VISUAL DCT – EPICS DATABASES CAN BE FUN

Rok Sabjan*, Sunil Sah and Matej Sekoranja, J. Stefan Instititute and Cosylab Ltd., Slovenia,

John Maclean, APS, Argonne National Laboratory, USA

Abstract

Visual DCT [1] is an EPICS [2] 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 [3] and GDCT [4]. Visually Visual DCT resembles GDCT - records can be created, moved and linked, fields and links can be easily modified. But Visual DCT offers more: using groups, records can be grouped together in a logical block, which allows a hierarchical design. Additional indication of data flow direction using arrows makes the design easier to understand. Visual DCT 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. Great effort has been taken and many tricks used to optimize the performance in order to compensate for the fact that Java is an interpreted language. Visual DCT is fast becoming the most popular tool among EPICS community, constantly being updated for performance tunning.

1 BASIC PRINCIPLES

VisualDCT was designed [5] to create and maintain EPICS record instance database (.db) files. In order for VisualDCT to execute properly, a database definition (.dbd) file (or files) 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 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.

VisualDCT's main objective is to offer powerful and intuitive development environment to the EPICS database engineer. With having numerous features and each of them "just one click away" time and effort can be focused on the database design instead on using the tools. A lot of focus has been put on visualization and performance.

In order to run VisualDCT, Java Runtime Environment 2 (version 1.4) is needed [6]. VisualDCT is distributed [7] as a Java ARchive package (.jar file), so there is only one file in the binary distribution.

2 NEW UPDATES

VisualDCT is fast becoming the most popular database configuration tool among EPICS community. The ever increasing number of users has also a very positive effect on the VisualDCT itself. Extensive use provides bug information and new feature requests. During this summer, VisualDCT was significantly updated. Several new features were introduced, from requests primarily coming from the users.

VisualDCT uses ANT [8] scripting language for build. This enables the developers to produce automated nightly builds and publish the results on the internet [7].

Configuration of VisualDCT was simplified. VisualDCT now uses Java Preferences API [9] (which comes with Java 1.4) and stores its configuration information on the system dependent backing store (Windows registry on Microsoft platforms or .files on Linux).

Multiple DBD files can now be used on a single DB file. Path to the needed DBDs is then stored in the DB. VisualDCT also offers a selection dialog window in which the user can manage all the DBD files in use.

As addition to groups, records and links, new graphical components have been added. The user can now create arbitrary comments in textboxes which also support html. This enables users even add images right into the VisualDCT. Other graphical objects such as lines, arrows and rectangles can be also used to illustrate the database.

The user has now the possibility to choose fields to be displayed. By default, VisualDCT displays only fields with non-default values. By clicking with the right mouse button on the field description in the Inspector dialog, the user can force the field to be displayed either always or never or in default mode. The current mode for the field is indicated by a small eye image next to the field description. The new Inspector dialog has now also the ability to display fields in alphabetical order or the order in which they are defined in the dbd file.

Inspector now compares format of all input and output fields to the definition in the `dbd` file. Any mismatch is represented with red text color.

* - rok.sabjan@cosylab.com

To improve on printing performance, VisualDCT now supports postscript. It is possible to export the visual contents to a postscript file or directly print it on a postscript printer.

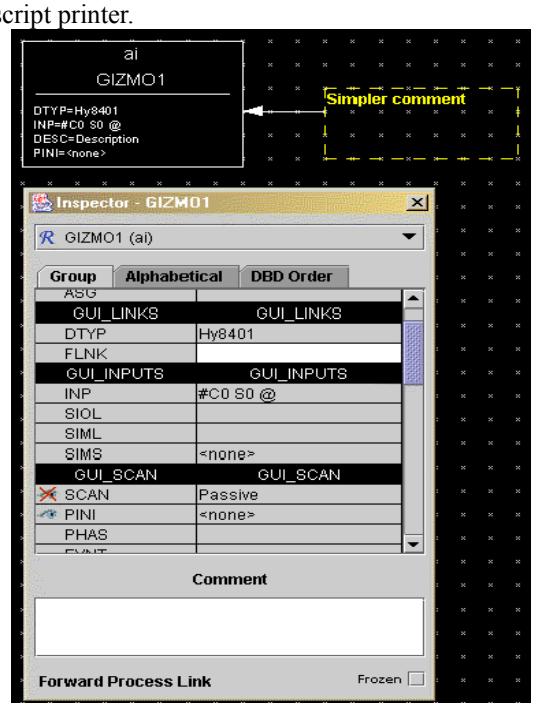

Figure 1: Graphical comments and new Inspector dialog

3 HIERARCHY SUPPORT

Some other database configuration tools also included support for some sort of hierarchical design. A new hierarchy design has been used for VisualDCT, which will be considered at integration of hierarchy support into EPICS base (presumably version 3.15). VisualDCT remains backward compatible as the hierarchy can be flattened into normal EPICS database.

The current EPICS template substitution mechanism is very restricted in its capabilities. It requires two input files (`.template` and `.substitutions`) that have radically different syntaxes, and it only allows macros to be passed downwards into a template instance. Hierarchical templates as implemented by VisualDCT must allow macro values to be passed into the template instance (giving values for fields within the expanded template), and values to be exported from the template instance to the higher level (usually the destination field name for a link in a record defined in the higher level `.vdb` file).

3.1 Expand and template statements

Macros are defined in an **`expand`** (Figure 2) statement and pass information into a template; ports are a kind of macro defined in a **`template`** (Figure 3) statement that pass information upwards out of a template instance to their calling database. Here's an example of a top-level file using the proposed syntax:

```
record(calc,"slide1:error") {
          field(INPA,
"$(slmot1.position)")
          ...
        }

        expand("slideMotor.vdb",
slmot1) {
          macro(name, "sm1")
          macro(address, "4")
          macro(demand,
"slide1:demand.VAL")
        }
```

Figure 2: Example of the expand statement

Note that we're using the macro syntax **$(template_instance_name.port_name)** to bring port values from the template instance into the higher level diagram. Unfortunately we have to allow port macros to be used before the related expand statement appears in the parent `.vdb` file, so the database flattening tool will have to make two passes through the data and should also detect loops in port/macro definitions.

When performing macro substitutions within strings, if a macro name is undefined the macro name and its surrounding **`$()`** characters will be left unchanged in the flat `.db` file. This allows templates to be used when creating a database that still takes macro arguments on loading with **`dbLoadRecords()`**. For undefined port macros though an error should probably be reported instead (but remember that these can't be properly checked and substituted until all **`expand`** statements and their related templates have been read in).

Inside the template file slideMotor.vdb, we can define ports using the new **template** statement. Ports are usually going to contain **record.field** names, but they can be literal strings and may use macros in their value:

```
template("Description of the Slide
Motor template...") {
          port(speed,
"$(name):speed.VAL", "Record to set
motor speed mm/sec")
          port(go,
"$(name):startmoving", "Forward link
to this to cause movement")
          port(position,
"$(motor.position)", "Current position
of the slide")
          port(greet, "Hello, world!",
"Just being friendly...")
        }
```

Figure 3: Definition of ports in a template file

Templates have special representation in VisualDCT. They are displayed almost like records. Inside the rectangle, template's macro values are shown and the user

can access template's ports as any other link fields of a record. The user can also use SHIFT-click technique on it to open the template itself:

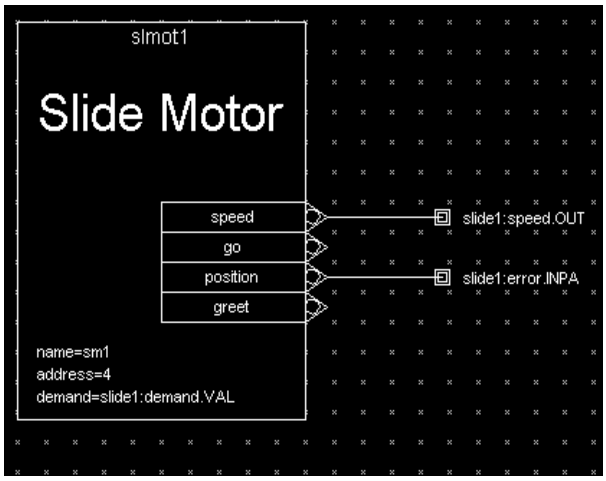

Figure 4: VisualDCT has special graphical representation for template instances

3.2 Database Flattening

In order to make database useful for EPICS all the templates have to be instantiated (e.g. the database needs to be flattened).

VisualDCT is the first tool that is capable of flattening this new database hierarchy syntax (capFast designs have always had similar hierarchical template capabilities), although at least one other will be produced in Base R3.15 for use with gnumake. The flattening process involves expanding all templates and replacing the macro and port macro variables with their strings. If a macro name is found that has no definition within its scope, it will be left exactly as it was found, which allows load-time macros to be used. We also strongly recommend that flattening tools mark the beginning and end of each template file in the flat database file using a comment as follows, to provide a way for other tools to refer back to the original template from the flat file:

```
 #
expand("/full/path/to/template.vdb",
instance_name)

          ... expanded contents of
template.vdb

  # end (instance_name)
```

Figure 5: Flattening tool comments

4 CONCLUSION

Taking into account all aforementioned features, we can justifiably state that EPICS has obtained a very powerful development tool, beating the competition in capability and user friendliness. We must not forget that a large proportion of EPICS community still uses text editors to configure their databases. With extensive use of shell scripts and manual editing, it gets very hard to maintain a large control system. So, though EPICS is very scalable, it gets very messy over a period of time and drives a lot of human resources to maintain all the databases.

We believe that VisualDCT can help tremendously in this area. Even the costs of initial database configuring are cut, but the biggest gain may occur after a month or a year, when something needs to be changed. Visual representation proves to be invaluable ("one picture is more than a thousand words"). This is proved by the use of VisualDCT around the world. It is even being used to reverse engineer text databases.

Another benefit for using VisualDCT is that using it is plain fun. Even computer non-experts are perfectly able to use it, because it requires mainly just common sense. With new updates regarding user interface, one can easily produce attractive databases.

VisualDCT can as such easily be declared as a RDD (rapid database development) tool, signalling a fresh wind in EPICS database design.

5 ACKNOWLEDGEMENT

We would like to express our gratitude to Steve Hunt of the Swiss Light Source [10] who was the first to encourage the development of VisualDCT and was the funder of the original version, which was the product of collaboration of SLS and Jozef Stefan Institute (JSI) [11].

The latest upgrades were produced by the newly founded Cosylab [12]. and financed by the Advanced Photon Source (Argonne National Laboratory) [13] and Diamond Synchrotron Light Source (UK) [14]. Their cooperation indicates the strong commitment of EPICS community to VisualDCT for what we are very thankful.

Last but not least, I would like to thank John Maclean and Andrew Johnson of APS who have actively contributed with many great ideas and have also been our greatest critics.

5 REFERENCES

[1]http://visualdct.cosylab.com
[2]http://www.aps.anl.gov/epics
[3]http://www.phase3.com/epics.html
[4]http://www.aps.anl.gov/asd/controls/epics/EpicsDocumentation/ReleaseNotes/GDCT313.html
[5]http://http://www.cosylab.com/PageFiles/Articles/ICALEPCS01-Visual_DCT/Visual_EPICS_Database_Configuration_Tool.pdf
[6]http://java.sun.com/j2se/
[7]http://www.cosylab.com/visualdct/builds/VisualDCT/
[8]http://jakarta.apache.org/ant/
[9]http://java.sun.com/j2se/1.4.1/docs/guide/lang/preferences.html
[10]http://www.sls.psi.ch/controls/controls-home.html
[11]http://kgb.ijs.si/
[12]http://www.cosylab.com
[13]http://www.aps.anl.gov
[14]http://www.diamond.ac.uk