white paper

A SYSTEM LEVEL APPROACH TO DESIGNING MODULAR SWITCHING SOFTWARE

RELIABLE DATA FIRST TIME EVERY TIME

VTI Instruments

Introduction

The system switch plays a crucial role in any automated test system as it provides ATE engineers the ability to distribute instrumentation I/O to multiple test points. The popularity of VXI, PXI and LXI modular instrumentation platforms created an opportunity for consolidation of instrumentation and switching within a single mainframe, resulting in high-channel count capability in a reduced footprint. A modular approach to switching systems also affords the ATE architect design flexibility through the use of domain-specific modules (i.e, power, RF, and low-frequency) that can be combined to cover a broad range of the signal spectrum. Unfortunately, many system designers treat the system switch as an afterthought and simply look to assemble a collection of individual switch modules that can accommodate the required I/O count.

The vast majority of ATE systems depend on these individual switch modules to operate together as a subsystem, interfacing through external cabling, mass interconnect panels, and test adapters, in order to construct higher channel count multiplexers or matrices. Ignoring the system perspective when assembling a switching system can lead to extended development schedules that delay the time to system readiness. The impact on signal integrity that occur when connecting multiple switches using external cabling can obviously lead to integration headaches, however, the development efforts related to programming switch states for automated tests can also contribute to schedule delays if the software infrastructure is targeted for discrete modules, particularly as the complexity of the switching system grows.

This paper will discuss software platforms and patented web-based configuration tools that VTI Instruments has developed to create a robust system-level switching programming environment that significantly reduces test program set development time, simplifies debugging and TPS validation activities, and maximizes hardware and software supportability.

Summary of Key Results

High-level tools that help in the development of path level programming require installing additional software adding to system level complexity. System integrators and end users have also developed internal solutions in an effort to simplify TPS development. While these solutions prove beneficial in the short term, they can pose long term support issues as the operational life of deployed equipment continues to grow. The software tools based on open-platform industry standards provided with VTI's LXI switching and I/O instruments such as the EX1200 and EX7000 series resolve many of these issues.

The first approach leverages the IVI Switch class application programmer's interface (API) to provide a common system level programming interface that treats multiple discrete switch modules as a single subsystem. The IVI API generally requires separate driver sessions to command and control each switch module. When input/output paths are constructed in a test system that span multiple modules using external wiring, the application developer must make a series of calls to each switch module in the chain. High-level standalone utilities have been introduced to the market allow users to create databases that incorporate knowledge of external wiring to provide a more elegant system-level programming model. These tools add cost to the development process, and the additional database files introduce a level of complexity to the software maintenance environment. VTI has developed algorithms within its IVI API that remove the need for the high-level tools to allow a single driver session to activate an entire series of individual relays across multiple switch cards.

In addition to the programming interface, VTI has also leveraged the strength of embedded web applications which provide the basis for the user interface for LXI instruments. Embedded web interfaces provide a powerful mechanism that not only enable access to instrumentation settings and measurement functionality, but also provide an innovative mechanism to address simplified path level programming. Built-in web functionality allows the user to identify and configure specific relay topology, define how the components will be connected, and the automatically generate path level drivers that reflect the current configuration. The resulting device configuration files can be uploaded to the switch instrument such that the switching system is 'aware' of all of the interconnects. If switch types and routing are modified, updates can be immediately generated to reflect the current state.

Main Contributions

The design of a switching system is highly dependent on the intended purpose of the automated test system within which it will reside. Variables such as number of test program sets supported, channel counts, frequency band and channel count guide the philosophy behind a switch design. Modular switching systems have gained wide acceptance because of the breadth of application coverage, topologies (i.e, discrete relays, matrices and multiplexers) and granularity in channel counts. Switching systems can be optimized to meet a unique test system need by mixing and matching modules to account for all the required I/O.

Traditional software models for instrumentation have historically been applied to switching products and ignore the fact that the system switch is a conduit between test instrument I/O and the device under test. Test sequences often require multiple relays on multiple modules to be closed to establish paths between stimulus and measurement channels. The traditional software model for switching, such as SCPI or plug&play, targets a relay or group of relays on a single module with no knowledge of how these relays connect to each other or to other relays on other modules to form a path. This puts a significant burden on the application developer who must have an intimate understanding of how individual relays are wired together through the mass interconnect and test adapter. Measurement paths are made by making a series of discrete relay calls and path states must be maintained within the software application creating a potentially complex management structure when multiple tests using multiple combinations of relay states are required.

When the LXI consortium mandated IVI as the required application programmer's interface, developers were presented software drivers that address signal switching as paths that connect input channels to output channels and removed the concept of controlling individual relay coils. The difference between the plug&play and IVI APIs are shown in the example below using a common SPST relay.

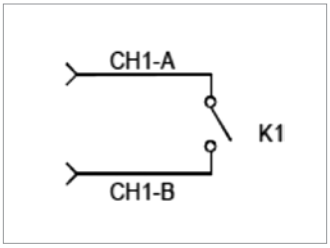

Figure 1

Plug&play

ViStatus _CloseRelay (ViSession instrHndl, ViInt16 moduleNmbr, ViInt16 relayNmbr);

Where relayNmbr is an integer corresponding to 'K1'.

IVI

ViStatus IviSwtch_Connect (ViSession Vi, ViConstString Channel1, ViConstString Channel2);

Where Channel 1 and Channel 2 are strings corresponding to CH1-A and CH1-B.

continued on next page

To create a path from a test asset to the UUT through a typical IVI Switch API, numerous 'sub-paths' must be established on switch modules to account for external wiring. The communication to each switch module in the mainframe occurs through separate driver sessions, ignoring how the switches interact within a system.

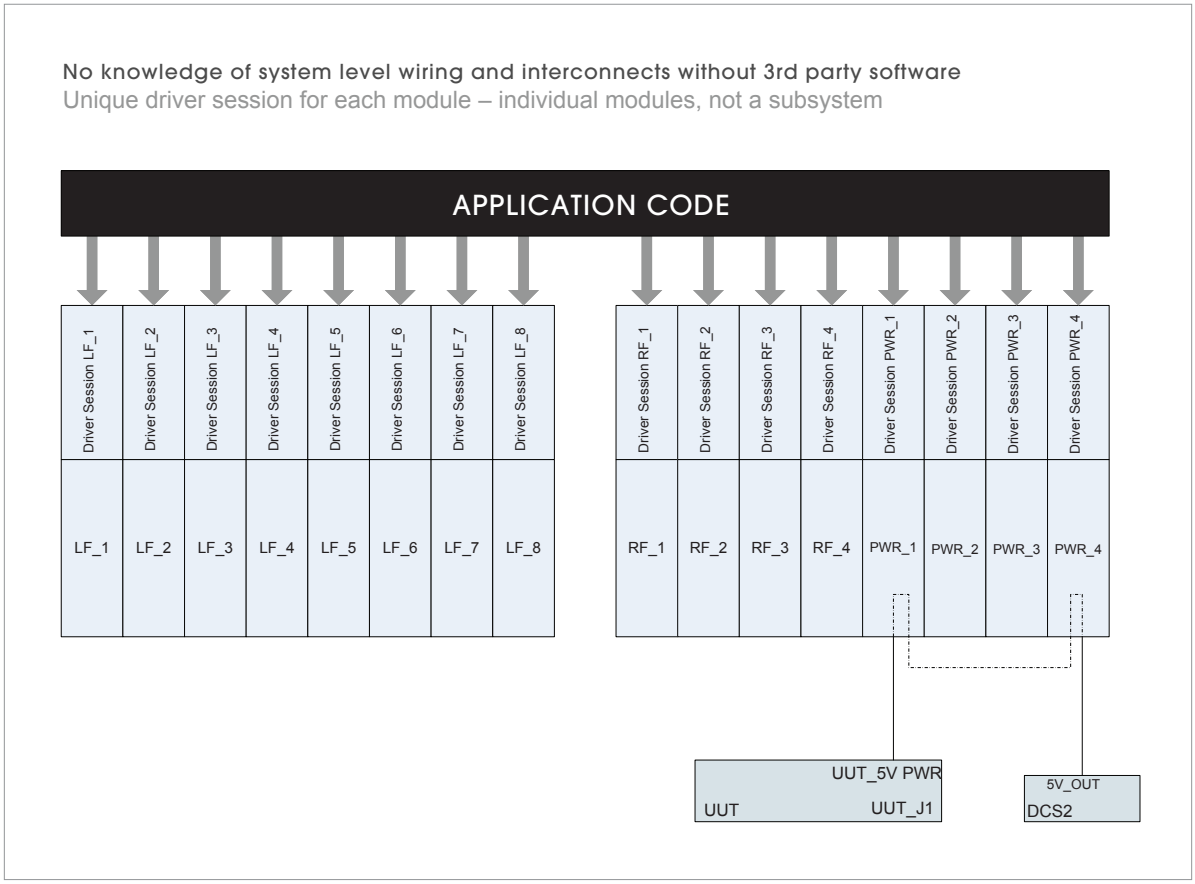

Figure 2

In this simple example, in order to connect the DCS2 power supply to the UUT, two calls are required.

ViStatus IviSwitch_Connect (SW13_Session, Ch1_A, Ch1_B);

ViStatus IviSwitch_Connect (SW15_Session, Ch3_A, Ch3_B);

continued on next page

A more intuitive approach implemented in the VTI switch API allows for the creation of switch subsystems that span multiple modules as detailed in Figure 3 in which a single session handle is created to communicate to multiple switch modules that would otherwise be controlled through dedicated sessions. Furthermore, logical channel names can be easily defined to represent the overall system architecture.

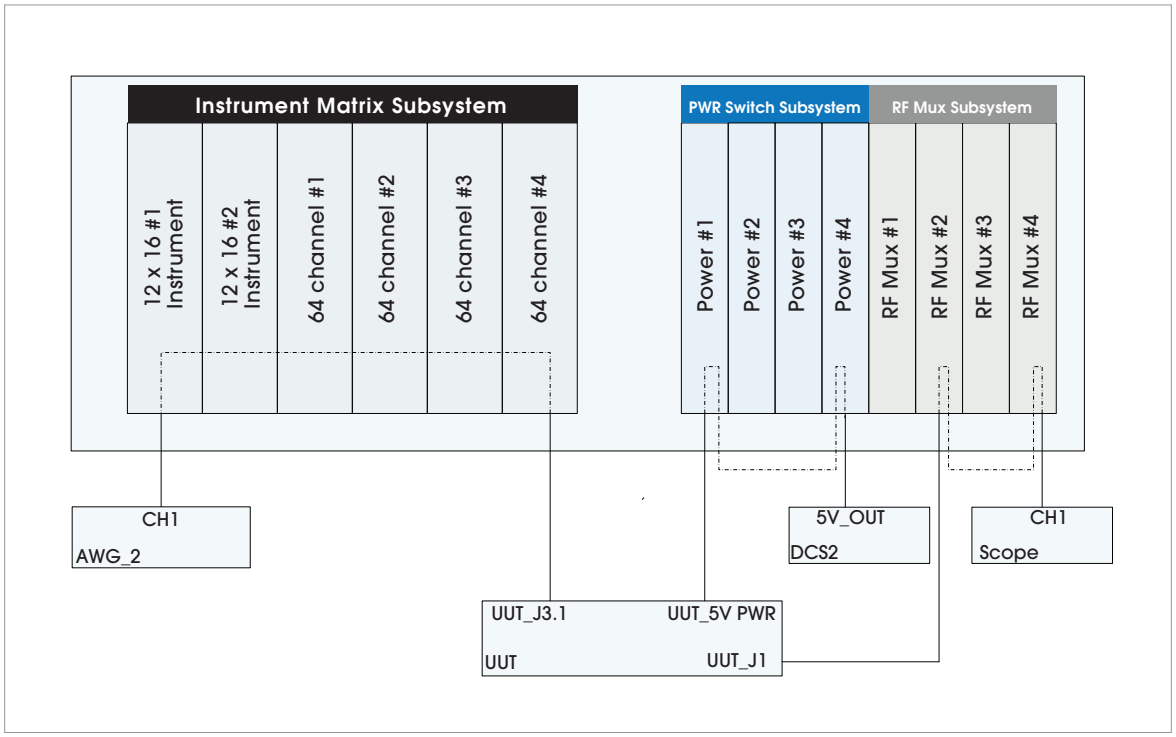

Figure 3

```
ViStatus IviSwtch_Connect (INSTR_Session, AWG1_CH, UUT_J3.1);
ViStatus IviSwtch_Connect (PWR_Session, DCS2, UUT_5V PWR);
ViStatus IviSwtch_Connect (RFMUX_Session, SCOPE_CH1, UUT_J1);
```

By creating a single entry-point into a switch subsystem, true end-end path level switching can be achieved with minimal calls creating a more elegant development environment with code that is more efficient to debug and maintain.

continued on next page

A higher level approach for creating a system level switch employs the same end-end path level philosophy, but embeds configuration and wiring knowledge in the switching hardware. VTI's patented methodology incorporates a web-based relay and component list that is uploaded in XML format to a switch control module; the wiring process can begin using the embedded configuration utility resident on the control board as shown in Figure 4.

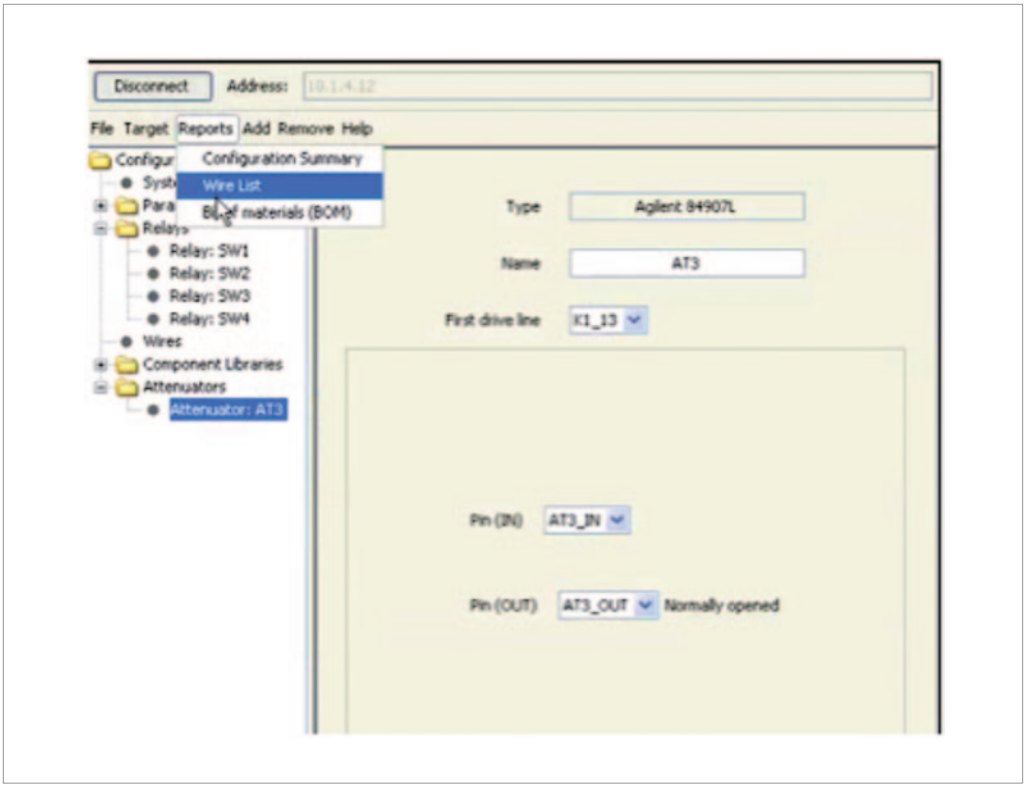

Figure 4

This intuitive process allows users to specify connections between pins on individual components (i.e, define paths) and create a file that can be uploaded to the switch control board; this effectively stores the unique personality of the switch subsystem in memory on the control board. This means that software development engineers are not required to delve into the internal details of the relay control schema and architect software based on individual components nor do they need to generate a list of possible paths.

continued on next page

Additional benefits related to debug and maintenance are also achieved by embedding system-level knowledge on the switching hardware. The XML files described above are used to generate graphical command and control interfaces that are accessed through common browser software tools such as Internet Explorer or Firefox as shown in Figure 5.

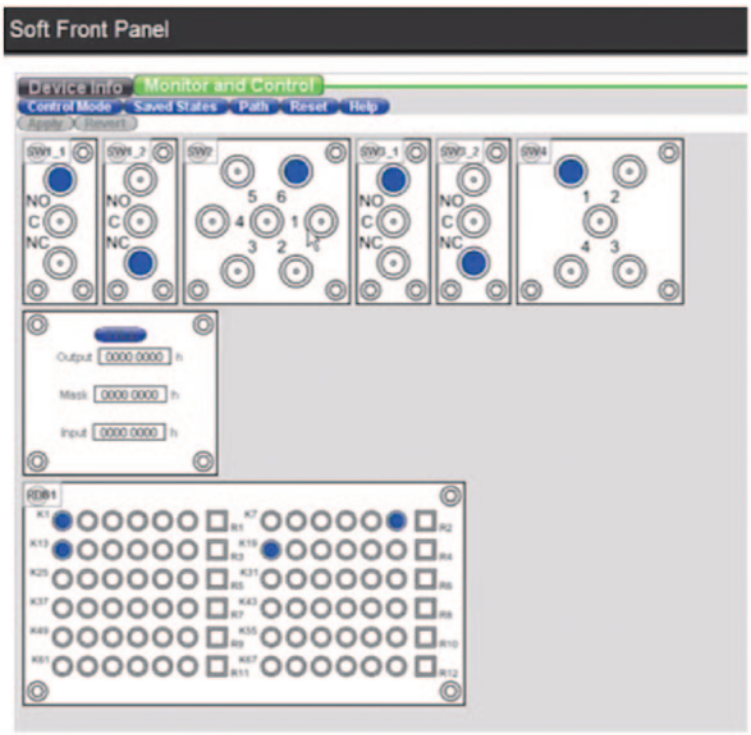

Figure 5

Summary

The signal switching subsystem is one of the most critical considerations in an ATE design. The switch system is the 'guardian' of a signal as it travels between test instrumentation and the unit under test, and extends beyond individual relay modules through external wiring that forms the input to output paths. A well-designed hardware architecture ensures that the integrity of the signal is not corrupted and yields reliable test results. New developments in switching software interfaces extend the concept of the system switch to application code development and field maintenance. By extending the industry standard IVI API to beyond the module to the system level and integrating innovative web-based configuration tools, VTI Instruments has provided a framework that cuts development time and costs when compared to the typical software model.