AUTOMATIC CREATION OF LabVIEW NETWORK SHARED VARIABLES

Thomas Kluge, Siemens AG, Erlangen, Germany
Harm Schroeder, ASTRUM IT GmbH, Erlangen, Germany

Abstract

We are in the process of preparing the LabVIEW controlled system components of our Solid State Direct Drive® experiments [1, 2, 3, 4] for the integration into a Supervisory Control And Data Acquisition (SCADA) or distributed control system. The predetermined route to this is the generation of LabVIEW network shared variables that can easily be exported by LabVIEW to the SCADA system using OLE for Process Control (OPC) or other means. Many repetitive tasks are associated with the creation of the shared variables and the required code. We are introducing an efficient and inexpensive procedure that automatically creates shared variable libraries and sets default values for the shared variables. Furthermore, LabVIEW controls are created that are used for managing the connection to the shared variable inside the LabVIEW code operating on the shared variables. The procedure takes as input an XML spreadsheet defining the required input. The procedure utilizes XSLT and LabVIEW scripting. In a later state of the project the code generation can be expanded to also create code and configuration files that will become necessary in order to access the shared variables from the SCADA system of choice.

PROBLEM INTRODUCTION

Using LabVIEW [5] network shared variables [6] makes it easy to connect LabVIEW controlled hardware components to a SCADA or distributed control systems via OPC [8] (see Fig. fig:ExVarDef). You have to provide a set of network shared variables and LabVIEW code that reacts on changes to these variables or updates them. LabVIEW's shared variable engine provides the OPC server.

However, when creating and using shared variable libraries in LabVIEW we've been facing the following tasks that require similar or even identical actions or coding for each variable.

- creating a shared variable library,
- creating a shared variable,
- initializing the value of a new shared variable,
- opening and closing the connection to a shared variable,
- reading and writing shared variable values.

Later, when the connection to a SCADA or distributed control system has to be done, for each library configuration files will have to be written that describe the shared variable libraries for the integration into the system. With multiple instances of the same hardware component the variable creation tasks are multiplied with the number of instances.

The effort for this tasks is significant as it is comparable to the implementation of the real functionality of the code that connects the shared variables with the hardware driver calls.

MATERIALS AND METHODS

Shared Variable Library Description

A shared variable library is identified by a location URL and a name. The library can contain variables of simple types (integer, floating point, boolean, string) and one-dimensional arrays of simple types. Each variable has a unique name inside the library. For each variable we want to define a default value that can be used for initialization of newly created variables.

Required LabVIEW Clusters

For the generic LabVIEW functionality described later we need a set of LabVIEW clusters for each shared variable library. The clusters for each shared variable in the library contain one element named identically to the shared variable in order to store specific information.

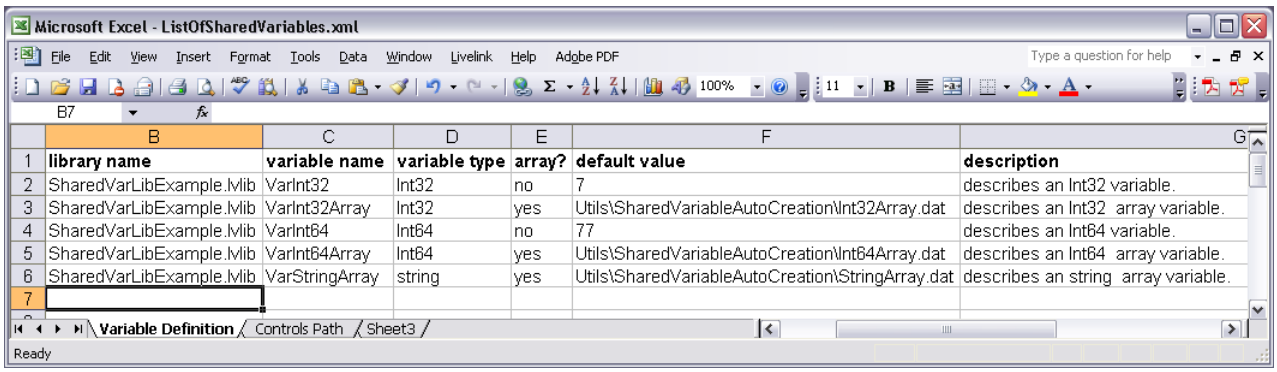
- The elements of the “type cluster" have the same type as the shared variable. The “type cluster" can be used for getting information about the data types of the shared variables and for storing values of the shared variables.
- The elements of the “refnum cluster" are LabVIEW's shared variable refnums. They are used for maintaining information about open connections to shared variables.
- The elements of the “access cluster" are booleans. They are used for controlling which shared variables are accessed by generic LabVIEW functionality.

Generic LabVIEW Functionality

Using references to the LabVIEW controls just introduced the following functionality can be provided by generic LabVIEW Virtual Instruments (VI) for all or only a subset of the shared variables inside a library

- creating a variable,
- opening the connection to a variable,
- closing the connection to a variable,
- reading values,
- writing values,
- pre-setting access control cluster.

Shared variable library specific VIs providing this functionality for the clients of the shared variale library just use references to the specific clusters and call these generic VIs.



	B	C	D	E	F	G
1	**library name**	**variable name**	**variable type**	**array?**	**default value**	**description**
2	SharedVarLibExample.lvlib	VarInt32	Int32	no	7	describes an Int32 variable.
3	SharedVarLibExample.lvlib	VarInt32Array	Int32	yes	Utils\SharedVariableAutoCreation\Int32Array.dat	describes an Int32 array variable.
4	SharedVarLibExample.lvlib	VarInt64	Int64	no	77	describes an Int64 variable.
5	SharedVarLibExample.lvlib	VarInt64Array	Int64	yes	Utils\SharedVariableAutoCreation\Int64Array.dat	describes an Int64 array variable.
6	SharedVarLibExample.lvlib	VarStringArray	string	yes	Utils\SharedVariableAutoCreation\StringArray.dat	describes an string array variable.
7						

Figure 1: Example of a shared variable library definition. The XML spreadsheet can be edited with Microsoft's Excel.

Automatic Code Generation

The information about the shared variables and the names of the generated artifacts are stored in an XML file [10] that conforms to the Microsoft XML Spreadsheet [13] schema [12]. This format makes it easy to edit the information and allows us to process it with XSLT stylesheets [11]. The XML file is XSLT-transformed into an XML string that conforms to the LVData XML schema [7] and is then converted into a LabVIEW cluster. This cluster is used by LabVIEW scripting [9] for the generation of all shared variable library specific clusters and VIs.

EXAMPLE

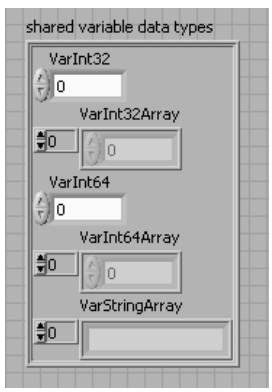


Figure 2: Example of an auto-generated "type cluster".

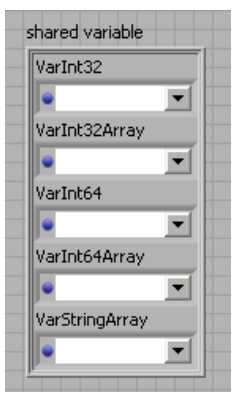


Figure 3: Example of an auto-generated "refnum cluster".

Figure 4: Example of an auto-generated "access cluster".

NEXT STEPS

The tool-chain we have introduced will be used in a next step for decoupling the LabVIEW graphical user interfaces from the underlying hardware control functionality. After this has been done the hardware functionality will be connected to a SCADA system. The tool-chain will be expanded in order to generate the SCADA system specific configuration files for the shared variable libraries.

SUMMARY

LabVIEW scripting has successfully been utilized for LabVIEW code generation allowing significant reduction of recurring work for generating and using LabVIEW shared variable libraries. The LabVIEW-independent XML input information can easily be utilized for further text-base code generation within the build process.

Figure 1 shows a screen-shot of the spreadsheet defining a shared variable library. The auto-generated LabVIEW clusters are shown in Figures 2, 3 and 4. Figure 5 gives you an impression of a generic VI that operates on the auto-generated clusters. The VI shown in Figure 6 is auto-generated and calls the generic VI with references to specific cluster instances. Finally, Figures 7 and 8 illustrate the use of LabVIEW scripting for creation of the cluster shown in Figure 2.

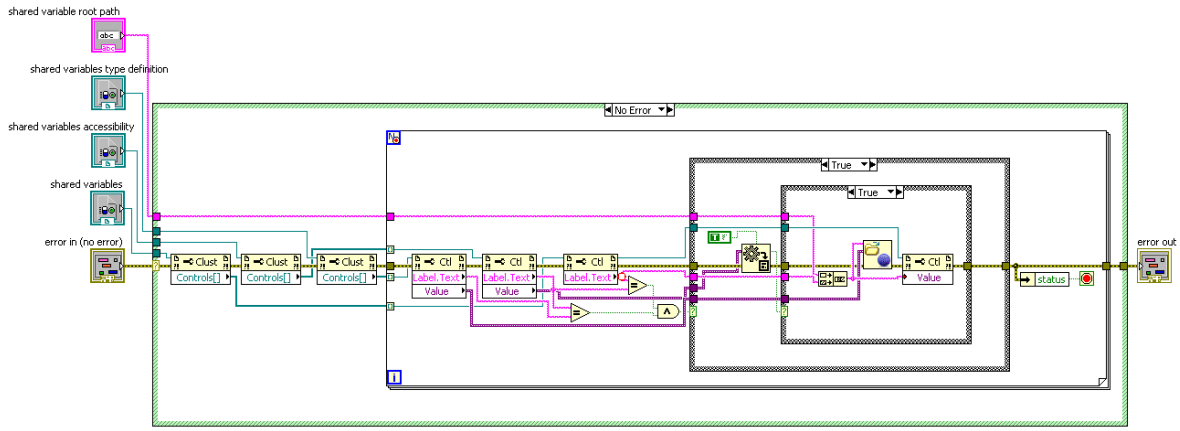


Figure 5: Example of a generic VI. The block diagram shows how shared variables are opened using the auto-generated clusters.

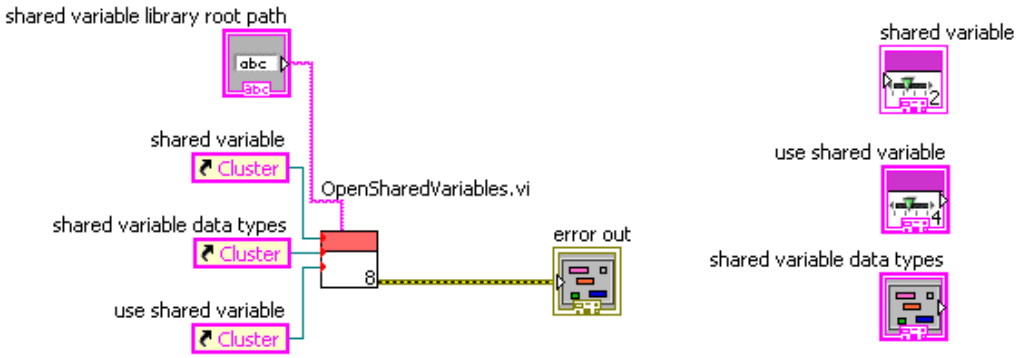


Figure 6: Example of an auto-generated VI that uses the generic VI from Figure 5 for opening shared variables from a specific library.

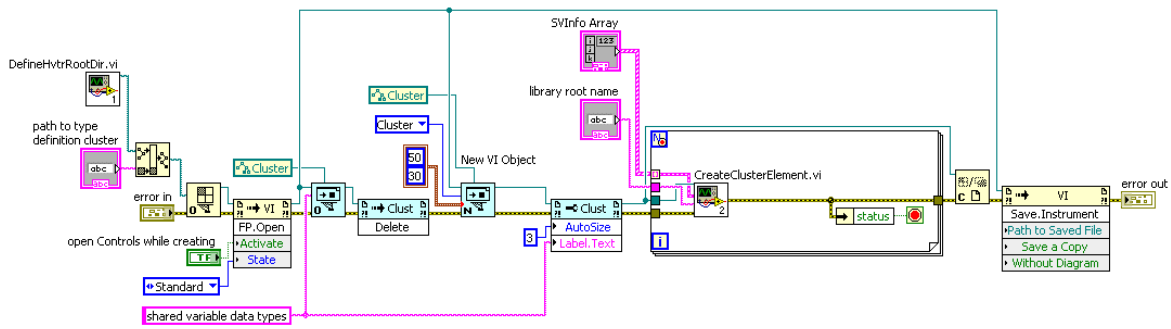


Figure 7: Example of a LabVIEW scripting VI. This VI creates the cluster shown in Figure 2. The sub-VI CreateClusterElement.vi is shown in Figure 8.

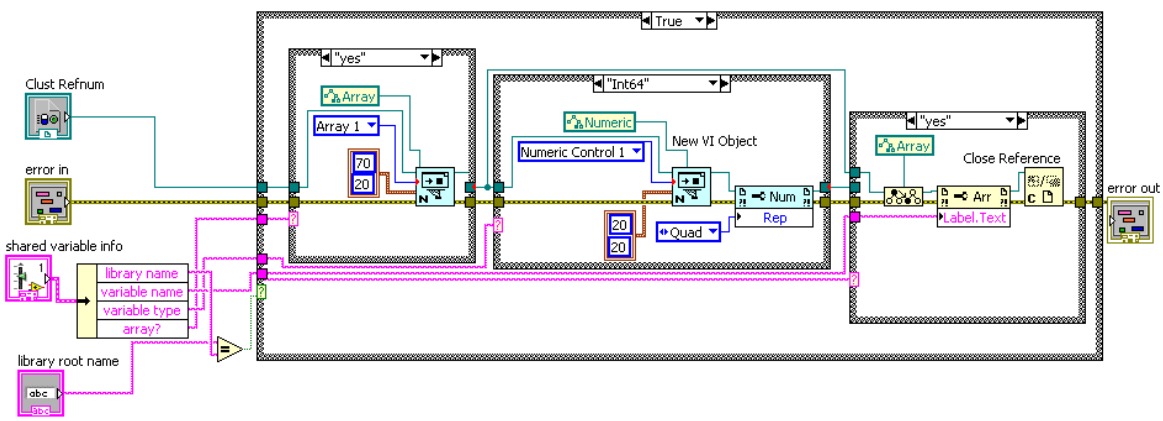


Figure 8: Example of a LabVIEW scripting VI. This VI creates the elements of the cluster shown in Figure 2.

REFERENCES

[1] O. Heid, T. Hughes, THPD002, IPAC10, Kyoto, Japan

[2] R. Irsigler, et al, 3B-9, PPC11, Chicago IL, USA

[3] O. Heid, T. Hughes, THP068, LINAC10, Tsukuba, Japan

[4] O. Heid, T. Hughes, MOPD42, HB2010, Morschach, Switzerland

[5] National Instruments LabVIEW, http://www.ni.com/labview

[6] Using the LabVIEW Shared Variable, http://zone.ni.com/devzone/cda/tut/p/id/4679

[7] LVData XML Schema, LVXMLSchema.xsd comes with LabVIEW installation in vi.lib/Utility sub-folder

[8] The OPC Foundation, http://www.opcfoundation.org/

[9] LabVIEW Scripting, https://decibel.ni.com/content/docs/DOC-4973

[10] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml

[11] XSL Transformations (XSLT), http://www.w3.org/TR/xslt

[12] XML Schema, http://www.w3.org/TR/xmlschema-1 and http://www.w3.org/TR/xmlschema-2

[13] Microsoft Office 2003 SpreadsheetML, http://msdn.microsoft.com/de-de/library/aa140066.aspx and http://www.microsoft.com/download/en/details.aspx?displaylang=en\&id=101