Design Talk: Automation

Make It Easier To Learn Instrument SCPI

By Conrad Proft, Agilent Technologies

Your job as a test engineer is to automate test equipment to verify your company's products. Many of you spend way too much time learning how to program that test equipment. Have you ever quickly and successfully configured an instrument from its front panel or Web interface and then spent hours or days trying to do that same task programmatically? Have you been frustrated with programming manuals that give you a dictionary of ASCII programming commands and few examples or interrelations between those commands? This frustration can come even if you are using instrument drivers for your programming environment.

Many instruments use the SCPI programming language which is designed to be self-explanatory and quite intuitive. Agilent requires that its products use SCPI as the native programming language. SCPI is supposed to reduce the programming efforts substantially, which it can once you become familiar with it and are using multiple instruments that utilize SCPI. Even if you are familiar with SCPI, instruments today have so much functionality built in that it can be time consuming to understand the many subsystems, parameters, and interrelations between commands. Since direct SCPI programming is used in over 50% of test systems, you need a plan for learning it quicker. If you can get your hands on any tools to help you learn quicker, you'd likely be happy to use them. Time is of the essence.

This paper does not teach you SCPI – it helps you to learn the SCPI of any instrument by describing a process. It also describes a free SCPI Learning utility that uses the process to determine the instrument SCPI commands necessary to take the instrument from one state to another. The utility is preconfigured for a number of commonly used Agilent instruments; however, once you understand the process you can easily modify the Excel spreadsheet tool to accommodate

new instruments – even instruments that are not SCPI but are an ASCII-type language that are similar in concept.

What is SCPI?
Part of the process is to understand how SCPI commands are organized and what they do. The SCPI (Standard Commands for Programmable Instruments) Consortium evolved to standardize the control language used between programmable instruments. Its aim has been to promote a common language and syntax suitable for all programmable instruments. Today, SCPI is supported by most of the manufacturers of programmable instruments including Agilent (HP), Tektronix, Keithley, Fluke, and Racal. More information can be found at www.scpiconsortium.org [1].

SCPI can be sent to an instrument over many interfaces such as LAN, GPIB, RS-232, and USB. All of these interfaces are directly supported by Agilent VISA or NI VISA I/O Libraries which make programming in the environment of your choice relatively easy, efficient, reliable, and fast. Many instruments from Agilent, for example, include several interfaces: LAN, GPIB, and USB, so you have programming flexibility for a particular environment.

SCPI is an ASCII language that consists of configuration and query commands that are specific to the instrument and a set of IEEE 488.2 operations and commands that are common to all SCPI-based instruments. SCPI commands have long and short forms, where the long form is very descriptive, and the short form is an abbreviation: “TRIGGER:COUNT” can be “TRIG:COUN”.

Every instrument has a state, and that state determines what it is going to do next. It might be configured for DCV, 5 volt range, 100usec aperture, take 10 readings on a trigger from an external source, etc. That state can be changed with commands, and it can be queried with commands. Some commands change multiple state parameters and some queries only return dynamic data that has been captured by the measurement engine. SCPI commands can be broadly categorized as follows:

Configuration Commands
These commands are “write-only” and are essentially a “preset” which completely sets up the instrument for a particular function, range, etc. Whatever state the instrument was in, that is now changed, and many state parameters may have changed. Once used, all you typically need is to trigger the instrument and read results back. Here are two examples that show parameters and no parameters. Manuals that describe these commands often have tables to show all the state changes of the instrument.

CONF:VOLT:DC 5.00, 0.1
CONF:ACPower
TRIG IMMEDIATE

Configure – Query Commands
These commands are single-state operations where only a single state variable is

changed and can be queried. For example, you can change the range of the instrument, and you can read it back. It is a command that looks the same for configure or query, except for the addition of a "?" at the end of the text to signify a query.

SENSE:VOLT:NPLC 1
SENSE:VOLT:NPLC?
SENSE:VOLT:RANGE 10
SEND:VOLT:RANGE?
TRIG:COUNT 10
TRIG:COUNT?

The query form of the previous command actually reads the state variable for that configuration of the instrument. The previous examples showed the CONF command, where many state variables are affected. It may take a number of queries to determine what changed in the instrument to determine what the CONF command changed.

Query Only Commands
These commands tend to be related to reading data from a buffer or actually causing measurements to be made. Other commands set up the instrument for these to work properly. These are the final commands you use in acquiring the data you want from the instrument.

FETCH?
READ?
MEAS:VOLT:DC?

Common Commands
These are the required commands in SCPI, according to the IEEE 488.2 standard. Since they are common to all instruments, they tend to do common things such as read commonly defined status registers, clearing error conditions, clearing reading buffers, reset, instrument model number, stored states, etc. Since these are common to all SCPI compliant instrument, once you learn them on one instrument, you know them for all.

*CLS
*IDN?
*RST

Reading the Instrument State Variables
As stated earlier, the instrument is a collection of state variables representing the state of the instrument. If you press buttons on the front panel, you are changing the state of the instrument as surely as you will when sending commands. And, this is the key to learning the instrument's SCPI – you make changes and see how the state of the instrument is affected. You use the query forms of commands to determine the state variable changes.

A properly designed instrument will have a configure command to change every

state variable and have an associated query of that state variable. This is the Configure-Query combination of SCPI commands. These are the commands we use to learn the SCPI of the instrument. You need to acquire all the SCPI commands of the instrument and separate these commands from all the other commands. This can be done quite easily for modern day instruments which have electronic manuals. Every well written manual or on-line help tool such as 34410A_SCPI_Reference.chm, provides a listing of all the SCPI commands of the instrument. You simply weed out all the commands that are not of this form. The following is an excerpt from an electronic manual to illustrate:

[SENSe:]RESistance:APERture {<seconds>|MIN|MAX|DEF}
[SENSe:]RESistance:APERture? [{MIN|MAX}]
[SENSe:]RESistance:APERture:ENABled?
[SENSe:]RESistance:NPLC {<PLCs>|MIN|MAX|DEF}
[SENSe:]RESistance:NPLC? [{MIN|MAX}]
[SENSe:]RESistance:NULL[:STATe] {ON|OFF}
[SENSe:]RESistance:NULL[:STATe]?
[SENSe:]RESistance:NULL:VALue {<value>|MIN|MAX}
[SENSe:]RESistance:NULL:VALue? [{MIN|MAX}]
You will note there are 4 Configure-Query pairs and one that is a query only. In this case, we throw out the singleton, since it is returning a state variable that is not directly controlled by a command. This singleton is valid only if you use one of the instrument's APER configure commands. We also throw out the non-query versions of the 4 pairs, and we eliminate all the parameters. Finally, we eliminate all optional SCPI command nodes that are denoted by the brackets "[]". Much of this can be done with text editor Replace and Replace ALL functions. There is no need to convert from long form SCPI to short form. The commands are valid either way. Here is the result:

RESistance:APERture?
RESistance:NPLC?
RESistance:NULL:STATe?
RESistance:VALue?

This may seem tedious, but I have taken instruments like the Agilent 34410A DMM or Agilent 33220A Function Generator and extracted all the valid queries in 15-30 minutes each. Once you have all these queries, you are ready to use the SCPI Learning Process.

SCPI Learning Process
To learn the instrument, you simply read all the instrument state variables, change the state of the instrument, and read all the state variables again. That's it! Those weird CONF commands can be executed in between, and you can see how it changed the instrument. Those 15 button pushes on the front panel changed state variables, and you can see the SCPI command changes. It really does not matter how you change the instrument, as long as you first capture the state, make the changes, and then capture the state again, you have the means for learning what happens. Here is the process in a sequence:

1. Bring the instrument to some initial state - typically Reset or power ON
2. Send every extracted query to the instrument and read the results from the queries
3. Change the state of the instrument – front panel, driver call, Web interface
4. Send every extracted query to instrument again
5. Compare the results and show which commands changed parameters

With sometimes 100 or more queries, you probably want this automated somehow. With a tool like Excel, it is pretty easy to run through all the queries and dump them into columns that can later be compared. If any two columns are different, you note those changes, and those are the state variables you need to change to get the instrument to that new state. With the Command-Query commands as the basis, you now have the new state variable result, and you know the original command (without the "?"). All you need to do is send that command with that result concatenated to it to change the instrument to that state.

For learning how to program the instrument, this technique is perfect. For using the commands to reprogram the instrument directly, you need to be aware of a peculiarity of certain instruments. Some instruments like commands in a particular order; otherwise, you can generate errors or warnings. You learn this quickly by sending the commands back to the instrument. It really is a function of the order in which you read the results, build the return command-parameter, and send the new command back to the instrument.

SCPI Learning Utility
The entire process above is illustrated in the free utility available on the Agilent web site at the following URL: www.agilent.com/find/learnSCPI [2]

These sequences follow the SCPI Learning Process discussed earlier. All you need do is determine the instruments VISA string by using the Agilent Connection Expert or NI's Measurement Automation Explorer and place that in the cell with sequence #1. You then press the Reset Instrument and Read Instrument State. After changes are made to the instrument, you press the Read Instrument State again, and then press the Generate SCPI button to see the SCPI commands and parameters required to move from one state to another. After seeing how things change and all the changes required, you can refer to the instrument's SCPI Reference manual to better understand the specific command.

You can see the instrument's Query commands in the lower left hand corner, under Start. You need only supply all the valid Configure-Query commands in this column, and the tool does the rest of the work.

Conclusion
Learning how to program an instrument does not need to be experimental. If you

can make the product work from the Web Page or Front Panel, you can learn the SCPI commands necessary to program the instrument. The SCPI Learning process teaches you how the instrument moves from state to state by illustrating what commands are needed. After which, you study the particular commands to understand why these commands cause the instrument state to change.

Sometimes, you are only interested in a certain subsystem of the instrument. In that case, you need only provide the Configure-Query commands for that subsystem. The free Excel SCPI Learning Tool provides an easy way for you to send the particular commands for understanding the instrument. You only task is to find all the Configure-Query command pairs and extract the Query form. You do not need to understand the commands first. You just need to locate the commands.

Locomotive Manufacturers Case Study

New System Status Module for Locomotive Control Consoles

www.eao.com [3]

PROBLEM:
Locomotives for passenger, freight, and switching duty are complex electromechanical systems. A typical locomotive generates information on the availability and status of a variety of critical subsystems. How control cab personnel interact with and respond to these messages affects the efficiency, safety, and reliability of locomotives, with potentially significant impacts on uptime availability and operating costs.

Conventional system status modules include up to six illuminated indicators per unit that display "ready" or "warning" states for various critical subsystems. There can be five to seven modules in an operator control cab. As part of the locomotive start-up routine, each module must be tested independently by the cab crew, a time-consuming process that slows ready time and may increase operator error. Replacing modules often is a maintenance headache that involves costly removal and lamp replacement. Illumination in most status modules uses power-hungry incandescent lamps, with relatively short service lives of approximately 5,000 – 10,000 hours, that are susceptible to premature failure from shock, vibration, and power fluctuations. Incandescent lamps are associated with reduced reliability and decreased energy efficiency in the challenging environment of an operating locomotive.

SOLUTION:
To operate trains more efficiently and contain skyrocketing operational costs, railroad managers are making use of monitoring solutions that increase productivity. New locomotives cost $1.75 million or more and have a life expectancy of about 30 years, while maintenance costs can range from $35,000 to as high as $100,000 a year. Normal maintenance costs plus repair or replacement of failed

parts may exceed the initial cost of a locomotive over its lifetime. These clearly are investments that require constant attention to achieve optimum performance and service life.

To meet safety and performance objectives, crews need to be assured that locomotives are in proper working order before leaving the yard. One clear contributor to assuring locomotive performance and safety is the attention the industry has given to human factors in the design and evaluation of new and existing locomotive cabs. The need for improvement has been closely examined by the Federal Railroad Administration (FRA) in its assessment of standards developed by the Association of American Railroads on crashworthiness and working conditions that affect safety and productivity. The FRA's conclusion is that the two most important factors are working conditions and the incorporation of information technology.

Source URL (retrieved on *05/26/2013 - 1:20am*):
http://www.ecnmag.com/articles/2009/11/design-talk-automation?qt-recent_content=0

Links:
[1] http://www.scpiconsortium.org/
[2] http://www.agilent.com/find/learnSCPI
[3] http://www.eao.com/