XVII IMEKO World Congress
Metrology in the 3rd Millennium
June 22–27, 2003, Dubrovnik, Croatia

A MULTI-THREAD VIRTUAL INSTRUMENT FOR INTENSIVE SPECTRUM ANALYSER TRAINING

Matteo Bertocco, Sandro Cappellazzo, Claudio Narduzzi

Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Padova, Italy

Abstract - The scanning spectrum analyser is a particularly complex instrument, which requires careful understanding of its operating principles and a good deal of training to be employed properly. In this paper the development of a spectrum analyser virtual instrument, that can be offered as intensive self-study laboratory exercises, is presented.

Keywords: spectrum analyser, virtual instrument.

1. INTRODUCTION

The scanning spectrum analyser is a particularly complex instrument. Correct use and interpretation of measurement results require careful understanding of its operating principles and of the effects of a multitude of set-up options [1]. Therefore, a good deal of training is required to properly instruct an operator. The instrument is also comparatively expensive, so that educational laboratories, particularly in an academic environment, are usually hardly able to field more than a few units. As a consequence of these constraints, university engineering courses in instrumentation and measurement may often be limited to the presentation of fairly general concepts in spectral analysis and much restricted hands-on exercises.

The work presented in this paper is the result of an effort to address these problems by developing laboratory experiences on spectrum analysers, that can be offered as intensive self-study exercises. Students in an advanced instrumentation course are thus allowed to familiarise themselves with the intricacies of spectrum analyser operation, without suffering the time restrictions of a supervised laboratory exercise.

2. LABORATORY ENVIRONMENT

The creation of this particular virtual instrument application has been made possible by the availability of a well-developed software and networking environment for measurement laboratories [2], that has been developed as a long-term effort within the Department of Information Engineering of the Università di Padova. Its architecture is based on a general *laboratory server*, which provides the interface with virtual instrumentation applications and handles the exchange of information and commands with the actual measurement systems. This exchange takes place through the offices of a number of local *bench servers* which actually control the measuring instruments and forward the data they produce to the laboratory server by way of an intranet. At present, each bench server is a personal computer equipped with a LAN interface and an IEEE 488 interface board; controlled instruments are connected to the local IEEE 488 bus.

The laboratory server provides functions that are essential for the organisation of laboratory activities. It allows multi-user access through passwords, enables the system manager to assign users time-restricted permissions and, above all, supports time-shared use of measurement resources through a reservation scheme based on lock/unlock software primitives.

Any laboratory experiment is associated to a *bench*, which is identified by a symbolic name. A bench server provides access to a group of instruments, that may be arranged to form one or more benches, depending on the possibility to change the measurement configuration, e.g., by means of programmable switches. Each user accesses a single bench through a *client application*, the relevant request being sent to the laboratory server, that disciplines the access queue. In general, the PC or workstation where the client application is running will differ from the machine where the servers are located.

It is assumed that several users may want to perform the same experiment, thus requiring the same bench; therefore a time multiplexing approach has been adopted. When access to a bench is granted a suitable time slot is allowed, wherein resources are locked for exclusive use. This prevents conflicts among users trying to access the same instrument, with each one possibly attempting to set-up a different configuration. On the other hand, since proper instrument configuration may require some time, particularly for inexperienced users, a time-out limit is set whenever a bench is locked, to prevent a user from taking over resources forever. If a user is forced to unlock a bench, he is automatically placed in a waiting queue, so that he may eventually be able to regain control and finish his experiment.

Although this arrangement may sound complex, the whole process is almost transparent (apart from the occasional standby during an experiment), since client applications have been designed to account for the lock/unlock mechanism. As will be discussed in the next section, current set-up parameters of the instruments making up a bench are always stored as variables in the application

that makes use of that bench, so that the appropriate set-up is automatically restored when the bench is available again.

In view of the educational use of the instruments, care must be taken to avoid potentially dangerous situations. While the user should be given as much freedom as possible in setting up instruments, dangers or possible damages must be avoided. Accordingly, the laboratory server can be instructed to filter instrument commands issuing from the virtual instrument application by setting range limits, so that some values cannot actually be accepted.

The introduction of passwords associated to specific time restrictions means that students can be arranged in shifts, much in the same way as in a conventional laboratory. However, a much larger amount of time is potentially available, since the instruments can be left unattended while in operation and can be accessed through a virtual instrument application at any time of the day.

3. VIRTUAL INSTRUMENT FEATURES

Development of the spectrum analyser virtual instrument had two main aims: to provide a realistic, yet reasonably simple user interface and to efficiently manage the acquisition of traces from the actual instrument. In addition, the peculiar characteristics of the laboratory environment had to be dealt with. This resulted in a number of requirements:

- controls in the user interface should provide a sufficiently realistic rendition of a spectrum analyser front panel;
- the application must provide good interaction: even though the time needed to transfer data through a network must be accounted for, response times must be short;
- users must be able to perform their exercises without needing detailed knowledge of the laboratory environment;
- time-shared operation should be as transparent as possible to users but, to avoid exceedingly long waiting times, the maximum number of time-shared simultaneous users may have to be limited.

Fig. 1 shows the front panel of the spectrum analyser application, realised using the LabVIEW visual programming environment.

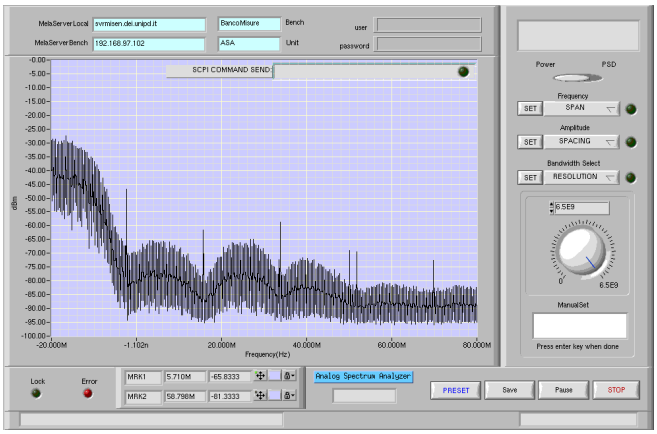

Fig. 1: LabVIEW virtual spectrum analyser front panel.

The panel has been divided in a number of functional areas. The area on top of the graphical display is dedicated to network and laboratory connection parameters; users are only asked to input the laboratory and bench server addresses, as well as the name of the experiment (*Bench*) they are interested in.

The lower part of the panel takes care of general operating conditions, with LEDs on the lower left corner indicating the user whether the bench is locked and working correctly.

The area at the right of the display houses instrument controls. Spectrum analyser front panels are often cluttered by a large number of buttons and knobs. This situation can be disorienting to newcomers; furthermore, for a training application aimed at the general use of the instrument, several controls are actually redundant. Therefore, it has been decided to simplify the panel as much as possible. As can be seen, only the two main controls, *Frequency* (which includes setting the frequency span) and *Amplitude*, have been retained.

In addition, *Bandwidth Select* has been made available as a primary control, allowing to change the spectrum analyser resolution and video bandwidths. Although this aspect is usually dealt with automatically in current instruments, it is quite significant from a teaching point of view and has been emphasised.

Conversely, the ability to change amplitude set-up parameters has been strongly restricted. It is well known that proper setting of the input attenuators is essential to optimise the performances of a spectrum analyser. However, trainees having free access to the attenuator set-up may inadvertently overload the mixer, with potentially disastrous results. For this reason, the instrument reference level is set within a bench configuration file for the specific experiment. The file resides on the laboratory server and cannot be altered by users. Therefore, only the vertical scale factor and the log/linear presentation can be chosen by the *Amplitude* control.

A drop-down menu is associated to each control, allowing the user to select the parameter he wishes to change. The parameter value can be input either directly from the keyboard, or by means of a virtual knob (or dial, in LabVIEW parlance). It should be noticed again that, to avoid overcrowding, a single knob has been used, requiring its implementation as a multifunction device.

Finally, it is possible to calibrate the virtual spectrum analyser vertical scale to read either power or power spectral density (the latter based on the indications of the noise marker in the actual instrument [3]). It is thus possible to employ the client application for the analysis of a variety of sources, including modern telecommunications signals.

4. MULTITHREAD IMPLEMENTATION

The LabVIEW client application has two main tasks. One is the collection of user inputs from the virtual front panel: these are turned into commands and forwarded to the spectrum analyser. The second task is the acquisition of the spectrum analyser trace data, that are sent to the front panel display. Implementation of the two activities is made more

complex by the need to ensure correct operation in a time-shared environment. In fact, it has to be remembered that:

– bench resources are locked for a finite time: the application may be suspended at any time if the timeout expires and an unlock is forced;
– acquisition of trace data takes place correctly only if the proper query/read command sequence is implemented.

Furthermore, whenever allocation of bench resources is lost, a new access must be automatically requested, without disrupting the operation of the client application. This process must be transparent to the user and should restore the original set-up of the bench, as well as of the virtual front panel.

As a result of these requirements, the decision was taken to implement the virtual instrument as a multi-thread software application, where three independent activities take place concurrently. The threads shown in Fig. 2 deal, respectively, with the acquisition and display of trace data, with front panel management and with the supervision of operations in the time-shared laboratory environment. It should be noted that the block diagram of Fig. 2 shows the main body of the program, which is actually part of a sequence of activities. These obviously start with the establishment of a network connection with the laboratory server and the setting up of the client application front panel. When the program is terminated, the final activity is the closure of the network connection.

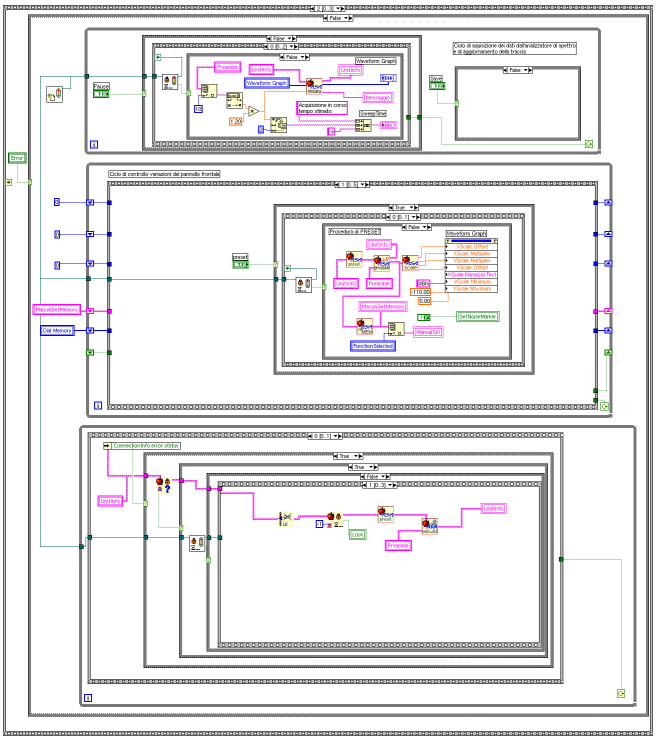

Fig. 2: multithread implementation of the LabVIEW client.

The advantage of using threads is that they allow the implementation of three independent and concurrent activities. It should be considered, however, that the threads share the communications channel between the virtual instrument application and the real instrument as a common resource. Therefore, the channel had to be protected by a mutually exclusive semaphore, that ensures activities cannot be interrupted at the wrong time. In this way the channel can be employed by one of the threads without interference from the others. This aspect is essential to guarantee that the exchange of commands and queries with the spectrum analyser takes place in an orderly fashion according to the relevant communication protocols.

In a training application, time requirements are generally not much restrictive. The trainee often needs plenty of time to figure out how best to set up the instrument and, at the same time, is willing to wait some length to see the results of his/her efforts. However, in a time-shared environment this is no longer true. While the single user may be working at a leisurely pace, the spectrum analyser is not: in fact, efficiency in the exchange of commands and data allows the highest possible number of time-multiplexed users on a single instrument. Therefore, particular care has been taken to optimise the use of the communications channel, ensuring at the same time that all activities within the virtual instrument are assigned a sufficient portion of the communications resources.

In this respect, careful management of the instrument status can help reduce network traffic. For this purpose, extensive status information is kept within the client application. At each iteration of the thread that manages the front panel, parameter values stored in the status memory are compared with the new inputs. To avoid useless repetitions, a new command is forwarded to the spectrum analyser only when the required set-up differs from the current status and, then, only the specific parameter that the user wishes to change is dealt with.

On the other hand, modern spectrum analysers are well known to have several linked parameters: when, for instance, the frequency span is changed, sweep time often changes automatically as well. Therefore, in the implementation of the client application, some care is necessary to keep the status memory synchronised with the actual instrument status. This is achieved by requesting the whole status information of the instrument whenever a new command is sent to the spectrum analyser, so that the memory is regularly updated. In this way it is also possible to deal easily with the very common case when a user-provided numerical parameter is approximated by the instrument to the nearest acceptable value. Furthermore, in time-multiplexed operation a set-up can be restored automatically, each time the instrument is available. This just requires reading the status memory and sending commands that set all the instrument status variables accordingly. Thus, users need not be forced to waste part of the allotted time-out interval, merely to recover the appropriate instrument configuration.

As far as data acquisition is concerned, the relevant thread simply reads trace data from the spectrum analyser. To reduce the acquisition time, data are accepted in a raw format, all conversions being carried out elsewhere. It should only be noticed that acquisition from a spectrum analyser may take a considerable length when long sweep times are selected. This should be taken into account when planning for time-shared operation, since this one factor may severely reduce the number of simultaneous users. At present, a limit of 20 s has been set. It should also be

considered that no data are available until a sweep has been finished, therefore the data acquisition thread within the client introduces a wait time equal to the spectrum analyser sweep time before querying the instrument for data.

4. CONCLUSIONS

Development of the whole application with LabVIEW was a rather demanding task, corresponding to several weeks of work. The process was made faster by the availability of a purpose developed user library, that provided all the functions necessary for interfacing to the laboratory server.

Although in a visual programming environment program development may appear simple, the block diagram of a complex application soon becomes unwieldy, unless suitable programming criteria are adopted. With LabVIEW, the main rule is to partition the programming tasks into subVI's that can be nested within the main program to help dominate complexity. The approach was used throughout this project, and resulted in the inclusion of over 50 subVI's in the main program. This brought the additional advantage that several subVI's are general purpose, allowing the progressive build-up of another user library, that will eventually speed-up the development of further client applications.

Preliminary tests have been completed, proving that the application fulfils the requirements. Connection to the spectrum analyser exercise has been tested both by local area network and by dial-up connection. Furthermore, a presentation has been given, where the client application resided in a portable computer, some 300 km from the laboratory [4]. In all cases, response times have been found to be quite acceptable, particularly in view of the comparatively long spectrum analyser scan times. The instrument currently employed on the spectrum analyser bench allows a full span sweep up to 6.5 GHz, in which case the update rate on the virtual instrument front panel is around 30 frames per second for a LAN connection. With a single connected user, the speed difference between the spectrum analyser screen and the front panel diagram is hardly noticeable.

These results suggest that up to four users could be simultaneously connected to the same instrument with completely independent measurement set-ups, leading to a considerable increase in instrument availability.

REFERENCES

[1] "Application Note 150 - Spectrum Analysis Basics", *Hewlett-Packard*, 1989.

[2] L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, V. Piuri, *A Web-Based Distributed Virtual Educational Laboratory*, IEEE Transactions on Instrumentation and Measurement, vol. 49, no. 2, pp. 349-356, April 2000.

[3] "Application Note 1303 - Spectrum Analyzer Measurements and Noise", *Agilent Technologies*, 1998.

[4] C. Narduzzi, round table presentation, XIX Congresso Nazionale Misure Elettriche ed Elettroniche, Parma, Italy, 9-11 September 2002.

Authors:

Prof. Matteo Bertocco, Dipartimento di Ingegneria dell'Informazione, Università di Padova, via G. Gradenigo, 6/b, I-35131 Padova, Italy. Phone: +39 049 827 7627. Fax: +39 049 827 7699. e-mail: matteo.bertocco@unipd.it

Ing. Sandro Cappellazzo, Ph.D., Dipartimento di Ingegneria dell'Informazione, Università di Padova, via G. Gradenigo, 6/b, I-35131 Padova, Italy. Phone: +39 049 827 7760. Fax: +39 049 827 7699. e-mail: frenco@dei.unipd.it

Prof. Claudio Narduzzi, Dipartimento di Ingegneria dell'Informazione, Università di Padova, via G. Gradenigo, 6/b, I-35131 Padova, Italy. Phone: +39 049 827 7649. Fax: +39 049 827 7699. e-mail: claudio.narduzzi@unipd.it