: database definition files and record instance database files (if this is already specified in DBs, specification of database definition file is not needed). DBD is recognised as a file with .dbd extension otherwise DB is assumed. If there is no DBDs specified an Open dialog will appear allowing you to specify DBD file. If even then there is no valid DBD specified VisualDCT will exit with the following output:

```
o) No DBD loaded! Exiting...
```

Listing 3: No DBD loaded error message.

An example of running VisualDCT, using test.dbd definition database and test.db instance database file:

```
java -jar VisualDCT.jar -DVDCT_DIR=~/epics test.dbd test.db
```

Listing 4: An example of running VisualDCT.

`VDCT_DIR` environment variable is used to define the default working directory.
For saving configuration data Java Preferences API is used. This means configuration is kept in a system depended configuration storage, e.g. registry when using Windows OS.

4. Hierarchy support

VisualDCT also supports hierarchical design of EPICS databases. For detailed information about it refer to VisualDCT EPICS Databases Hierarchy Support document.

5. Features

5.1. Rapid Database Development (RDD)

VisualDCT can be considered as a rapid database development tool - unintuitive database construction using ordinary text editors can be done quickly with a few simple mouse-clicks minimizing all unnecessary keyboard input. Visualization of the record instance database makes databases easier to understand, errors are much easier to find (e.g. broken links are indicated by a red cross) and helps find a better design of the databases. Allowing user to user hierarchal design and split databases into logical blocks.

5.2. Database file parser, input/output file

VisualDCT creates and maintains only one file, the record instance database (.db or .vdb) file, and does not have any additional graphical information file avoiding any possible consistency problems when having multiple files, all necessary visual composition data is stored as comments at the end of the DB file. An example of DB file:

```
#! Generated by VisualDCT v2.3
#! DBDSTART
#!  DBD("/home/matej/epics/test1.dbd")
#! DBDEND

path  ":/home/matej"
addpath  "epics:epics/templates"
include  "dummy.db"

# This is an record comment...
record(calc, error) {
   field(INPA, "$(slmot1.position)")
}

record(ao, speed) {
   field(DTYP, "Soft Channel")
   field(OUT, "$(slmot1.speed)")
   field(HIHI, "1208")
}

# This is an expand comment...
expand("slideMotor.vdb", slmot1) {
   macro(name, "sm1")
   macro(address, "4")
   macro(demand, "slide1:demand.VAL")
}

#! Further lines contain data used by VisualDCT

#!  TemplateInstance("slmot1",100,340,0,"")

#!  Record(error,640,20,0,0,"error")
#!  Field("error.INPA",16777215,0,"error.INPA")
#!  Link("error.INPA","error/INPA")
#!  Connector("error/INPA","$(slmot1.position)",417,117,16777215,"",0)
#!  Record(speed,720,500,0,0,"speed")
#!  Field("speed.OUT",16777215,0,"speed.OUT")
#!  Link("speed.OUT","speed/OUT")
#!  Connector("speed/OUT","$(slmot1.speed)",617,577,16777215,"",0)
```

Listing 5: An example of complete DB file.

VisualDCT has powerful parser which has ability to parse already existing DBs, files which have been created or modified with other tool. It also detects syntax errors in databases, including DBDs. Defective visual composition data or its absence are safely handled and do not raise any critical error, VisualDCT simply automatically layouts all objects without any visual data. What is more, VisualDCT preserves comments and record/field order in the record instance database, which offers the ability to edit your databases in other tools or manually without making any harm to the databases and VisualDCT.

5.3. Visual representation of objects

5.3.1. Record

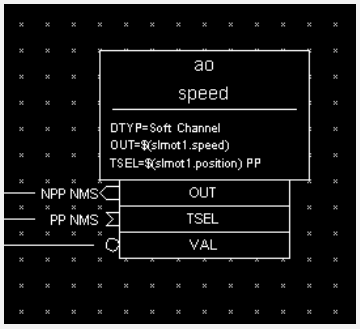


Figure 1: Record

Record is represented as a write square with its type and name written inside. Below the line inside the record there is an area where all fields values are shown, selection of fields depends on its visibility property.
There are three types of fields that can appear as part of the record (white squares below the record): **VARIABLE** (data), **INPUT**, **OUTPUT** and **FORWARD** fields. Variable fields hold a piece of data, such as the **VAL** or **HIHI** fields. Since the variable fields can be populated by other record's output fields and read by other record's input fields, a field node will appear below the record. Additionally indication of data flow direction using arrows makes fields easy to distinguish:

- **circle** - **VARIABLE** fields
- **out-arrow** - **OUTPUT** and **FORWARD** fields
- **in-arrow** - **INPUT** fields

A multi-point wire can be drawn between any two linkable fields simply by adding connectors (moveable small squares on a link line). If a link is an inter-group link (link between two fields which are not in the same group), the link is represented as a line going in the screen with the target link name shown by side.

5.3.2. Group

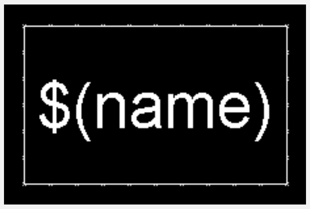
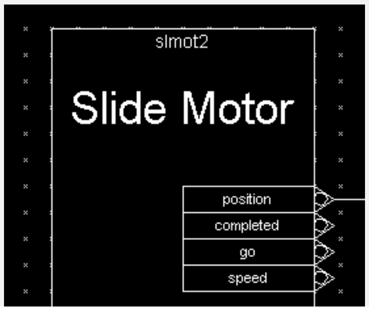


Figure 2: Group

Group is represented as a white square with its name inside. Double click over it descends into it.

5.3.3. Template Instance



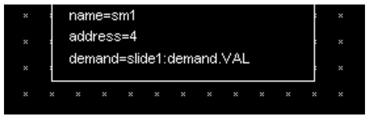


Figure 3: Template Instance

Template instance is represented as a larger write square. It its body it contains: name (id) at right at the top, template description below, template ports (values to be passed out of the template) and template instance properties (macro definitions to be passed inside the template).
To change template instance properties double click over it and use Inspector tool or Shift + double click to descent into it.

5.3.4. Links

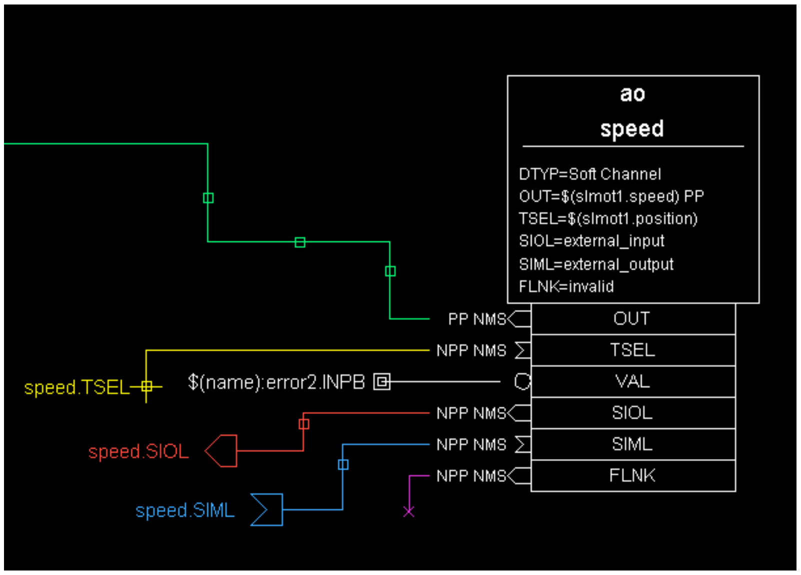
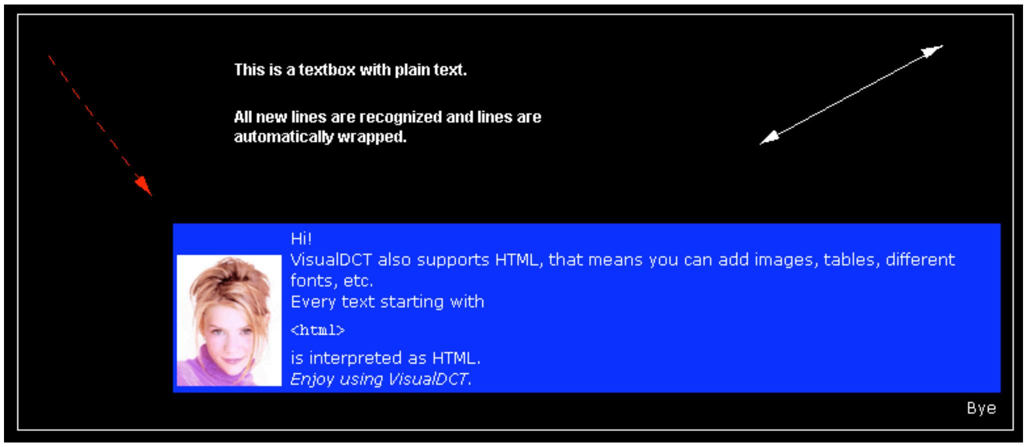


Figure 4: Links

VisualDCT distinguish several link types:

- **oridnary** - normal link. Like for any other link type, link wire can be freely broken using connectors.
- **invisible** - link with incomplete wire (to make complex databases more cleaner). To create it add connector, left button click over it and choose Mode - Invisible .
- **inter-group** - link between objects which are not in the same group.
- **external input** - link which target is an external object (invalid link for VisualDCT). To create it enter target, add connector to field, left button click over connectgor and choose Mode - External INPUT .
- **external output** - link which target is an external object (invalid link for VisualDCT). To create it enter target, add connector, left button click over connector and choose Mode - External OUPUT .
- **invalid** - link for which VisualDCT did not found its target.

5.3.5. Other Graphical Objects


This is a textbox with plain text.

All new lines are recognized and lines are automatically wrapped.

Hi!
VisualDCT also supports HTML, that means you can add images, tables, different fonts, etc.
Every text starting with
`<html>`
is interpreted as HTML.
Enjoy using VisualDCT.

Bye

Figure 5: Line/arrow, box and textbox (plain and HTML)

5.4. Linking

There are two ways of linking:

- value of the **INPUT**, **OUTPUT** or **FORWARD** link field is entered using Inspector tool
- using linking capability of VisualDCT using only mouse:
    1. Right click on the parent record of the **INPUT**, **OUTPUT** or **FORWARD** link field
    2. Pop-up menu will appear, choose the appropriate link field
    3. The parent record will blink until the **VARIABLE** field or record if **FORWARD** link is determined; to do this, there are tree options:
        - left click on the record - **VAR** field is used, or record if **FORWARD** link
        - left click on the field - link to clicked filed is created
        - right click on the record - pop-up menu will appear allowing you to select the **VARIABLE** field

5.5. Grouping

Grouping is based on the naming, for instance record with name `grp1:ao001` belongs to group `grp1` and record `grp1:grp2:ao002` belongs to group `grp2` which belongs to `grp1`, so groups can be also nested. In previous examples **:** character was used as a grouping separator, which is the default, but it can be easily changed in Settings window (View - Settings...).
Double click on the group descends into the group and shows only the records and groups in this group, use View - Level Up (Shift + Up) to ascend from the group.
Grouping can be easily achieved on the naming basis, simply by renaming records, or using in the Group or Ungroup commands from the Edit menu on the object selection.

6. User Interface

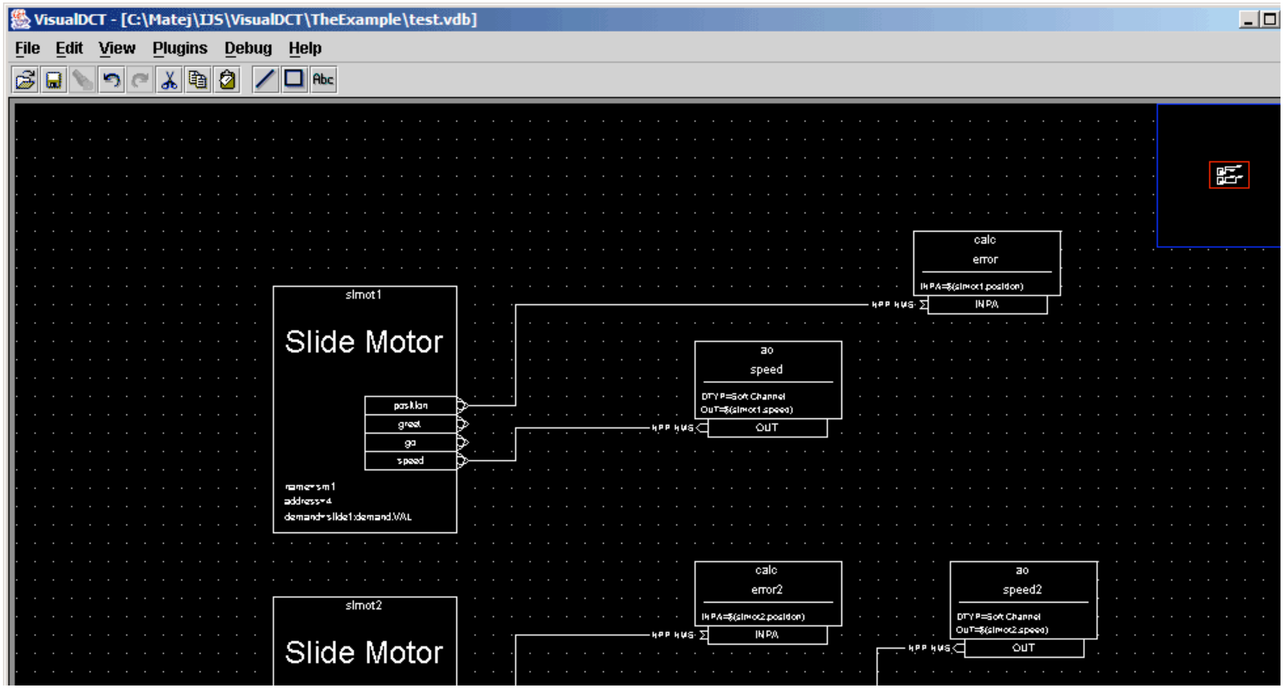
As every powerful IDE also VisualDCT provides indispensable facilities as clipboard and undo support. A great effort was given to synchronization between the record instance database and its visualization. Every change done visually is immediately reflected in the database and vice versa; all actions like moving, renaming and deletion of records which affect links are automatically fixed by the VisualDCT.

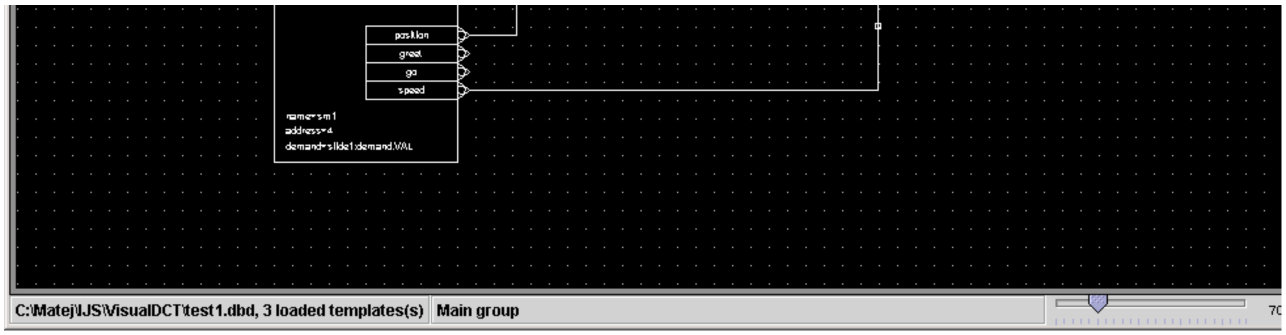
6.1. Graphical User Interface

Graphical User Interface of the VisualDCT consists of 3 main windows:

1. Main window
2. Inspector window
3. Console window

6.1.1. Main window





position
great
go
speed
name=sm1
address=4
demand=slide1:demand.VAL

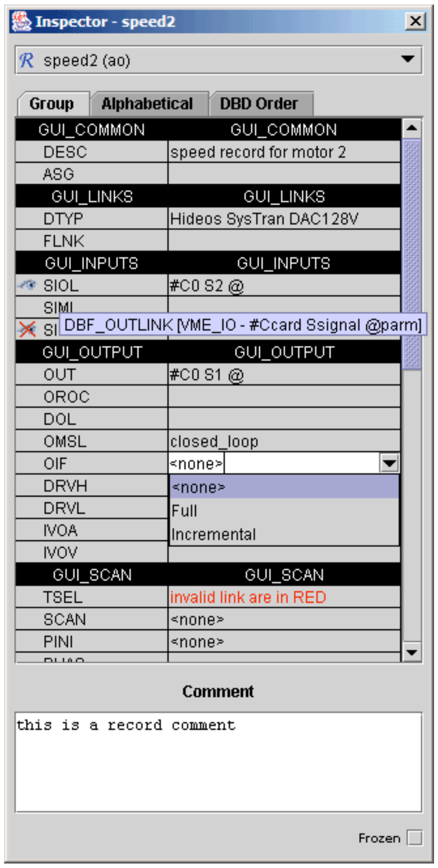
C:\Matej\IJS\VisualDCT\test1.dbd, 3 loaded templates(s) | Main group

Figure 6: Main window

The main window consists of:

1. **Main menu**
2. **Toolbar** - makes access to the frequently used actions easier.
3. **Workspace with Navigator** - it is the main component of the VisualDCT, it provides visualization and the capability of editing the record instance database. Navigator is a miniature view of the whole workspace. Using mouse over the navigator you can easily move through the workspace.
4. **Status bar with Zoom scale slider** - shows the name of the active definition database and the name of the current group. Zoom scale slider is used to easily change zoom scale.

6.1.2. Inspector window



Inspector - speed2

speed2 (ao)

Group | Alphabetical | DBD Order

GUI_COMMON	GUI_COMMON
DESC	speed record for motor 2
ASG	
GUI_LINKS	GUI_LINKS
DTYP	Hideos SysTran DAC128V
FLNK	
GUI_INPUTS	GUI_INPUTS
SIOL	#C0 S2 @
SIML	
SI	DBF_OUTLINK [VME_IO - #Ccard Ssignal @parm]
GUI_OUTPUT	GUI_OUTPUT
OUT	#C0 S1 @
OROC	
DOL	
OMSL	closed_loop
OIF	<none>
DRVH	<none>
DRVL	Full
IVOA	Incremental
IVOV	
GUI_SCAN	GUI_SCAN
TSEL	invalid link are in RED
SCAN	<none>
PINI	<none>

Comment

```
this is a record comment
```

Frozen

Figure 7: Inspector window

The inspector tool provides a capability of inspecting (examining) and modifying of all objects properties. Basically the inspector tool is already all needed to edit record instance databases - it replaces ordinary text editor.
The inspector window consists of:

1. **Object combobox** - shows currently inspected object and allows user to choose another object in the current database.
2. **Property table** - name-value pair table allowing user to inspect or modify fields. Record fields are grouped according `promptgroup` field defined in definition database.

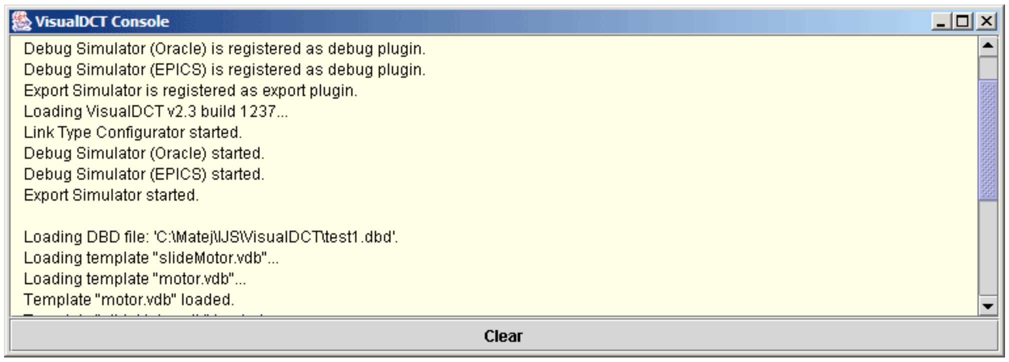
3. **Comment textarea** - shows record comment and allows user to modify it.
4. **Status bar** - provides basic help, the value of the `prompt` field defined in definition database is shown for fields and so helping uses to understand the meaning of the fields (e.g. LBRK - "Last break point").

Each field has additional property called `visibility`, whether the field value is shown inside the record body (see Record representation). It can be changed by clicking right mouse button over left column. Tree icons indicate the visibility state of the field:

- - field value is shown if value differs from default.
- - field value is always shown.
- - field value is never shown.

A macro definition can be entered for any field, including menus and links. Any changes to fields take place immediately in the visual composition.

6.1.3. Console window



```
Debug Simulator (Oracle) is registered as debug plugin.
Debug Simulator (EPICS) is registered as debug plugin.
Export Simulator is registered as export plugin.
Loading VisualDCT v2.3 build 1237...
Link Type Configurator started.
Debug Simulator (Oracle) started.
Debug Simulator (EPICS) started.
Export Simulator started.

Loading DBD file: 'C:\Matej\IJS\VisualDCT\test1.dbd'.
Loading template "slideMotor.vdb"...
Loading template "motor.vdb"...
Template "motor.vdb" loaded.
```

Figure 8: Console window

Console window is used to replace standard output of the JVM, which is often ignored by the user. All output is redirected to the console which pops up every time a new message appears in it and so informs user about the new message.

6.2. Command reference

This section describes all commands available by the VisualDCT.

6.2.1. Menu command reference

This section describes menu commands available by the VisualDCT.

6.2.1.1. File Menu

- **New** - close the currently active database, and allow the user to create a new database.
- **Open** - close the currently active database, and provide a file selection window which will allow the user to open a new existing database. The record instance database will be checked for consistency with loaded DBDs.
- **Import DB** - provide a file selection window which will allow the user to specify a new existing database which will be added (appended) to the existing active database (only loaded into template registry if contains `templates`).
- **Manage DBDs** - pop-up a DBD Manager dialog allowing to remove/load other DBDs.
- **Save** - save the currently active database.
- **Save As** - save the currently active database, and allow the user to specify a name of the file into which the database will be saved.
- **Save As Group** - save the currently active group of database as an standalone database, and allow the user to specify a name of the file into which the database will be saved.
- **Generate** - save the currently active database as a flat database.
- **Generate As Group** - save the currently active group of database as a flat standalone database.
- **Export** - menu containing all export plugins.
- **Export as PostScript** - exports current view to the PostScript (.ps) file.
- **Print** - print the current visible area of the database.
- **Print as PostScript** - print the current visible area of database as PostScript.
- **Print Preview** - display a view of the active database as it will be printed.
- **Page Setup** - change the printer page options.
- **Exit** - exit the VisualDCT.

6.2.1.2. Edit Menu

- **Undo** - undo the last action.
- **Redo** - redo the previously undone action.
- **Cut** - cut the selection and put it on the clipboard.
- **Copy** - copy the selection and put it on the clipboard.
- **Paste** - insert the clipboard contents to the workspace.
- **Move/Rename** - move/rename the selection.

- **Group** - group the selection.
- **Ungroup** - ungroup the selection of groups.
- **Delete** - delete the selection.
- **Select All** - select all objects in the current group.

6.2.1.3. View Menu

- **Flat View** - not implemented.
- **Level up** - move to the parent group.
- **Zoom In** - increase zoom scale by 10%.
- **Zoom Out** - decrease zoom scale by 10%.
- **Zoom Selection** - zoom the selection to fit the screen.
- **Base view** - move to the centere of the workspace and set zoom scale to 100%.
- **Toolbar** - toggle toolbar visibility.
- **Statusbar** - toggle statusbar visibility.
- **Navigator** - toggle navigator visibility.
- **Show Grid** - toggle grid visibility on the workspace.
- **Snap to Grid** - snap objects to the grid.
- **Settings** - pop-up the settings dialog.

6.2.1.4. Plugin Menu

- **Plugin Manager** - pop-up the plugin manager dialog.
- `list of installed plugins...`

6.2.1.5. Debug Menu

- **Start** - start debug plugin.
  - list of installed debug plugins...
- **Stop** - stop the running debug plugin.

6.2.1.6. Help Menu

- **Help Topics** - list help topics (temporarily only mouse commands are shown).
- **Books Online** - not implemented
- **About Box** - display program information, version number and copyright.

6.2.2. Mouse command reference

Button	Trigger	Actor	Action
left	click	record, group, template	Select object
	double-click	group	Descend into group
		field, record, template	Inspect object
		connector	Find target
		blank workspace	Create new record
	Shift + double-click	template	Descend into template
	click, drag	record, group, template / selection	Move object / selection
		navigator	Move through the workspace
	Shift + drag	blank workspace	
	drag	blank workspace	Select record, groups, templates
right	click	object, blank workspace	Popup object specific menu
		left navigator column of a field	Change field visibility
	Shift + click	field with more than one link	Rotate link
	drag	blank workspace	Zoom in selection

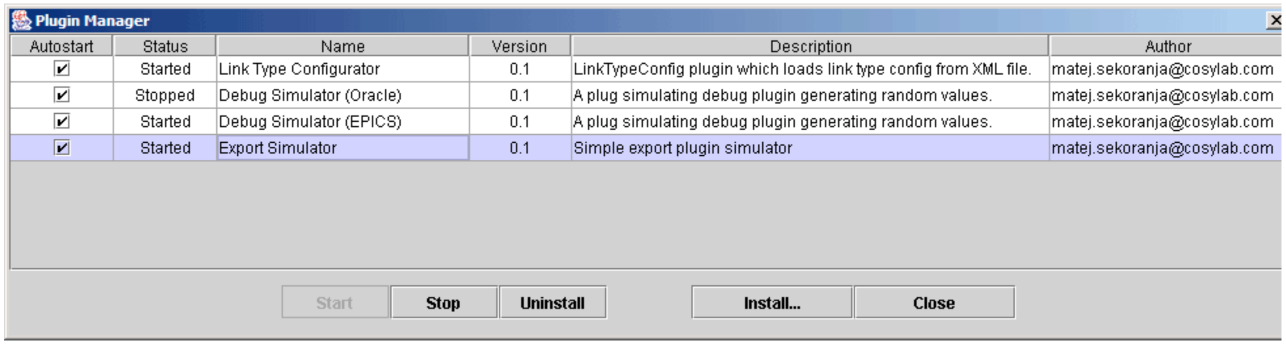
Table 1: Mouse command reference

6.2.3. Keyboard command reference

Among all visually documented (on the left side of menu items) combinations there is one additional combination:

Ctrl + arrow key to navigate through workspace.

7. Plugins



Autostart	Status	Name	Version	Description	Author
✓	Started	Link Type Configurator	0.1	LinkTypeConfig plugin which loads link type config from XML file.	matej.sekoranja@cosylab.com
✓	Stopped	Debug Simulator (Oracle)	0.1	A plug simulating debug plugin generating random values.	matej.sekoranja@cosylab.com
✓	Started	Debug Simulator (EPICS)	0.1	A plug simulating debug plugin generating random values.	matej.sekoranja@cosylab.com
✓	Started	Export Simulator	0.1	Simple export plugin simulator	matej.sekoranja@cosylab.com

Figure 9: Plugin Manager window

To make VisualDCT more flexsible support for plugins was implemented. For detailed information about plugins refer to VisualDCT Plugins document.

8. Future Plans

Since VisualDCT is an active project, there are some features to be implemented in the future releases of VisualDCT and all bug reports, suggestions and ideas are very appriciated.

9. Conclusion

If this manual did not meet all of your expectations or if you have any questions or suggestions, please feel free to contact the author.
Enjoy using VisualDCT.