DSC Upgrade to Legacy Tag System

Authors:
Jeff Scott, Director of Operations, Data Science Automation, Inc.
Ryan W. Vallieu, Senior Automation Systems Architect, Data Science Automation, Inc.

NI Product(s) Used:
LabVIEW 2014
DSC Module

Category:
Industrial Machinery & Control

The Challenge
Migrating a legacy DSC application with four hundred plus tags from LabVIEW 7.1 and DSC Wizard generated tag monitoring code loops to LabVIEW 2014 to increase the system capability for scaling and code adaptability and maintainability.

The Solution
Data Science Automation combined the benefits of the National Instruments Shared Variable Engine, Datalogging and Supervisory Control module, User Event Structures and DAQmx drivers to create a more upgradable, maintainable and customizable system for control and datalogging.

Introduction
Data Science Automation (DSA) is a premier National Instruments (NI) Alliance Partner that specializes in automating and educating the world leading companies. Clients choose DSA because of DSA's deep knowledge of National Instruments products, disciplined process of developing adaptive project solutions, staff of skilled Certified LabVIEW Architects and Certified Professional Instructors, and unique focus on empowerment through education and co-development.

Migratory Headaches
DSA was hired to complete a LabVIEW version/DSC migration project that was originally attempted by our customer. The original system was created in LabVIEW 7.1 and was running on Windows XP which has lost support by Microsoft and forced a system upgrade for our customer. They originally tried to migrate to LabVIEW 2014 through the use of Shared Variables and binding the shared variables to the DAQmx channels in a similar manner to the original system that was utilizing DAQ-OPC. The client ran into issues misunderstanding the migration path to LabVIEW 2014/DSC 2014 from DSC 7.1. They originally only tried to upgrade and replace the SCF DAQ-OPC tags with the Shared Variables linked to the DAQmx channels, and did not understand that the DSC 7.1 VIs and functions had been completely deprecated in LabVIEW 2014.

Original System Architecture
Figure 1 shows the original system architecture as implemented by the original system developer. The original developer heavily leveraged the DSC Wizard functionality within the LabVIEW environment to create alarm and tag monitoring loops on the block diagrams of each main screen VI. Therefore there were around four hundred parallel loops running in the system each tied to a tag on multiple block diagrams. This made the code very difficult to read since the block diagrams were very large.

In addition each tag was also bound to the front panel controls and indicators. DAQ-OPC was leveraged to bind the tags directly to the DAQ channels.

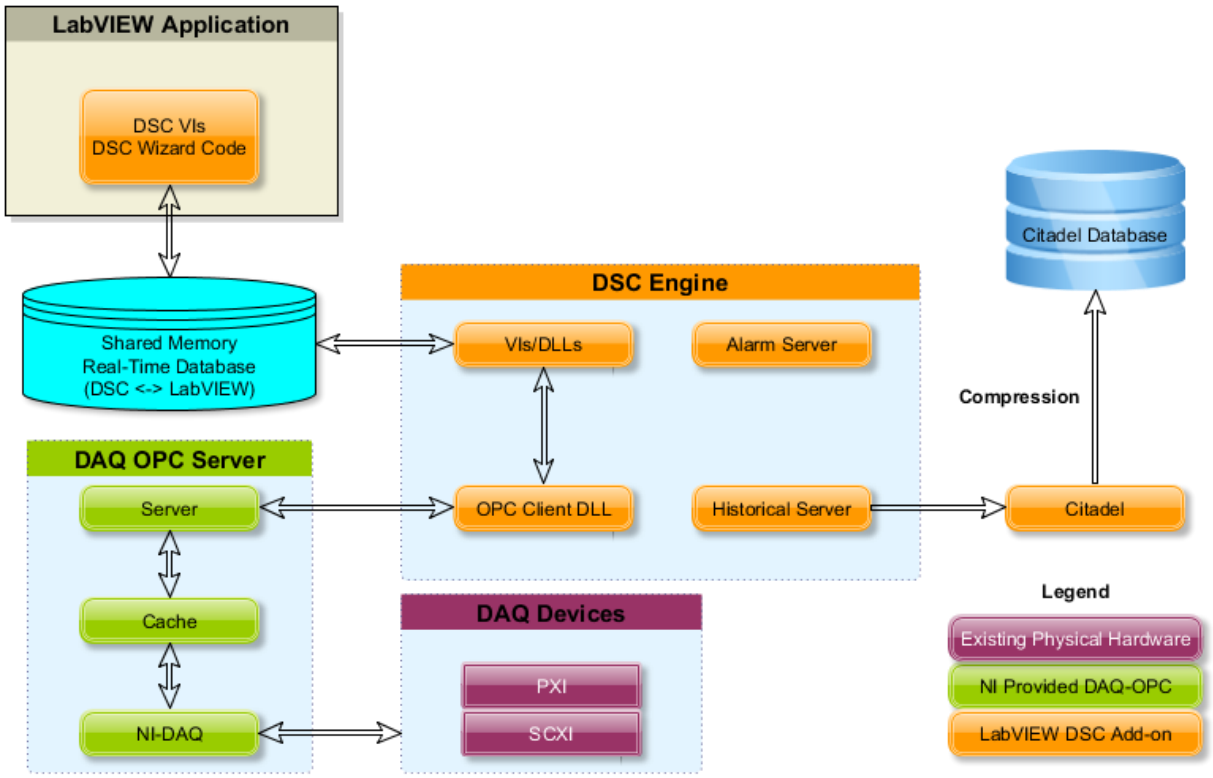

Figure 1: Original System Architecture

Utilizing Experts

Our customer came to us for help in finalizing this upgrade due to our Alliance Partner affiliation with National Instruments and our expertise with LabVIEW. After requirements gathering and understanding the true desires of the customer for the new and future features they wanted to integrate into the system without losing the capabilities of the existing system, DSA was able to propose a robust and adaptable solution.

There were more than a few upgrade paths possible based on the multiple NI system offerings and the capability of LabVIEW.

DSA proposed to leverage Shared Variables (SV) and the NI Shared Variable Engine (SVE) through programmatic access and leverage DSC through the SVE and DSC Palette. Programmatic variable access was recommended since this is an application with a larger number of variables with the possible need to adapt hardware changes in the future. In the Managing I/O with Shared Variables (DSC Module) topic of LabVIEW Help this programmatic SV approach is covered and the comparison is recreated on the following table (Table 1):

Data Access Method	Recommended Use	Supported Method	Required Product
Static	Small applications that contain a small number of shared variables and do not gain in complexity over time.	Shared Variable nodes	LabVIEW Development System
		Front panel data binding	
Programmatic	Applications that contain a large number of shared variables or require dynamic creation of shared variables.	Shared Variable VI and functions	
		Shared Variable Event Structure support	LabVIEW Datalogging and Supervisory Control (DSC) Module
		Tags VIs	

Table 1 - Utilizing the SVE with LabVIEW

The use of the LabVIEW DSC module with the Shared Variable approach opened up the possibility of utilizing Event Structure support for Shared Variables. This was the backbone of the base architecture for the operator visible control and monitoring screens.

Direct binding of the SV (tags) could have been utilized if desired as this capability does exist if global virtual channels are utilized along with the creation of an I/O Server. The reason this was not chosen was due to the following issues:

- Labor intensive creation of global virtual channels
- Labor intensive linking of shared variables to the virtual channels
- Limited scaling capabilities

DSA utilized a custom written LabVIEW I/O Engine based on DAQmx VIs to access the hardware for the I/O and utilize the Shared Variable API VIs (not the Shared Variable Static nodes) and LabVIEW DSC Tag VIs to push/pull the data to the Shared Variables (tags). Using this approach the source of the data is configured at run-time and loaded from a configuration file. Future upgrade changes are minimized by modular code that encapsulates the source that pushes the data to/from the control and monitoring code through the SVE. In this case an Acquisition Engine was utilized to discretely run Digital Input, Digital Output, Analog Input and Analog Output modules in parallel with the Shared Variable Engine. Analog Input and Digital Input engines utilized the SVE Programmatic API write VIs to update the system with the data values. Outputs for Analog and Digital controls were handled by User Event Structures configured to monitor for Value Changes on front panel controls and updating the Variables through the DSC Tag Write VIs on the block diagram of the control/monitor screens exposed to the operator. The DAQmx Output Engine then monitored the tags for Value Change Events and updated the DAQmx output engines respectively.
Software Scaling was implemented rather than utilizing the built-in scaling capability of DAQmx Scales and the scaling available within the shared variables due to the desire to implement complex custom scaling in the future. This scaling is applied in the I/O engine and the scaling information is stored in the hardware configuration file along with the physical channel information needed to programmatically create the DAQmx channels and tasks.
Figure 2 shows the proposed software architecture for the system upgrade. This design was based on the maximum capability and scalability for the system that would satisfy the customers' requirements within the allotted project timeline. Not shown on the drawing is the hardware configuration file that maps the Shared Variables to the DAQmx Channels and describes the software scaling formulas for each channel.

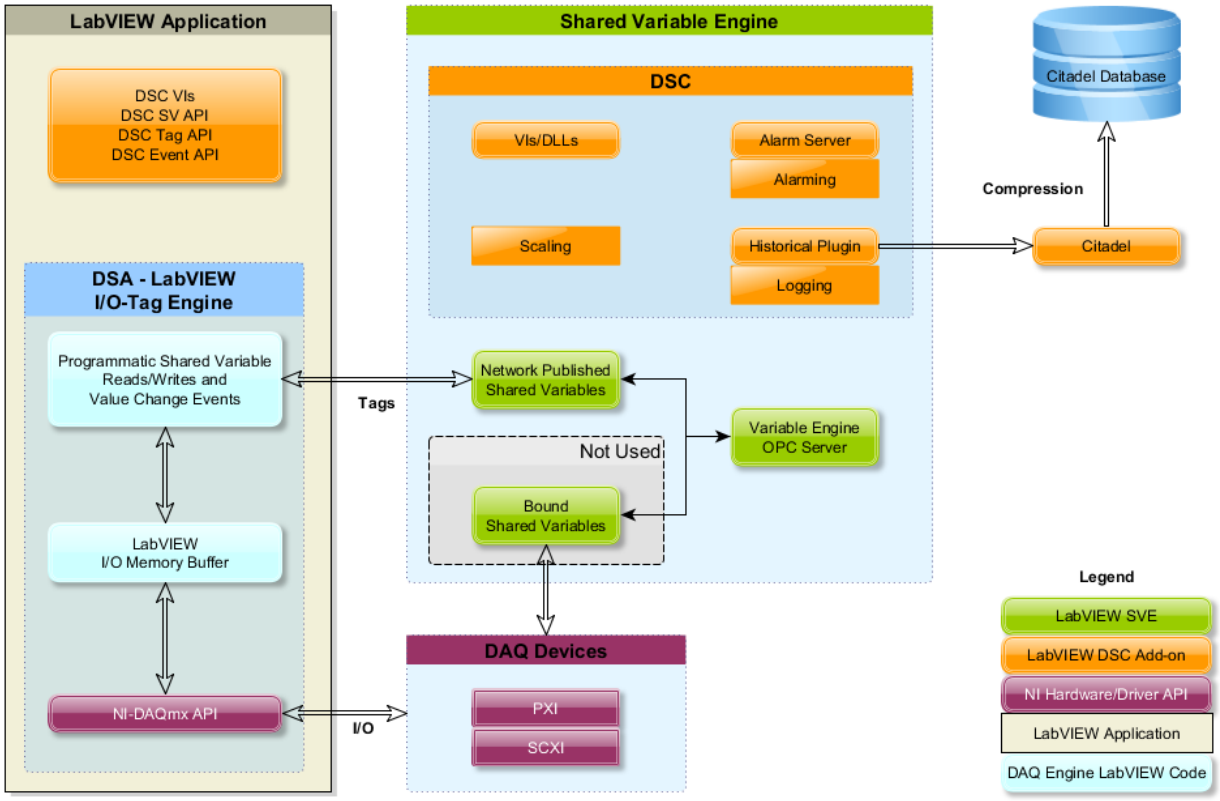

Figure 2: Proposed System Upgrade Architecture

Easy Replication and Scalability

The utilization of LabVIEW DSC brought with it built-in alarming and logging capabilities that the customer was already used to leveraging in the legacy solution. Carrying over the original Citadel database was not required, and thus logging capabilities were built-in to the system in a similar manner.

The real gain in code maintainability and adaptability with utilizing the DSC module with SVE is the support of Event based programming. A generic base-line architecture was created to be the basis of any front panel screen that the operator would be interacting with. Each screen was centered on a particular piece of the hardware system and registered for tag updates and alarms for the data tags relevant to that piece of equipment. This condensed the original DSC Wizard Code with hundreds of Wizard Loops on the block diagram to a single User Event Structure. Figure 3 shows an example of the original LabVIEW 7.1 DSC Wizard generated loops that were present on the block diagrams of each operator facing screen.

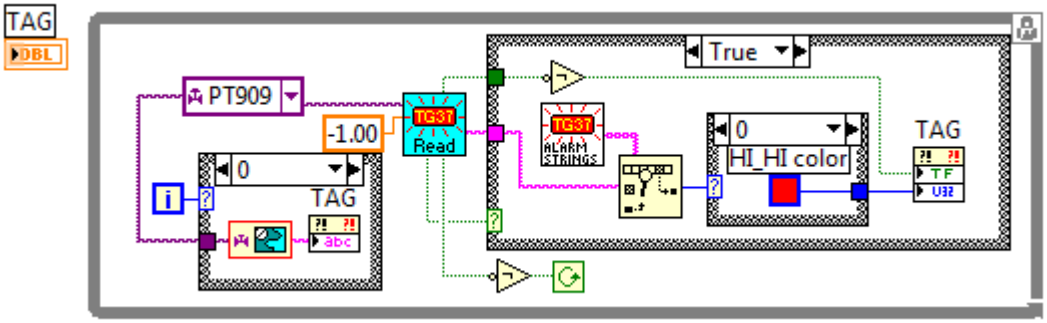

Figure 3: Legacy DSC Wizard Code

Figure 4,

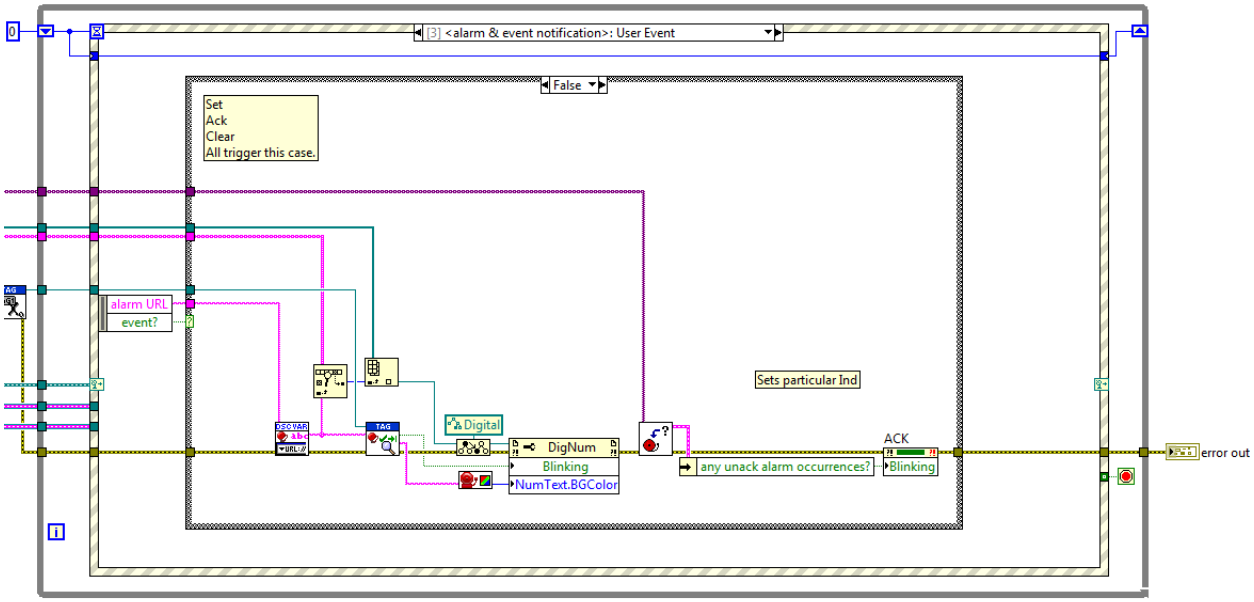

Figure 5,

and Figure 6 show the new template code and the adaptability and cleaner block diagram brought about by the use of the User Event Structure and Event based programming. The initialization code includes a template VI with tag lists that define the variables that the VI should register for either Value Change, Alarm Event Notification or Controls that will push new values out via the DSC Tag VIs.

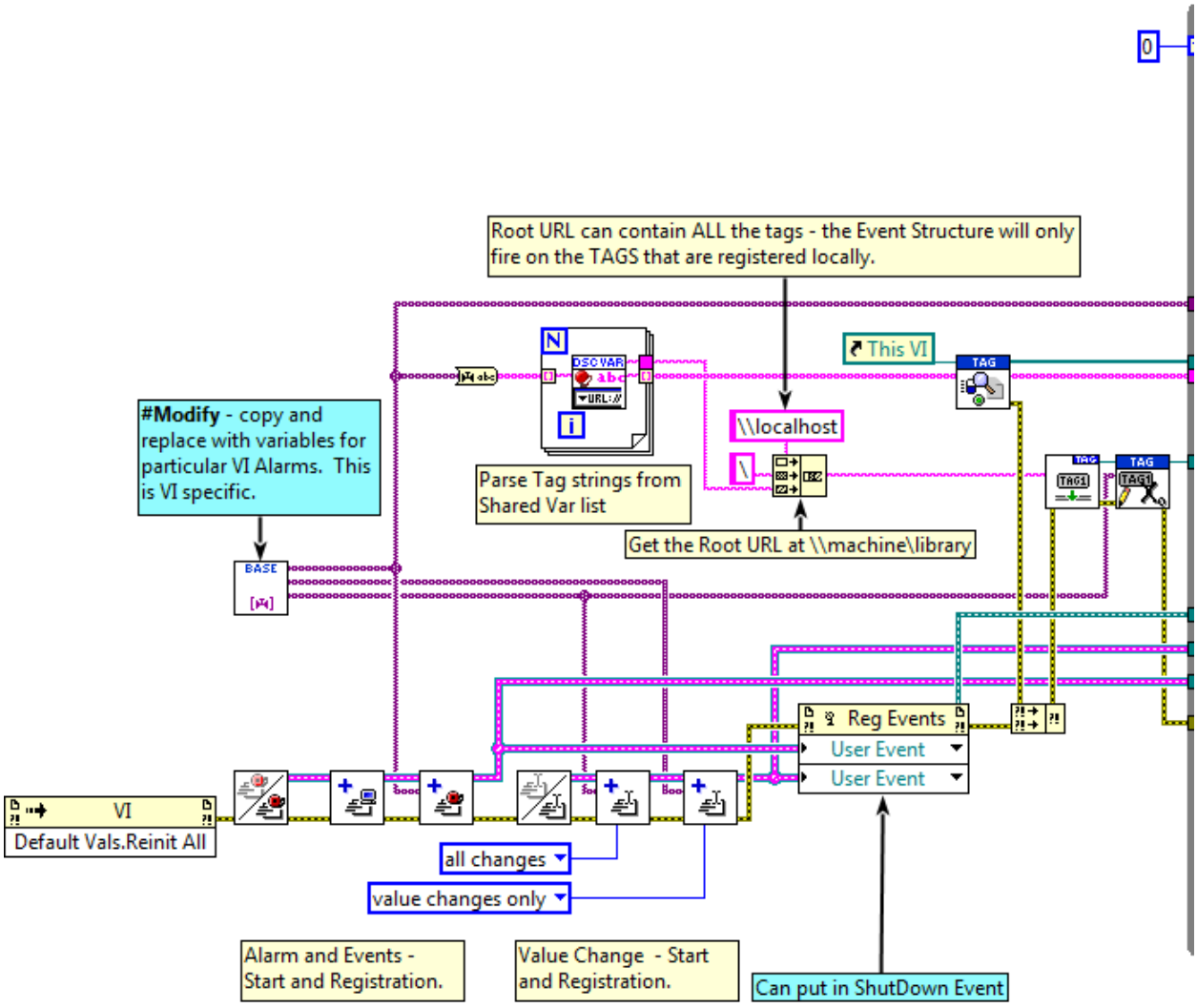

Figure 4: Initialization Code for Template VI

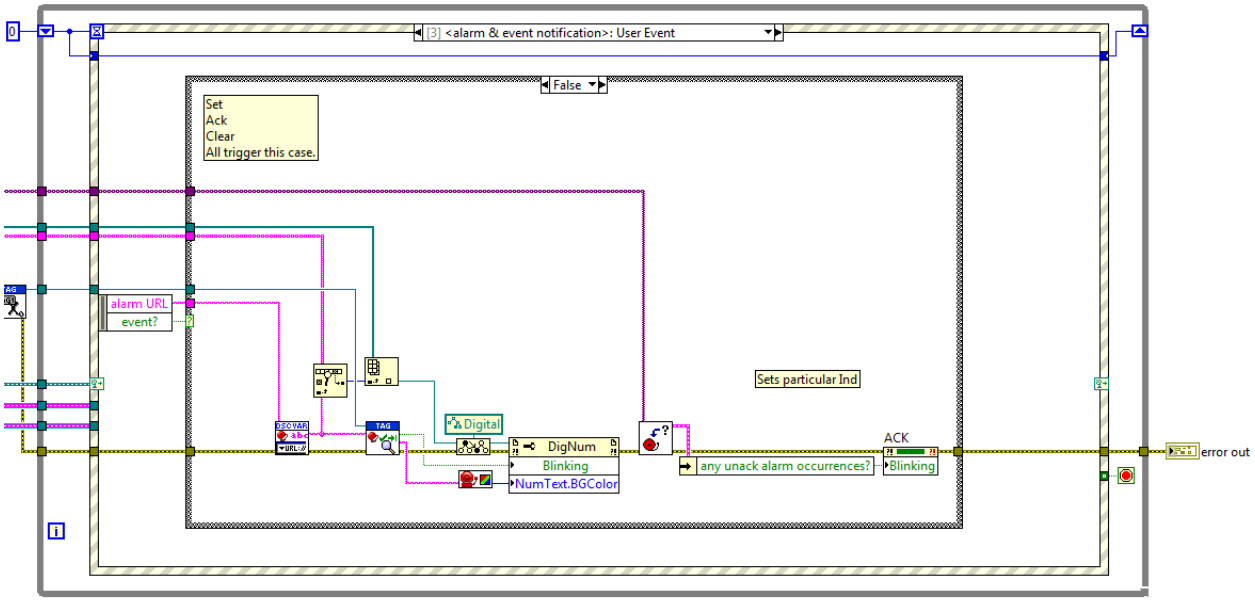

Figure 5:

Alarm Notification Handling Code

Figure 5 shows the replication of the background color manipulation for the numeric controls that were part of the original DSC Wizard Code that the customer wanted to port through with the update. In the original Legacy Code each loop tied to a tag/indicator maintained its own palette of colors for the background of the control based on the alarm level. Luckily the same palette was utilized for most of the screens, but where they were not all the same, groups of color palettes were able to be utilized and a look-up scheme was utilized to select the proper palette.

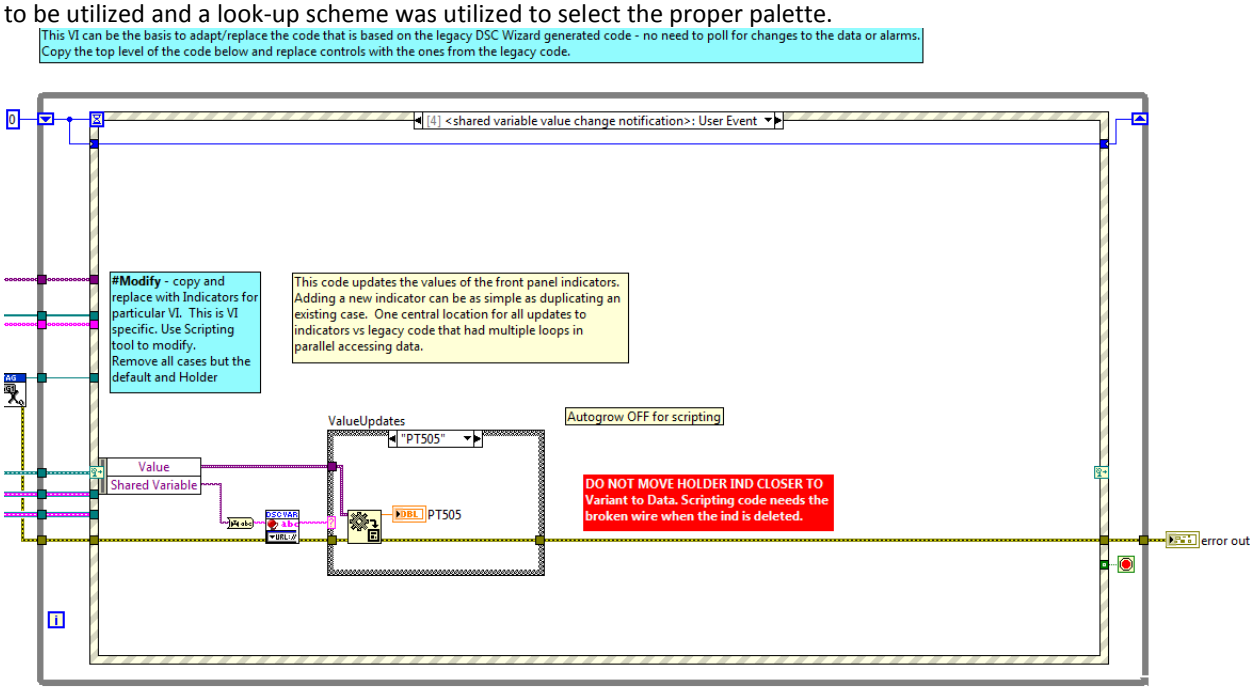

Figure 6: Tag (Shared Variable) Value Change Handling Code

Figure 6 also shows the Tag Value change code that updates the front panel indicators and controls. It was desired to avoid using the control label and control reference to modify the Value Property of the controls based on value change detection the due to the possibility of frequent value changes and the performance impact of utilizing value property nodes for hundreds of tag updates. Using the terminals was the desired method implemented.

LabVIEW Scripting Tools Reduced Coding Time

Figure 6 also shows code documentation that makes mention of Scripting Code tools that were created to help the developers prevent spelling mistakes with tag associations when adding tags to the Value Change Case Structure. Scripting code was created to take any front panel indicators that were placed inside a flat sequence structure labeled with a particular name, create a new case in the value change case structure copied from the "holder" case, delete the holder copy indicator and move the tag indicator into the case and change the case label to make the label of the Tag indicator. Not only did this prevent human spelling errors in the tag associations it also reduced coding time by removing the manual labor of moving the indicators and creating the case structure frames.

Conclusion

The leveraging of LabVIEW Shared Variable Engine with LabVIEW DSC for event based programming utilizing the User Event Structure to respond to Tag Updates greatly reduced the complexity of the original legacy code. Upgradability and maintainability were greatly increased as well. The customer also gained channel scaling ability that was not able to be done in the original system. This upgrade satisfied the customer's minimum requirements of moving from LabVIEW 7.1 to LabVIEW 2014 and Windows XP to Windows 7+. Another benefit that was realized in addition to satisfying the base requirements was much faster code reaction and front panel update time over the legacy code.

Contact Information

Ryan W. Vallieu, info@DSAutomation.com
Data Science Automation Inc., 375 Valley Brook Road, Suite 106, McMurray, PA 15317-3370
www.DSAutomation.com