Aquisition and Retrieval Performance of the Tektronix TDS 2014 as a Data Logger

Adam Black
Centre for Advanced Internet Architectures. Technical Report 070717A
Swinburne University of Technology
Melbourne, Australia
adamblack@swin.edu.au

Abstract- **The Tektronix TDS 2014 is a four channel DSO (digital storage oscilloscope) with an RS232 port for remote administration. This document has been written to inform the reader how the oscilloscope can be operated as an automated voltage logger using a PC. We conducted some tests which confirmed that multiple channels of the oscilloscope could be synchronously captured if the trigger conditions in the DSO were configured appropriately. We also tested the serial transfer time of voltage samples, encoded in plain text and binary format. We discovered plain text encoding resulted in considerably longer transfer times compared with binary encoding and is therefore the least preferred method.**

I. Introduction

The Tektronix TDS 2014 [1] is a four input DSO with an 8-bit analogue to digital converter. The device has a colour LCD display, front panel control buttons for manual operation and a 9-pin RS232 serial port for remote administration via a PC or terminal. This report describes how the oscilloscope can be used as a voltage logger by conversing over the RS232 interface.

This document is structured as follows. Section II defines the key technical terms relating to the oscilloscope which have been mentioned throughout this paper. Section III explains how the DSO can be remotely operated through the RS232 interface and introduces some important instructions used to gather information from the oscilloscope. Section IV discusses how voltage data from multiple channels of the DSO can be simultaneously captured and then transferred to a PC which is then experimentally tested in Section V. Section VI evaluates the time required to transfer voltage samples from the oscilloscope to a PC and Section VII concludes the report, followed by the Acknowledgments and References.

II. Terminology

This section introduces some important terminology relating to the oscilloscope. The list of terms and definitions are as follows:

ADC (analogue to digital converter) – Converts an analogue voltage level into a discrete digital representation. The granularity of an ADC can be determined from the number of bits it uses. For example, an 8-bit ADC can digitally represent an analogue signal using 2^8=256 unique voltage levels. The more bits an ADC uses the less uncertainty exists in the digital representation.

Division – The oscilloscope's display has a grid superimposed on the waveform display area. The vertical lines of the grid are used to represent a quantity of time and the horizontal lines are used to represent voltage levels.

Vertical scale – Measured in Volts per division, the vertical scale determines the voltage represented by a single division along the y-axis

Vertical offset – Used to shift the input voltage of a particular channel by a DC signal. This is useful to view large voltages which would fall out of range of the screen.

Timebase – Measured in seconds per division, the timebase determines the quantity of time represented by a single division along the horizontal axis.

Sampling interval – The interval of time between consecutive sample points.

Vertical resolution – The smallest distinguishable voltage increment along the vertical axis.

Trigger – An event which causes the oscilloscope to begin capturing the input waveform(s) of the enabled channels. Triggers have sometimes been referred to as updates, because they cause the waveforms to be updated on the screen. Triggers can be caused by rising edges or falling edges, they can also be forced by the user.

Sample point – An instantaneous measurement of an input voltage at a particular point in time. In practice a sample cannot be captured over an infinitely small period of time and is not truly instantaneous by definition.

Record – A collection of sample points separated in time by the sampling interval. The number of sample points contained within a record is known as the record length.

III. Remote Administration Introduction

The TDS 2014 can be manually operated through the RS232 port with software such as HyperTerminal [2] or cu [3]. Alternatively custom software can be written to automate the acquisition of voltage data.

Instructions issued to the oscilloscope should be transmitted using ASCII character encoding and must conform to the syntax specified in the TDS200/1000/2000 Series Oscilloscope Programmer Manual [4]. There are two types of instructions, commands and queries. Command are used to change settings in the oscilloscope or perform specific actions while queries are used to retrieve information from the oscilloscope. Queries can be differentiated from commands because all queries end with a question mark.

A. Acquiring data

Successful two-way communication between the oscilloscope and PC cannot be established unless the two communicating devices are configured with the same BAUD rate. The BAUD rate is specified in bits per second and determines the speed of the serial connection.

The oscilloscope's `curve?` [4] query is used to transfer waveform data to another RS232 device. The query returns a record of voltage samples for a selected channel. Each sample point is represented as a number originating from the oscilloscope's ADC and must be shifted and scaled accordingly to represent a meaningful voltage quantity.

Each sample point is separated in time by the sampling interval, this can be determined from the following formula:

$$Sampling\ interval=\frac{(Sec/div)}{250}$$

The TDS 2014 has a record length of 2500 meaning 2500 sample points are stored in memory for each channel. The user can elect that a subset of the full record is returned upon a `curve?` query. For example, if the user specifies a data start value of one and data stop value of 200, the first 200 sample points of the record would be transferred.

When collecting voltage data from the oscilloscope it is important to consider the vertical resolution of the samples. In the case of the TDS 2014, each vertical division contains 25 possible voltage levels therefore the vertical resolution of the DSO can be determined from the following formula:

$$Vertical\,resolution=\frac{(Volts/div)}{25}$$

The oscilloscope can send curve data using a width of 1 or 2 bytes. When the data width is set to two the most significant byte in each sample point is specified by the 8 bit ADC value and the least significant byte contains all zeros.

The encoding method used when sending curve data can be specified using either ASCII or binary representations. When the ASCII encoding mode is selected each digit in the curve response is encoded using the ASCII character set. The binary encoding implementations represent each data point in binary integer form.

There are four binary encoding modes available: RIBINARY, RPBINARY, SRIBINARY and SRPBINARY [4].

RIBINARY and SRIBINARY represent each data point as a two's complement [] signed integer. The RIBINARY mode sends the most significant byte first and the SRIBINARY mode sends the least significant byte first, assuming the data width is 2 bytes.

RPBINARY and SRPBINARY represent each data point as an unsigned integer. RPBINARY transfers the most significant byte first while SRIBINARY transfers the least significant byte first, assuming the data width is 2 bytes.

Table 1 displays the range of possible values received from a `curve?` query using different encoding methods and data widths. Note the values: -128 and -32768 are reserved to indicate an invalid data point when using signed encoding and a zero is used to indicate an invalid data point when using unsigned encoding.

Data Encoding Mode	**Range (width = 1)**	**Range (width = 2)**
ASCII	-128 to 127	-32768 to 32767
RIBINARY/SRIBINARY	-128 to 127	-32768 to 32767
RPBINARY/SRPBINARY	0 to 255	0 to 65536

Table 1: Range of data point values

B. Manual Triggering

When multiple channels need to be captured simultaneously the oscilloscope must be prevented from triggering automatically. The `curve?` query can only return data belonging to one channel at a time which means unwanted triggers can occur between the acquisition of two channels and cause the records to be out of sync.

A possible solution is to prevent the oscilloscope from triggering updates unless explicitly directed. This means the user can trigger the oscilloscope, collect curve data from the channels of interest and then initialise a new trigger when a new set of sample is required.

To use manual triggering set the trigger mode to normal and use the oscilloscope's external input as the trigger source. Then adjust the triggering level it's maximum setting which is 1.6V and short-circuit the external input channel to ground. Once completed successfully the oscilloscope should never trigger updates automatically and a manual trigger can be forced by issuing the command, `trigger force`.

IV. Acquiring Oscilloscope Waveform Data

This section describes the principles for acquiring waveform data which has been synchronously captured from two or more channels of the oscilloscope.

A. Equipment

- A PC with a serial port.

- Tektronix TDS 2014 oscilloscope with necessary coaxial cables.
- 9-pin Female-Female RS232 serial cable.

B. *Method*

Figure 1 represents the sequence of events when reading two oscilloscope channels. This concept can be extended to read any number of channels.

The basic principle is; once a trigger has taken place the waveform data will be retained in memory until the next trigger is issued. Between triggers the waveform data can be collected one channel at a time.

The following is a more detailed explanation of the flowchart in Figure 1.

- The oscilloscope is sent the `trigger force` command, which updates the oscilloscope with new voltage data.
- The process then has to wait until the trigger update is complete. The `trigger:state?` query can be used to poll the oscilloscope regarding the status of the most recent trigger. A status of "Ready" means the trigger has been completed.
- The next step (`select source: CH1`) informs the oscilloscope which channel to retrieve when the `curve?` query is issued.
- The `curve?` query retrieves the sample points belonging to channel one.
- The curve/acquisition process is then repeated for channel two.
- After the required channels have been read a new trigger can be issued (using the `trigger force` command) and the process can be repeated from the start.

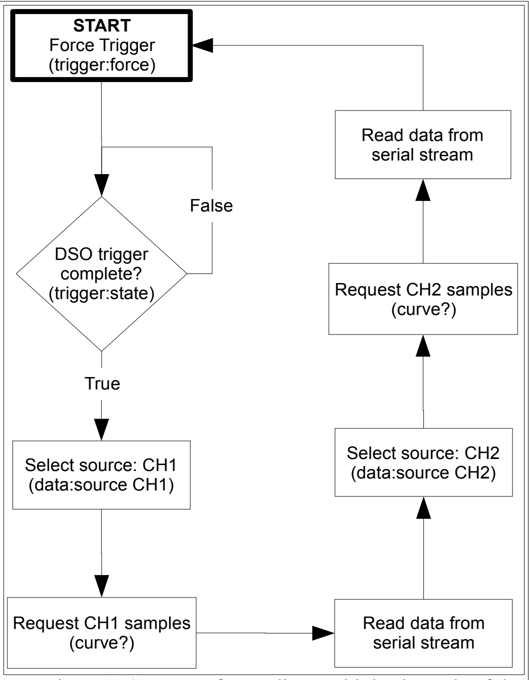
START
Force Trigger
(trigger:force)
False
DSO trigger complete? (trigger:state)
True
Select source: CH1 (data:source CH1)
Request CH1 samples (curve?)
Read data from serial stream
Select source: CH2 (data:source CH2)
Request CH2 samples (curve?)
Read data from serial stream

Figure 1: Sequence for reading multiple channels of the TDS 2014. Commands are queries are shown in brackets.

V. Synchronous Channel Capturing

In this section we will test whether voltage data from multiple channels of the oscilloscope can be synchronously captured and then transferred to a PC as described in Section IV.

A. *Equipment and configuration*

- A PC running FreeBSD 6.2 with a serial port. The following software was chosen to communicate with the oscilloscope over the serial link:
  - python 2.4.3.
  - pyserial 2.2.
- Tektronix TDS 2014 oscilloscope with necessary coaxial cables.
- DSE (Dick Smith Electronics) Q 1459 digital multimeter (used for generating a pulse train).
- 9-pin Female-Female RS232 serial cable.

B. *Method*

The 50Hz square wave output of the DSE multimeter was connected to channel one and two of the DSO which were both configured with a vertical scale of 1 V/div and no vertical offset. The data start and stop values were set to 1 and 200 respectively and manual triggering was configured as explained in Section III.B.

The oscilloscope was triggered and the records for channel one and two were retrieved by issuing the `curve?` query. This was repeated with a 5 second interval between consecutive triggers for a total of 200 iterations.

C. *Results*

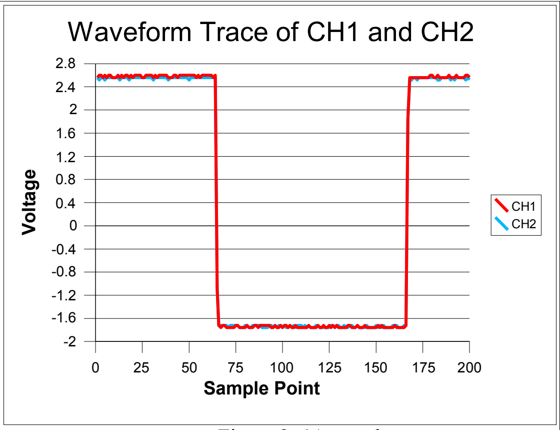

Figure 2: 1st record

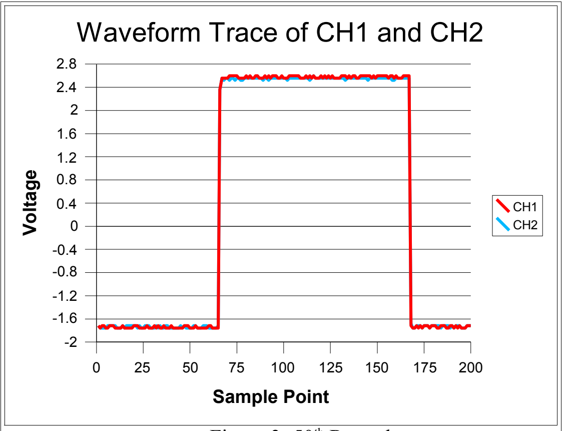

Figure 3: 50th Record

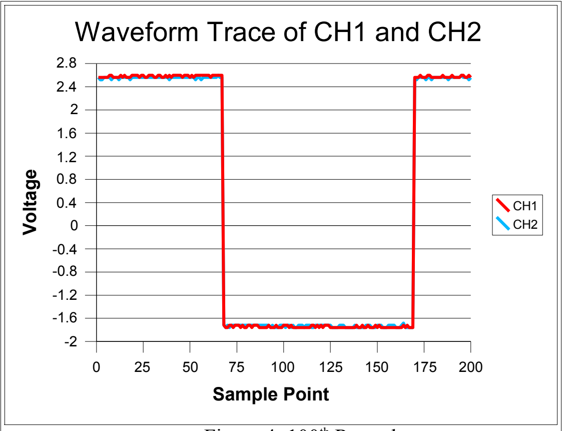

Figure 4: 100th Record

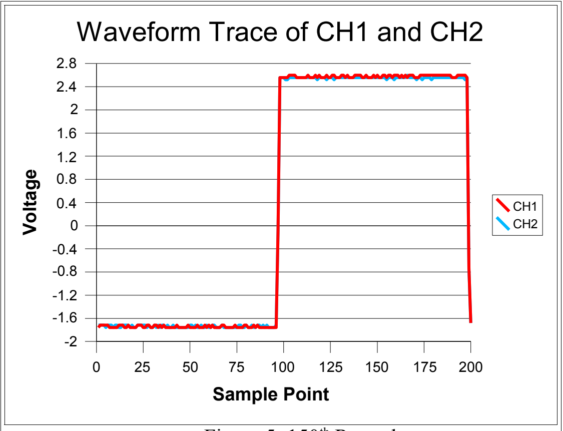

Figure 5: 150th Record

Figures 2 through 5 show several waveform traces of channel one and two captured at different points in time. In theory we would expect the rising and falling edges to occur at the same time which would indicate the channels were being simultaneously recorded.

It is quite clear from the graphs that both channels were acquired in phase. There is however some voltage variability between the both channels at identical times. This is most likely due to factors such as induced random noise in the system and manufacturing imperfections in the oscilloscope which cause the characteristics of one channel to differ very slightly from another.

D. *Conclusion*

From the results we can see the oscilloscope is capable of capturing multiple channels simultaneously. This can be done by manually triggering the oscilloscope and then receiving the curve data for the required channels. Once complete a new trigger can be initialised and the process can restart from the beginning.

VI. Serial Transfer Performance

The TDS 2014 DSO allows the user to acquire a record's length of sample points by issuing the `curve?` query. If continuous voltage logging is required the user must repeatedly trigger and capture curve data until the required number of records have been collected. The interval of time between subsequent triggers (and hence waveform captures) is limited by the trigger time and serial transfer time.

In this section we will examine the time required to transfer a varying number of curve sample points using ASCII and binary encoding. These encoding modes have been introduced in Section III.A.

A. *Equipment and configuration*

- A PC running FreeBSD 6.2 with a serial port. The following software was chosen to communicate with the oscilloscope over the serial link:
  - python 2.4.3.
  - pyserial 2.2.
- Tektronix TDS 2014 oscilloscope with necessary coaxial cables.
- Instek GPC-1850D Power Supply.
- 9-pin Female-Female RS232 serial cable.

B. *Method*

The RS232 interface of the oscilloscope was configured with a BAUD rate of 19,200 bps[1], the flow control was set to hardware and no parity bits were used.

Next we disabled automatic triggering as explained in Section III.B and set the vertical scale to 2V/div with no vertical offset.

We ran two types of experiments to assess the oscilloscope's RS232 transfer performance.

The first experiment was to measure the time required to send various quantities of sample points over the RS232 link. This was achieved by connecting a fixed 5V signal from the power supply to channel one of the oscilloscope and then manually forcing a trigger once. We then measured the time required to send the `curve?` query and fetch records of various lengths. The record

1 The BAUD rate includes the start and stop bit used to maintain synchronisation between the PC and oscilloscope, hence for a BAUD rate of 19200 bps the actual transfer rate would be 19200 / (8+2) = 1920 bytes/sec.

length was varied by holding the `data:start` position constant and altering the `data:stop` position from 50 to 2500 in increments of 50. For each `data:stop` settings we measured the `curve?` transfer time over ten individual trials and averaged the result. The experiment was conducted for three encoding modes; ASCII, RIBINARY and RPBINARY.

The second experiment was to determine the curve response time of the oscilloscope, measured from the time the curve query was issued, up until the time the first sample was received on the serial input buffer of the PC.

For this experiment we used a mixture of RPBINARY and ASCII encoding modes. We also ran two tests for each encoding mode using a different number of digits per sample point. For the first test an input voltage of 0.25 VDC was fed into channel one of the oscilloscope to produce sample points containing one digit as demonstrated below:

$$ADC_{level}=\frac{0.25\,V}{2\,V/div}*25\,levels/div\approx 3$$

In the second scenario a fixed 5 VDC signal was connected to channel one, producing an ideal ADC level of 62.5. After rounding the input voltage to an integer value we would expect to see two digits per sample point.

For the various input voltages and encoding modes used, we measured the curve response time for different data stop positions ranging from 50 to 2500 in increments of 50. The data start position was held constant at one and ten trials were repeated and averaged.

C. Results

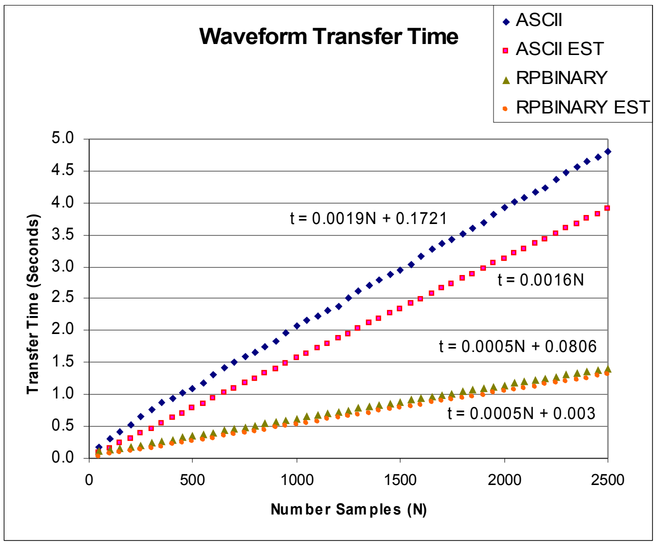

Figure 6: Waveform transfer time for different quantities of data points and encoding methods

Figure 6 compares the theoretical and experimental transfer times for the ASCII and RPBINARY transfer modes.

When the oscilloscope is configured to send `curve?` query results using ASCII encoding it transmits 1 byte per digit. Below is an example of some ASCII encoded curve data.

2,-10,123,56,-119,-4\n

From the sample data above it should be noticed the number of bytes transmitted per data point can range from 1 to 4 bytes, only taking the digits into account.

In the experiment a 5 VDC signal was used which caused each sample points to contain two digits. Therefore for each data point transferred the oscilloscope would need to send two numeric digits and a comma or newline. The following formula can be used to estimate the transfer time (t) for a particular number of data points (N) assuming ASCII encoding.

$$t_{ASCII\ EST}=\frac{3*N*10}{19200}$$

When the curve data is transferred in binary mode the oscilloscope sends some header information followed by the curve data and finally a newline. The header consists of a "#" to indicate the beginning of the curve transfer, the successive character is an single numeric digit in ASCII format indicating how many bytes make up the subsequent ASCII number, which informs the user how many sample points will be transferred by the oscilloscope.

Below is the example of what the header should look like when the record length is set to 1, 10, 100 and 1000.

#11

#210

#3100

#41000

The following formula can be used to estimate the transfer time (t) for a particular number of data points (N) assuming binary encoding.

$$t_{BIN\ EST}=\frac{(len(header)+1+N)*10}{19200}$$

From the results we found that the RIBINARY and RPBINARY encoding modes performed very similarly so we have chosen not to display the RIBINARY results in Figures 6 and 7.

Figure 6 shows that the experimental transfer time for RPBINARY encoded data is closely related to the estimated time since the gradient of the two graphs is the same to four decimal places. The Y-intercept is the only significant difference between the two curves which indicates an overhead is imposed when issuing a `curve?` query.

In Figure 6 we can see the gradient and y-intercept differ quite significantly between the experimental and theoretical ASCII plots. This suggests the RS232 link is not being fully utilised during transfer. Since this does not seem to occur when using RPBINARY encoding, it is reasonable to assume the oscilloscope is more efficient at generating binary encoded data than it is at producing ASCII encoded data.

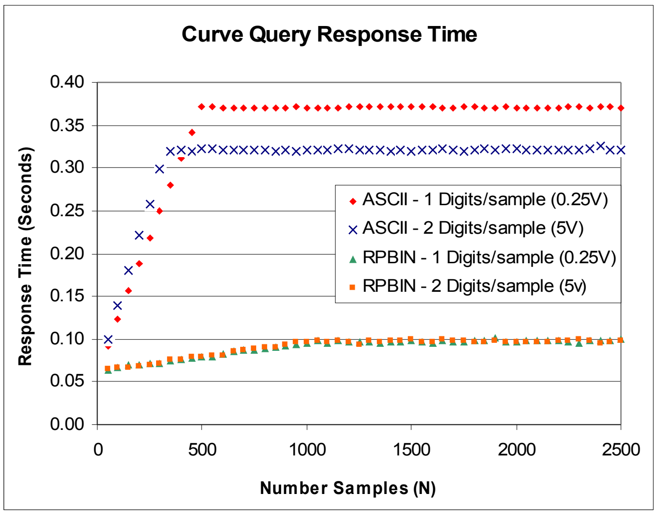

Figure 7: The time required for the oscilloscope to respond to a curve? query

From Figure 7 it seems the oscilloscope buffers the curve query results before sending them to the host. The trend of the curves suggest that the oscilloscope's buffer must be full before any data points are transmitted, except when there are insufficient sample points to fill the buffer. In this case the oscilloscope begins transferring curve data once the last sample point in the record is buffered.

The oscilloscope appears to have a 1 KB buffer for outgoing serial data since the curve query response times plateau when the samples consume more than 1 KB of memory. For example, consider the ASCII encoded data containing one digit per sample. Each sample point requires 2 bytes to buffer because of the commas separating each value, hence 500 samples would consume approximately 1KB of memory. This is also evident for the 2 digits/sample ASCII data which require 3 bytes per data point, and the RPBINARY data which require 1 byte per sample point regardless of the number of digits contained within each sample point.

Extrapolating the curves in Figure 7 to N = 0 reveals that all curves intersect the y-axis at approximately 0.06 seconds. This suggests that responses to `curve?` queries will begin no sooner than 60 milliseconds after the query is issued.

D. Conclusion

The oscilloscope offers various encoding modes when sending records. The binary modes encode each sample point as a binary integer while the ASCII mode sends each digit using ASCII characters.

From Figure 6 we can say the ASCII mode is the least efficient encoding method requiring approximately 5 seconds to transfer a full record of samples, compared to 1.5 seconds for RPBINARY encoding. This means the ASCII mode should be avoided if the quickest possible record transfer times are desired.

VII. Conclusion

The Tektronix TDS 2014 is a digital storage oscilloscope which has an 8 bit analogue to digital converter and four input channels. The oscilloscope features an RS232 serial port which allows for remote administration.

A useful command for acquiring voltage data from the oscilloscope is the `curve?` query which returns a record of samples for a selected input channel. The data returned from a `curve?` query is an integer originating from the oscilloscope's ADC and can be converted to an actual voltage by shifting and scaling the value.

In Section IV the report explains theoretically how multiple channels of the oscilloscope can be captured simultaneously by disabling automatic triggering.

In Section V we have conducted an experiment to test whether two channels of the oscilloscope can be captured simultaneously by connecting a square wave signal into channel one and two of the DSO. We then transferred voltage samples to the PC and plotted a waveform trace of both channels to test whether the they were captured in phase. From the experimental results we confirmed that multiple channels could be captured simultaneously and transferred to a PC by triggering the oscilloscope manually.

In Section VI we have tested the serial transfer speed of the oscilloscope when sending voltage samples to a PC over the RS232 port. We set the oscilloscope's BAUD rate to the maximum speed of 19200 bps and measured the time taken to retrieve a varying numbers of sample points using ASCII and binary data encoding modes. From the results we found the ASCII mode to be the least efficient requiring approximately 5 seconds to retrieve a record of 2500 sample points, while the RPBINARY mode took approximately 1.5 seconds. The RIBINARY and RPBINARY modes performed very similarly.

Acknowledgments

I'd like to thank Thuy Nguyen and Grenville Armitage for their advice and assistance in the development of this report.

References

[1] "Digital Storage Oscilloscopes TDS1000 Series • TDS2000 Series", http://www.tek.com/site/ps/0,,3G-15314-INTRO_EN,00.html (accessed 12th July 2007)

[2] "What is HyperTerminal?", http://www.tech-faq.com/hyperterminal.shtml (accessed 12th July 2007)

[3] "Unix man pages: cu", http://www.csee.usf.edu/cgi-bin/man-cgi?cu+1 (accessed 12th July 2007)

[4] Tektronix Inc, "TDS200/1000/2000 Series Oscilloscope Programmer Manual", http://www.physics.utah.edu/~kieda/tk2002_prog.pdf (accessed 12th July 2007)

[5] Wikipedia contributors, "Two's Complement" Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Two%27s_complement&oldid=144854173 (accessed 17th July 2007)