Meta-Instruments and the Open Instrumentation Project

Peter D. Hiscocks
Syscomp Electronic Design Limited
Toronto, Ontario
Email: syscomp-support@magma.ca

James Gaston
Wardrop Engineering Incorporated
Toronto, Ontario
Email: james.gaston@wardrop.com

***Abstract*— The Open Instrumentation System provides low-cost instrumentation for use by students, technologists and engineers. In this paper we describe recent developments in the project.**

In addition to the generator and oscilloscope functions currently available, the system hardware can serve as 'configurable instrumentation'. As an example, we show how the hardware may be programmed to form a low-frequency network analyser for the automatic measurement of amplitude and phase response.

The interface between each instrument and host PC is via a USB connection. We show how the USB interface may be implemented as a high-speed asynchronous serial connection (tty or COM port), thereby avoiding the complexities of the USB protocol.

Finally, we indicate some lessons from the development of this project.

I. Introduction

In the early 1990's, personal computers became available to engineering students. This changed the learning environment dramatically:

- Students now had complete and total access to computing technology
- Learning improved and new modes of teaching became feasible
- Course material could increase in sophistication and complexity
- The institutional cost of providing computing to students decreased dramatically

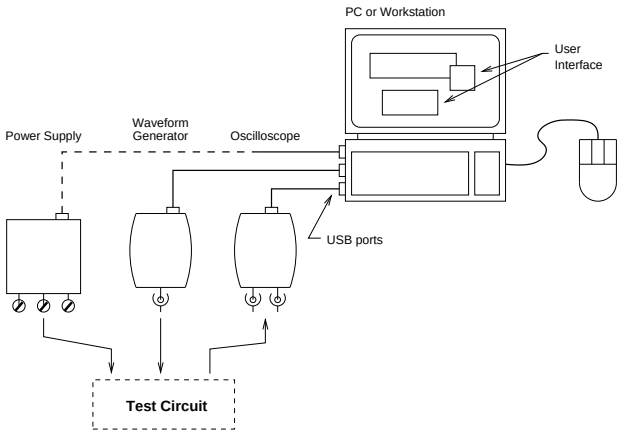

Fig. 1. Block Diagram

The Open Instrumentation Project extends these benefits into the specific realm of electrical engineering. It makes low-cost instrumentation available to engineers, technologists, hobbiests and students. so that it is feasible for anyone to own their own electronics lab. The same benefits that were achieved in computer education with the availability of low-cost computers, will now extend to anyone who needs to use electronic measuring equipment.

Within this conceptual framework, Syscomp [1] has developed programming interfaces, open-source host software and low-cost proprietary hardware for an oscilloscope and waveform generator[1].

The system is organized as hardware devices that connect to a host PC via USB connections as shown in the block diagram of figure 1.

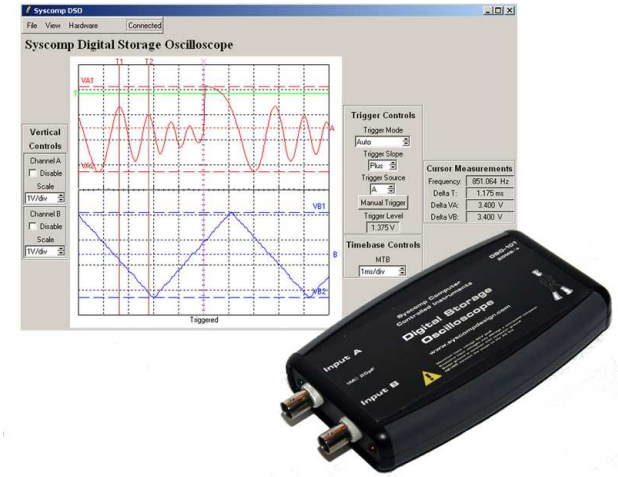

Fig. 2. Oscilloscope Hardware and GUI

The host PC operates a graphical user interface for each of the various instruments. There are no mechanical switches or controls - all operation of the instrument takes place from the computer GUI. The host also supports analytical tools (such as a data histogram), and utilities such as printing and storing waveform data.

[1]As well, a prototype power supply and voltmeter have been demonstrated [2]. Many other instruments are possible within this framework.

The oscilloscope hardware and host graphical user interface are shown in figure 2.

II. Interface

In [2] we described an Open Instrumentation prototype that used an asynchronous serial communication interface. The current version was revised to use the USB2.0 interface, for the following reasons:

- The USB connection can supply electrical power to the instrument hardware, eliminating the necessity for an AC power adaptor. This reduces the size and cost of the hardware.
- The data transfer rate is higher by orders of magnitude. This isn't important for many instruments, but it's critical for the oscilloscope display update rate.
- USB enumeration automatically recognizes instruments when they are connected. It also directs the flow of information so hardware is effectively connected to the corresponding user interface software.
- The serial port is rapidly being replaced by USB. Many modern laptop computers have no physical serial port. This requires anyone interfacing hardware to a PC to deal with the USB interface in some form or other. For one-off or small quantity an external plug-in *USB-Serial converter* or *USB-Printer Port converter* can provide these legacy ports, but this is not a long-term solution.

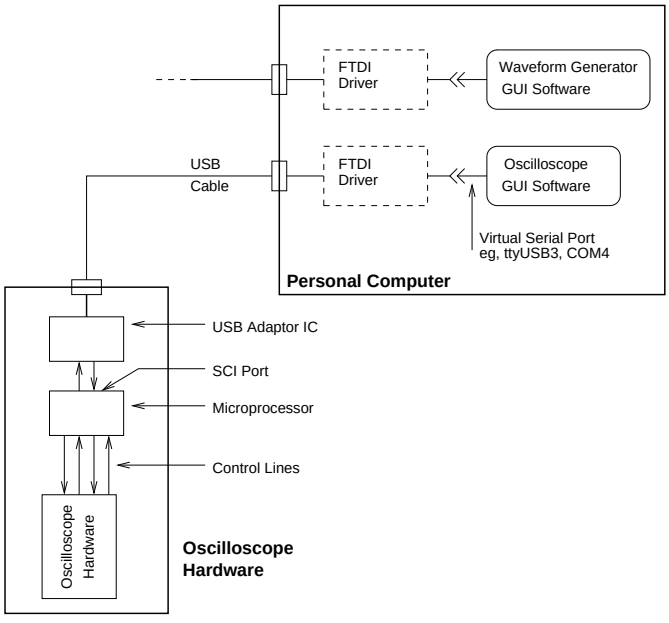

Fig. 3. USB-Serial Communication

Altogether, these points make a strong case for USB. However, the USB protocol is complex and development of USB driver software is very challenging. A small engineering team (student or professional) is unlikely to have the necessary expertise to develop and maintain drivers for a variety of computing platforms.

Fortunately, it is possible to disguise a USB connection as a high-speed serial connection, and thereby avoid much of the complexity [3]. There are two components to this system (figure 3). In the hardware, an IC converts the USB data stream to asynchronous serial format. This is then handled by the serial communications interface of a microprocessor. In the host, driver software intercepts calls to a serial (tty or COM port) and converts them into a USB data stream. In our case, the hardware is provided by FTDI device, FT232BM [4]. On a Windows platform, the FTDI drivers must be installed. On a Linux platform, the drivers are part of the kernel[2].

With this arrangement in place, the instrument software on the host PC can communicate with the instrument hardware by reading from and writing to a virtual serial port. Tcl/Tk is currently used for the open-source instrument software and Matlab has been used for hardware testing. A terminal emulator (*Hyperterminal* under Windows, *Minicom* or *Seyon* under Linux), may be used to send ASCII command strings to the hardware for debugging purposes.

III. Configurable Instrumentation

In [2], we described the concept of *configurable instrumentation*. It is possible to configure the hardware - the waveform generator and oscilloscope in this case - as a new instrument that uses the existing hardware without modification and new host control software that operates both instruments as a *meta-instrument*. This meta-instrument may be added to the repertoire of Open Instrumentation without any cost to the end user.

To illustrate this concept, we have built software that operates the waveform generator and oscilloscope as a low frequency *network analyser*. The analyser displays the amplitude and phase response of a two-port device, which might be a passive network, an active filter or bandwidth-limited amplifier.

A block diagram of the network analyser is shown in figure 4. The waveform generator produces a sine wave signal which sweeps over the frequency region of interest within the range 1Hz to 100kHz. The input signal to the network (the *reference*) is captured by channel A of the oscilloscope, and the output of the network by channel B. The software then compares these two signals to determine the magnitude and phase of the transfer function. The dynamic range of the oscilloscope A/D is nominally 48db. This is augmented by gain-switching the oscilloscope preamp as the sweep is occurring.

As in the case of other software in this system, the network-analyser is constructed with the Tcl/Tk language. Tcl/Tk has a number of advantages for this type of application: it is a scripting language and therefore relatively powerful and simple to use [5], it operates on both Windows and Linux platforms, it provides access to the serial ports and of course it is open source software.

Other instruments are possible. For example the signal generator and oscilloscope can function as a low-power semi-

[2]Since kernel version 2.4

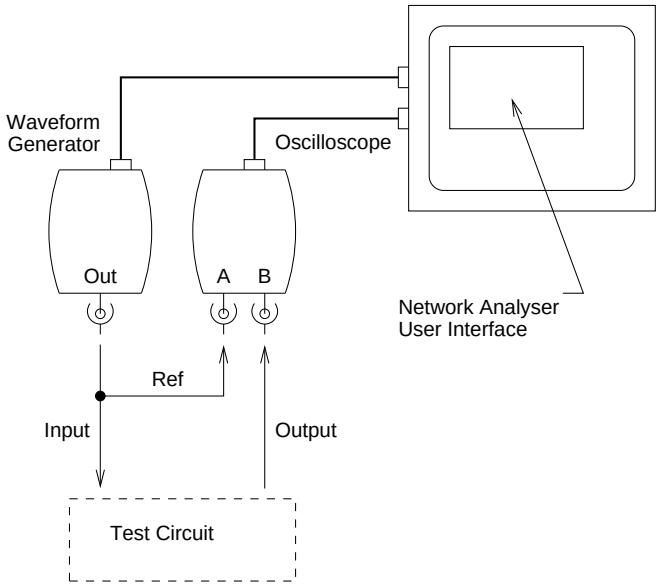

Fig. 4. Network Analyser

conductor device curve tracer. Students have already contributed a waveform editor for the generator and a waveform math package for the oscilloscope. Hopefully others will contribute features and modifications as in other open-software projects.

IV. Lessons Learned

It has been said that *An engineer is someone who can do for one dollar what any fool can do for ten*. That is, the challenge is not so much designing a given device, but designing it to meet a particular cost constraint. The main focus of this project has been reducing the cost of the hardware component so that the unit is affordable by students.

The evolution of this design is guided by several important princibles:

- *It is vital to verify design specifications with the end user. Don't assume you know what the user wants.*
  The original oscilloscope concept was similar to a conventional analog oscilloscope which includes an integral waveform display, but this made the design completely uneconomic. However, students indicated that it was acceptable to use a laptop computer to control the scope hardware. Hosting the control software on a PC has a number of advantages. Switches and hardware controls can be eliminated for a huge saving in cost and size. And a PC host computer can provide additional features - being able to incorporate waveform data in reports, for example.
- *Use the latest technology.*
  Earlier prototypes used MSI (medium scale integration) parts in through-hole packages, with the intention that the hardware could be assembled by students from a kit of parts. However, MSI parts are becoming obsolescent. The highest integration in electronics is in VLSI parts with large pin counts and surface mount packages. A careful economic analysis of designs indicated that a kit unit would be far bulkier and no less expensive than a design based on state-of-the-art surface mount parts.
  This same rule guided the change, described above, from a serial port interface to a USB interface.
  Using the latest technology requires being aware of recent vendor offerings in a variety of magazines and newsletters. It is also necessary to determine whether these products are real or vapourware, to assess the cost of the development environment, and to determine how troubleshooting will take place.
- *Track costs very carefully.*
  It is tedious to track the cost of every last item in a design, but it's essential in order to determine whether the design is saleable at the target price point. This will prevent unpleasant surprises during the manufacture of the hardware.
- *Beware of the Creeping Feature Creature*
  It is a great temptation, especially among engineer technophiles, to add neat features to the design. Features add cost - even in software, because they must be documented and maintained. It is far better to deliver a feature-sparse version in a short time frame and then add enhanced features to new versions.

V. Conclusions

The same technology that has made low-cost computing available to electrical engineering students – sophisticated, low-cost hardware and open software – can provide affordable electronic instrumentation for industrial and university EE lab measurements. The Open Instrumentation Project provides a framework for ongoing development of the concepts. We hope that others will contribute open-source software and open or proprietary hardware to the project.

References

[1] Syscomp Electronic Design Limited
`http://www.syscompdesign.com`
[2] Peter Hiscocks, James Gaston, *An Instrumentation System Using Open Software*, Conference Proceedings, International Conference for Upcoming Engineers, University of Windsor Ontario, May 2005
[3] Jeff Pollard, *USB, FTDI Style*, Circuit Cellar Magazine #132, July 2001, p 28
[4] Future Technology Devices International (FTDI)
`http://www.ftdichip.com`
[5] John Ousterhout, *Scripting: Higher Level Programming for the 21st Century*, IEEE Computer Magazine, March 1998
`http://home.pacbell.net/ouster/scripting.html`