Virtual Remote: The Centralized Expert

Remote operation of bit error rate test sets using an X Windows based "virtual instrument" allows network operators to monitor remote sites from a central office. The extensive use of a common firmware development platform allowed the fast-track development of virtual remote software and rapid integration into all instruments built using the platform.

by Hamish Butler

In today's competitive marketplace, network operators must provide their customers with a cost-effective and efficient service. Large corporate customers expect the network operator to provide a guaranteed quality of service at the best possible price. This means that the network operator has to run the network as efficiently as possible and as cheaply as possible. Let's look at the issues of efficiency and cost separately.

First, the network must be efficient. That is, it must reliably transport customer data on demand. It must have a low error rate and high uptime. If data is lost in transit or if customers are unable to send data because the network is down they are likely to switch to another carrier. It is no longer acceptable for the network operator to wait for faults to be reported and then to fix them; a preventive maintenance approach is required. Preventive maintenance involves monitoring the performance of the network links and looking for any degradation in performance. If the performance degrades beyond a certain level the customer traffic must be switched to another data link with the required level of performance. The defective link must then be fixed as quickly as possible.

Network operators must be able to provide a competitively priced service to their customers and still be able to run the network and make a profit. This places heavy demands on network test and maintenance:

- Test and maintenance departments are under pressure to maintain the required level of service using fewer staff.
- Expenses are under tight control. Traveling between sites is expensive and unproductive.
- Network capacity is at a premium. The amount of spare capacity that can be held in reserve is limited. If a data link has errors and traffic is switched to a spare link it is probable that the errored link must be fixed as soon as possible so that it can then provide the spare capacity.

Traditional approaches have used comparatively low-skilled technicians to provide the first level of troubleshooting and maintenance. These technicians may have been based at remote sites or based at a central site and dispatched to remote sites as required. If these technicians were unable to diagnose a fault a more skilled technician was sent to the remote site. This process is time-consuming, expensive, and prone to error. It is not unknown for a technician to travel to a remote site only to find that the fault lies somewhere else. Sometimes the unskilled technician has even introduced faults into the system while attempting to diagnose the original fault.

The ability to perform testing of remote sites from a central office has many advantages. Skilled technicians can be concentrated at one site. Less time is spent traveling. Fault locations can be diagnosed before dispatching technicians to fix the faults. Preventive maintenance is improved through the ability to monitor many remote sites from one place.

The traditional approach to such centralized testing has been to locate portable test equipment at remote sites and to communicate with these instruments using an RS-232 link and modems. Using this method it was normal to interrogate the instrument by writing a controller program for a PC or workstation. This method is restricted because of the limited amount of information that is returned to the operator. The operator is only given the information that the programmer requested when the program was written. For example, if the program is monitoring one or more selected results and something that is not being monitored by the program changes, there will be no immediate feedback.

It is sometimes necessary for the skilled technician at the central site to give instructions to an unskilled technician at the remote site. Using the above method the skilled technician has no direct feedback of the tasks performed by the unskilled technician.

The Virtual Instrument

Engineers at the HP Queensferry Telecommunications Operation have been developing instrument firmware using an X Windows instrument simulation for several years. The engineers responsible for the instrument simulator had often thought how useful it would be if the simulator could be used to control an actual instrument.

In the second half of 1991 an HP field engineer started working with a large network operator on a contract for HP telecommunications test sets and computers. This customer was setting up a large-scale centralized test and maintenance system to monitor the network. Working with the customer the HP field engineer developed the idea of the virtual instrument.

Fig. 1. An image of two instruments in a "dark office" superimposed on an image of an operator at a workstation using virtual remote software to control the two instruments. Representations of the instruments' front panel are displayed on the screen.

This was the vision presented to the customer: From a single central office, customer personnel would be able to use an HP workstation to bring up an accurate simulation of a remote instrument on the display. They would be able to display several instruments simultaneously. Fig. 1 illustrates this concept. The photograph shows an operator in a central office using virtual remote to control two instruments in a distant "dark office." Each virtual instrument would be operated by using the mouse to press keys. These key presses would be relayed to the remote instrument. As the remote instrument updated any of its feedback mechanisms—display, LEDs, or audio—the instrument simulation would relay these changes to the operator in the central office.

The important task now was to see how R&D could best implement the virtual instrument application. The development of the product was split into two stages. The first was to take the current instrument simulator and use it to produce a prototype virtual instrument application. The second stage was to take the prototype application and turn it into a polished software product that would meet the rigorous demands of HP customers.

Product Description

The product that emerged from this effort is the HP 15800A virtual remote capability software. It runs on HP 9000 Series 300, 400, or 700 workstations under the HP-UX* 8.0 operating system.† The software provides for the centralized supervision, operation, and collection of results from remotely located HP 377xxA Option V01 telecommunications test sets and analyzers. Option V01 virtual remote capability enhances the instrument firmware to respond to the HP 15800A software.

A single workstation can control up to twelve remote test sets. The display shows windows identical to the front panels and screens of the test sets. When a technician is controlling a remote test set manually, all actions and results can be monitored at the workstation.

At present, Option V01 is available for the following test sets:

- HP 37701B T1 tester
- HP 37702A digital data tester
- HP 37704A SONET test set
- HP 37714A PDH/SDH test set
- HP 37717A PDH/SDH test set
- HP 37721A digital transmission analyzer
- HP 37722A digital telecomm analyzer
- HP 37724A PDH/SDH test set
- HP 37732A telecomm/datacomm analyzer.

Option V01 can be retrofitted to existing instruments by changing ROMs.

Virtual Remote Design Concept

As has already been explained, instruments at the Queensferry Telecommunications Operation are developed using an instrument simulator. This instrument simulator is in fact part of a larger common firmware platform.[1] This common firmware platform incorporates compiler-based code generation and simulation tools along with the source code for a core generic instrument. The design of virtual remote is so closely tied to the instrument simulator that a brief explanation is required.

The core of Queensferry Telecommunications Operation's common firmware platform is a compilation and simulation tool known as ISS—Instrument Software System. ISS is based on an abstract, high-level, instrument definition language. This language is used to define many aspects of the instrument operation:

- The instrument user interface
- All instrument control variables
- Interaction and dependencies between control variables
- Instrument display data—text and graphics
- Instrument results
- Instrument data input—hard and soft keys
- Relationships between data input (keys), control variables, and data output (display data)
- Printer output data
- Remote control command definitions.

The instrument firmware development process is as follows:

- Produce the instrument definition using the ISS language
- Simulate the instrument operation using the ISS simulator
- Use the ISS compiler to generate an embedded instrument database
- Compile the instrument database into embedded instrument code
- Test firmware operation in the target instrument.

The ISS simulator and the target instrument share the core components, mainly the ISS database processing engine. The development process is illustrated in Fig. 2.

As shown in Fig. 2, the ISS database and the ISS engine are common to both the ISS simulator and the actual instrument. The instrument operator interacts with the instrument using keys on its front panel and data displayed on its screen. On the workstation, the key input and screen output are handled by a set of functions written in the C programming language. These functions are implemented using X Windows function calls.

In the instrument the screen output and key input are handled by an alternative implementation of the same C functions. In this case the functions interact directly with the instrument's display control and keyboard input hardware.

The basic instrument architecture is shown in Fig. 3.

† A PC version, HP 15801A, is now available as well.

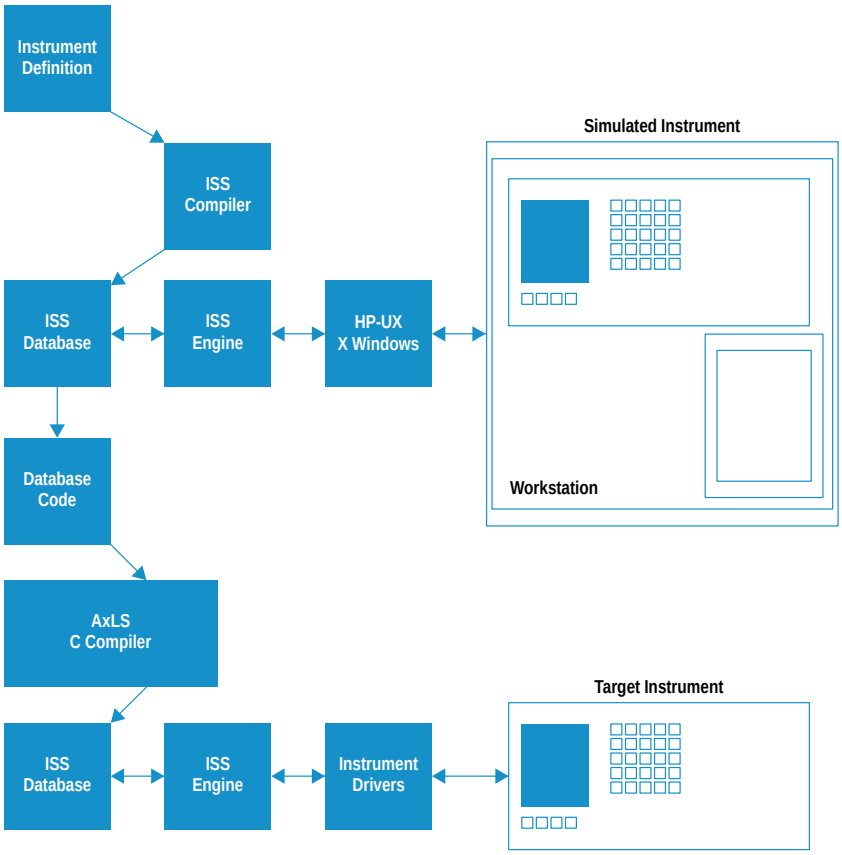

Fig. 2. Instrument development using the ISS compiler and simulator. The ISS database and the ISS engine are common to both the simulator and the actual instrument.

The key to the development of virtual remote was to connect the ISS engine embedded in the instrument with the screen output and key input functions used on the workstation. The use of the same set of display functions in both the workstation and the embedded system gave the potential to update both displays at the same time. We also wanted to duplicate the function calls made by the instrument. At the same time as the function call is performed in the instrument it must also be performed on the workstation. This technique is known as a remote procedure call. A UNIX® application

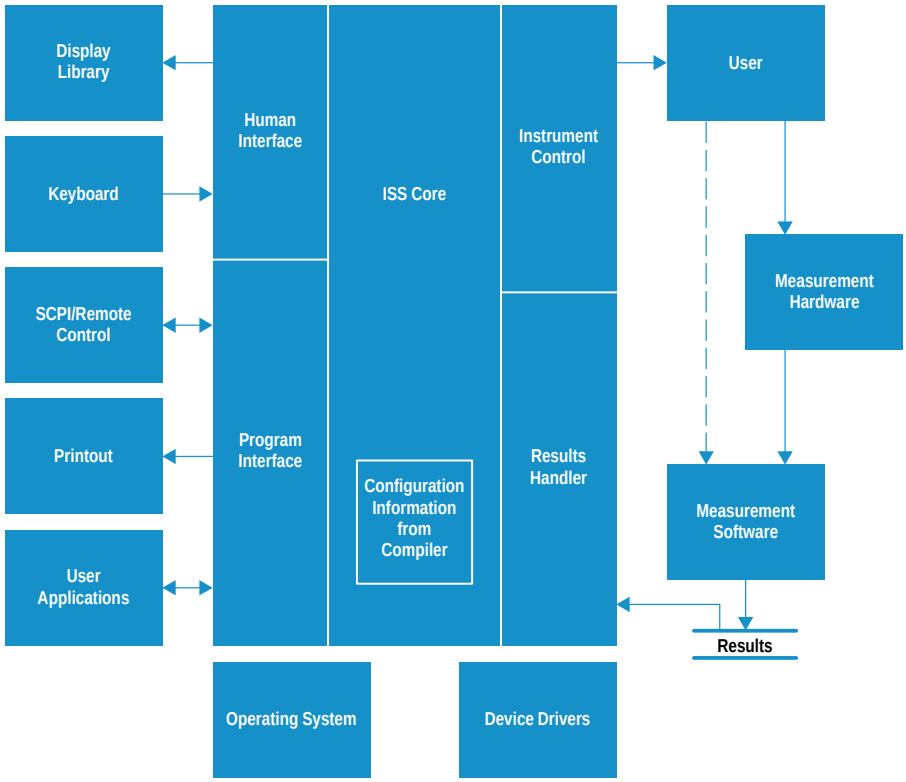

Fig. 3. Basic instrument architecture.

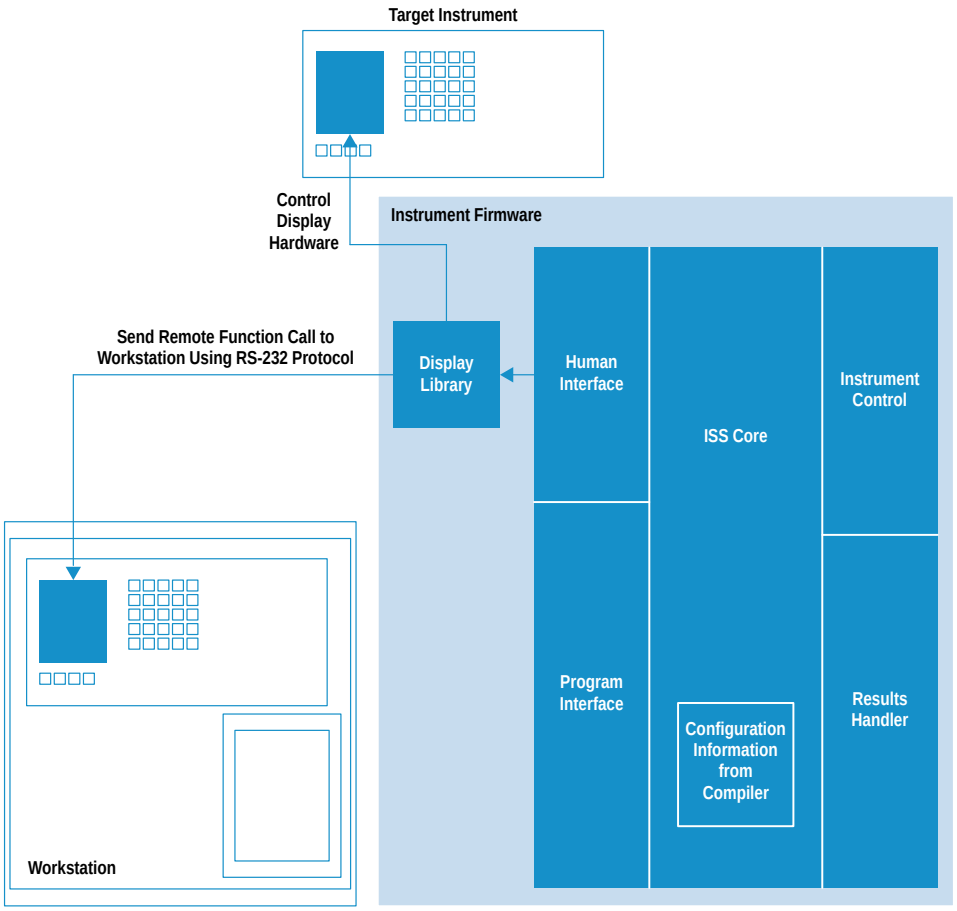

Fig. 4. Instrument-to-virtual-display connection.

that needs to make remote procedure calls to another machine on the network would use Sun Microsystems' RPC. In our instrument environment we could not support this. It was decided to develop a protocol of our own between the instrument and the workstation that could be used to initiate remote display functions. This protocol would use ASCII data transmitted between the instrument and the workstation on a 9600-baud RS-232 data link. This is illustrated in Fig. 4.

The ISS database engine is event-driven. The keyboard process in the instrument waits for a key to be pressed and then acts upon that key. In the case of the display we had to output to two devices at once—the simulated instrument and the target instrument. In the case of the keyboard we had to accept input data from two sources—a local keyboard and a remote keyboard. This is accomplished through the use of our dedicated protocol. Keys pressed locally are processed as they always were. A key pressed on the workstation is encoded and transmitted over the RS-232 link to the instrument. When the data is received by the instrument it is decoded and the key code is placed in the key input queue. This is shown in Fig. 5.

It would have been possible to use the above techniques to develop an application that ran on the workstation and was used to communicate with a remote instrument. This application would have to have embedded knowledge about the instrument it communicated with so that it could render an accurate representation of the instrument on the workstation screen and have knowledge of the keys present on the instrument front panel and the key codes assigned to each one.

Because our division manufactures many different instruments, this solution would require a separate virtual remote application for each instrument, which is unsatisfactory for the following reasons:

- Developing multiple applications would require extra engineering effort.
- Extra administration effort would be required for sales and support.
- Customers often use several different instruments. Some applications require different instruments used in combination. It would be unfriendly and expensive for our customers to have to buy and use a separate application program for each different instrument.
- This approach may have presented problems with new instruments or instrument enhancements released after the application code was produced.

The ideal solution is to have a single virtual remote application program that can be used with any instrument that is already shipping or that may ship in the future. To do this the virtual remote application has to be completely generic, sourcing all of its instrument-specific data from the instrument itself.

Once again, the use of the ISS simulator for instrument development helped us to find the solution we required. The ISS simulator already displayed a representation of the instrument on the workstation and had knowledge of the instrument key positions and key codes. This data is contained in the instrument definition coded using the ISS language. In the simulator this data is compiled into RAM data structures

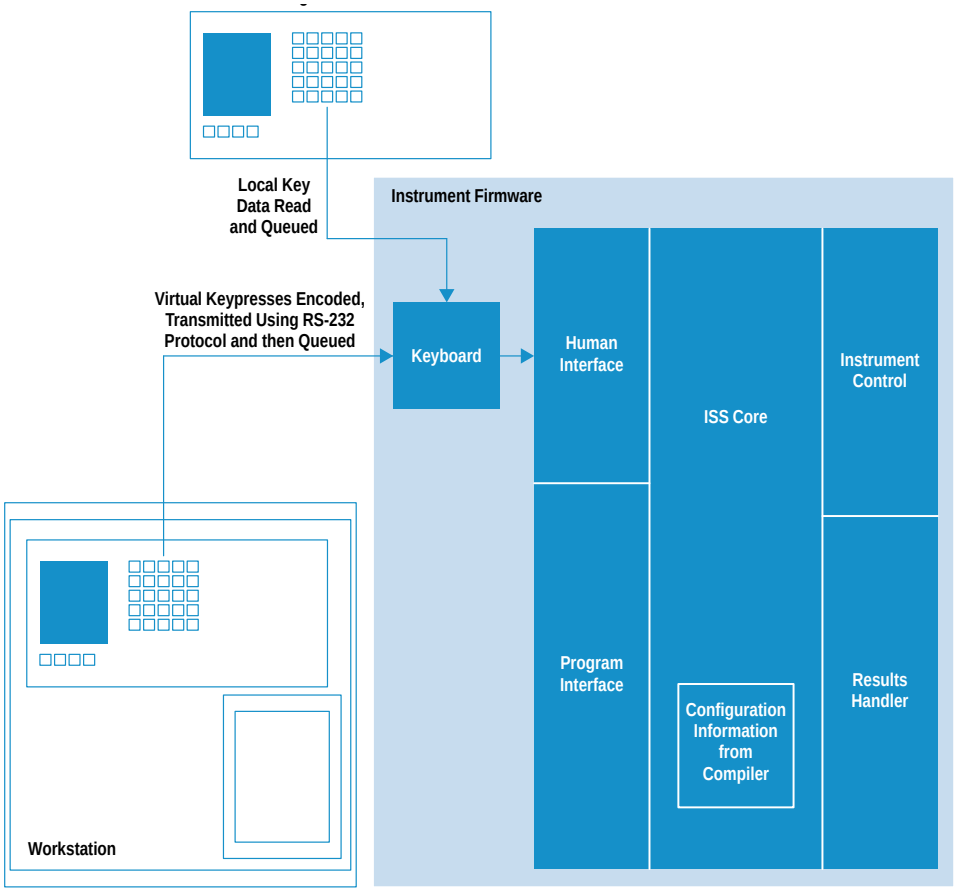

Fig. 5. Virtual-keyboard-to-instrument-keyboard process.

which are then used to render the instrument and process the keys.

The development of virtual remote required these data structures to be added to the instrument database embedded into the instrument. Once this was done it was an easy step to enhance the instrument-to-workstation protocol to allow the data held in ROM in the instrument to be sent over the RS-232 data link to the virtual remote application in the workstation. The virtual remote application then holds the data in a RAM data structure and uses it to render the instrument and process key presses.

Fig. 6 shows a workstation screen displaying the virtual remote representation of an instrument being controlled. All of the data needed to draw this image is stored within the instrument and is sent to the virtual remote application upon request. This allows the virtual remote software to control many different instruments without any prior knowledge of how they look or how they operate.

Virtual Remote Development

Using the above design concepts it was relatively easy to produce a working virtual remote prototype. The prototype application was extremely useful and very successful. It proved that the design concept was feasible, it illustrated the concept of the virtual instrument, it allowed the instrument-to-workstation protocol to be tested, it helped prove the viability of the product, and it gave a base upon which the final product would be built.

Once the prototype was complete the next stage of the project was to take the prototype and turn it into an application suitable for use by customers. This application had to implement features not supported in the initial prototype:

- Multiple simultaneous connections
- Connection arbitration and verification
- Connection to and dialing of remote instruments using modems
- Socket connections to remote instruments over a LAN
- Industry-standard OSF/Motif graphical user interface.

The first change needed to convert the prototype into a product was to convert the application to use an OSF/Motif™ user interface style. This was done using HP's Interface Architect (also known as UIM/X) user interface management system. The prototype virtual remote had relied entirely on the display library written at the Queensferry Telecommunications Operation for all X display functions. This display library relies on low-level Xlib functions. This approach was adequate for the simulation of the instrument display and front panel. The implementation of application-specific display widgets such as menus and error dialogs was, however, somewhat nonstandard. Interface Architect was used to provide the top-level OSF/Motif application widgets and error dialogs. The low-level display functions using the shared display library were then embedded inside the OSF/Motif application.

The next stage in the development was to split the single-threaded application into constituent parts that could be used to create a multiprocess application capable of supporting the requirements of the final system. The design used is shown in Fig. 7.

Fig. 6. A workstation screen displaying a picture of an instrument drawn by the virtual remote application. All of the data used to draw this image is stored in the instrument and is sent to the virtual remote application when requested. This allows virtual remote to be used with many different instruments without having any special knowledge about how they look or operate.

This design has several advantages. The independent server process is the core of the system. This process is responsible for the arbitration and initiation of all connection requests. If all connections were made using TCP/IP a single server would be responsible for all instruments connected to the network. When using RS-232 for direct or modem connections a server process is required for each workstation. This process is responsible for instruments connected through the RS-232 ports of the workstation upon which it is running. Each server process accepts connection requests from any front-end process. The front-end process may be running on the same host as the server or on any other networked workstation. The separation of the front end from the server gives us the ability to make connections across the network to instruments physically connected to remote workstations. A second advantage of separating the front end from the server is that it allows the provision of more than one front end. At present only two front ends have been provided: an OSF/Motif-based menu interface and a command-line interface. It would be relatively easy to design a new front end, perhaps for integration with some larger system.

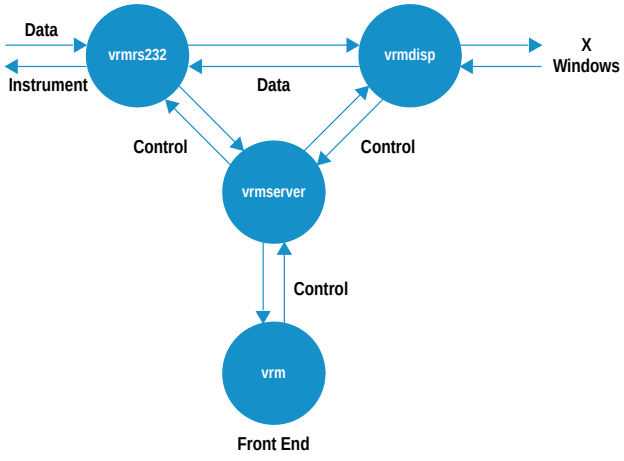

Fig. 7. Virtual remote software architecture.

Once the front end has verified that no one is connected to a particular instrument, it attempts to establish a connection to the instrument. Here again, the communications process has been decoupled from the server. The server must determine the communications channel used for connecting to the instrument. This data is stored in the instrument configuration file. The server then uses this data to start the appropriate communications driver process. The communications drivers provided at present are RS-232 direct connect, RS-232 modem, and TCP/IP. Queensferry Telecommunications Operation instruments have traditionally supported RS-232 and HP-IB (IEEE 488, IEC 625) communications ports. The TCP/IP method of connection is provided to allow networked communications and to allow more simultaneous connections than would be possible using RS-232 ports provided on current workstations. This method operates by using TCP/IP to connect the workstation over the network to a terminal multiplexer. This device maps TCP/IP addresses to RS-232 ports.

The latest generation of instruments produced by the Queensferry Telecommunications Operation can have a ThinLan/Ethertwist interface that allows TCP/IP communication directly to the instrument. This removes the need for the terminal multiplexer and allows any workstation on the network to connect to any instrument.

New communications channels can be provided simply by writing a new driver process.

Once the server has started the communications process and established that the instrument is responding, it starts up the virtual remote display process. This process is the part of the

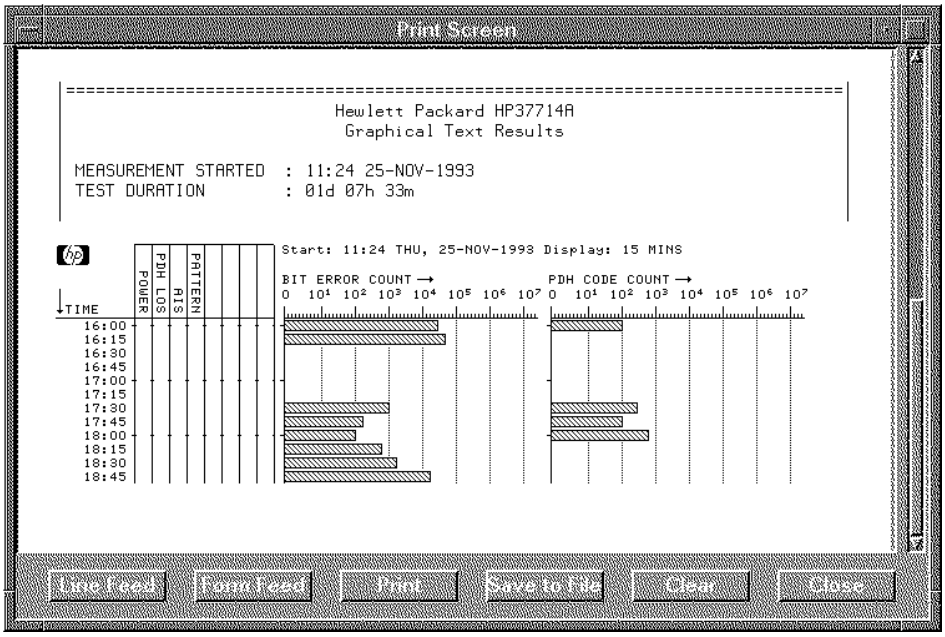

Fig. 8. This is a screen of the virtual printer application. The virtual printer displays print data that the instrument would normally send to a real printer. During virtual remote operation the data is sent to the virtual remote application and is displayed by the virtual printer. The pushbuttons on the virtual printer can be used to send the buffered printer data to a real printer connected to the workstation, save the data to a file, clear the buffer, and so on.

application most closely related to the lab prototype. This process interrogates the instrument for its X display data and uses this data to draw the image of the instrument. It then processes key presses, encodes them, and sends them to the instrument. Data received from the instrument is decoded and the appropriate action taken. This may be the drawing of characters on the screen, illuminating LEDs, or simulating the behavior of an HP ThinkJet printer.

Printer Simulation

The test instruments designed at the Queensferry Telecommunications Operation allow a rich set of data to be logged to an external printer under a variety of operator-defined conditions. An external printer is not of much use if the instrument to which it is connected is not on the same site as the operator using the instrument. It was therefore decided to provide a virtual printer along with the virtual instrument.

Once again this development was facilitated by the ISS development platform. The ISS language/compiler/simulator provides facilities for the instrument designer to compose logging format definitions, which are formatted in the instrument as required. To allow testing of the formats for correctness, an X Windows printer simulator had been developed and integrated with the ISS simulator. The simulator could take the compiled format definitions and simulate their output to an HP ThinkJet printer.

To provide the virtual printer capability to the virtual remote application, all that was required was to integrate the X Windows printer application with the virtual remote application and provide a means to multiplex the printer data with other virtual remote data transmitted between the workstation and the instrument.

The integration of the X Windows printer application with virtual remote was a straightforward task. To maintain the overall look and feel of the application the low-level Xlib code used for the printer application was encapsulated within an OSF/Motif user interface. The separation of the printer data from the standard virtual remote protocol was accomplished by extending the protocol to provide a means of encoding the printer data. This was necessary to prevent raw printer data from clashing with the virtual remote protocol. The instrument logging output driver can detect when the instrument is in the virtual mode of operation and will then output the printer data in encoded rather than raw form. Mutual exclusion is used to prevent the display process and the printer output process from attempting to send data to the workstation at the same time. Using this technique data throughput can be maintained by both processes in a controlled manner.

The virtual printer receives the encoded printer data and decodes it. This data is then displayed in the X Windows HP ThinkJet printer simulation. The simulator can display both text and graphics. At the same time as the printer data is displayed it is appended to a data buffer. This buffered data can be saved to a named file or sent to a printer spooler by the operator. The online buffer can be cleared on demand.

Fig. 8 shows the virtual printer application. The main portion of the application is the "virtual paper," where the text and graphics of the printer data are displayed. A scroll bar along the side of the screen can be used to scroll the paper backwards and forwards. Pushbuttons along the bottom of the screen are used to control the functions described above.

Instrument Customization

In addition to the development of the workstation application code the instrument firmware must also be modified for virtual remote operation. The modifications required affect the following areas:

- Instrument communications drivers (RS-232 interrupt service routine)
- Instrument display library code
- Instrument keyboard processing code
- Instrument external interface selection
- Instrument simulation data.

The first three areas above are contained in the common code distributed to all projects using the Queensferry Telecommunications Operation common platform. The remaining areas are instrument-specific and must be done on an instrument-by-instrument basis.

First, the instrument external interface selection code must be enhanced slightly. A new external connection mode, virtual remote, must be added alongside the existing selections such as printer and remote control. This is a very minor change to the instrument ISS definition file. Two new C functions must be provided. These functions are called by the common code when a virtual connection is being established. These functions are used to make one or two minor configuration changes to the instrument. They cannot be common since they relate to variables defined in individual instrument ISS definition files.

The second instrument customization is the instrument simulation data. In the past the simulation data has only been used internally for instrument simulation during development. In this application it is the simulation of the instrument operator-machine interface that is important, not the quality of the displayed instrument image. The virtual remote application, on the other hand, uses an image of the instrument to make remote operation by a customer easy. In this case the quality of the image displayed is very important. The instrument designer must ensure that the instrument simulation data is as accurate as possible. The areas of concern are instrument dimensions, key positions, text labels, and connector positions and accuracy. Using the syntax of the ISS simulator it is possible to construct a fairly accurate representation of the instrument.

The use of the Queensferry Telecommunications Operation common firmware platform has enabled the virtual remote firmware to be retrofitted in a complete family of instruments with only a small amount of effort. All new instruments developed using the platform will build in virtual remote support from the start.

Summary

Since its release, virtual remote has proved to be a very popular product. A broad spectrum of customers have put virtual remote to work, sometimes in interesting applications:

- A major network operator has installed virtual remote as part of their network monitoring and maintenance system.
- Virtual remote has been used in the installation of cable television networks, allowing centralized testing of the network.
- Virtual remote has been used to test satellite communications channels, allowing simultaneous monitoring of the data links at geographically separate ground stations from a central control center.

The success of virtual remote for HP-UX workstations led to a demand for virtual remote on PCs. HP 15801A PC virtual remote was developed so that as far as possible the source code is shared with the HP 15800A HP-UX product.

Fig. 9 shows an operator using virtual remote on a PC in an office environment. The instrument being controlled is at a remote site and the connection is through a modem and a telephone line.

A new generation of instruments designed at the Queensferry Telecommunications Operation have for the first time provided a LAN TCP/IP communications interface. This interface will allow us to have direct network communications between a workstation running virtual remote and the instruments that are being monitored. Almost all advance orders for this new family of instruments have included the instrument virtual remote option. The close links between virtual remote and the common firmware platform allowed the new instrument to be developed with no additional work required to provide virtual remote operation.

Fig. 9. The virtual remote software is also available for PCs.

The concept of virtual remote has stimulated our customers to find interesting new ways to test their networks. Suggestions for enhancements have come from within the Queensferry Telecommunications Operation and also from our customers. These ideas will be evaluated and may be used to develop virtual remote into a product that can help our customers still further.

Acknowledgments

The author would like to thank everyone who has contributed to the development of virtual remote: past and present members of the Queensferry Telecommunications Operation common firmware group, the engineers responsible for installing virtual remote into individual instruments, and the engineers from our marketing and quality departments. We would also like to thank the engineers of the software house Ascada, who contributed to this program under contract to Hewlett-Packard. Special thanks should go to Malcolm Rix for his work on the first prototype version of virtual remote.

Reference

1. M. Rix, "Case Study of a Successful Firmware Reuse Program," *Proceedings of the 1992 HP Software Engineering Productivity Conference.*