An open source implementation of a data acquisition system for a current pulse ERT system using an industry standard interface

K.H. Hauslaib[1] , E.W. Randall[2]

[1]*Electrical Engineering, University of Cape Town, Rondebosch 7701, South Africa*
[2]*Chemical Engineering, University of Cape Town , Rondebosch 7701, South Africa*

Abstract

The design and applications of the Current Pulse Electrical Resistance Tomography (ERT) system developed at the University of Cape Town (UCT) has been described in several journal papers and at previous world congresses. Details of the electronic circuits were published at WCIPT6. This paper deals with the replacement of the USB DAQ and two micro-controllers with an industry standard IO system available from National Instruments and enables the transmission of data at 1000 fps over an TCP/IP network. The LabVIEW code will be placed in the public domain to coincide with WCIPT7. This will provide a complete open source design for an ERT system.

Keywords: Electrical Resistance Tomography, current pulse, National Instruments, RIO, LabVIEW

1. Introduction

Since the proof of concept design was published in 2003 several implementations of the system have been published. On-going developments resulted in a reliable system where the ERT hardware was implemented on two circuit boards, i.e. a 16 channel amplifier and current source plus multiplexers, [1]. The system was capable of capturing 850 frames/second for adjacent or opposite pairs strategies from a ring of 16 electrodes. In addition to the two circuit boards this configuration requires a USB DAQ , two MCUs and an embedded PC. Although the embedded PC could be operated remotely the data sets could not be transmitted to a remote system in real time. A version [2] of the instrument was implemented which allowed single frames to be transmitted via TCP/IP to a remote PC on request. This required a PC located at the ERT instrument running a minimal version of the UCT capture software. The remote software was coded in Matlab and only capable of operating at a few frames/second, possibly because of poor implementation. It did, however, demonstrate the viability of remote real-time feedback control based on tomographic measurements.

As the USB DAQ used for the past ten years is nearing the end of its production and is lacking in driver support for new operating systems an alternative was investigated which would in addition allow remote system operation. TCP/IP communications has been proved effective in instrumentation applications and was incorporated in a tomography instrument using proprietary hardware and software which achieved a capture rate of 100 fps for a 12 electrode system. [3] The benefits of employing FPGA devices in the high speed front end electronics of a PET scanner has been described [4]. The prime consideration in this work was to select a modern Ethernet linked industry standard device capable of high speed data transfer to a PC over an IP network. This would enable remote data acquisition with the possibility of feedback control to the process being monitored.

The National Instruments RIO technology, which incorporates both an MCU and an FPGA, was found to meet these requirements and has the additional benefit of allowing processing previously carried out on the PC to be implemented on the device. The RIO is fully supported by LabVIEW which provided high level functions which greatly simplify the implementation of the TCP/IP functions and programming the FPGA. The control of the instrument, the frame capture code and data transmission is significantly more reliable as it is executed by the RIO in a "real-time" environment. The TCP/IP protocol allows the data frames to be transmitted to a remote

PC (at any distance), at a capture rate of 1000 frames/second (from 16 electrodes, using adjacent pairs), providing adequate network band width is available.

LabVIEW provides high level functions simplifying the coding of the TCP/IP communications and programming of the FPGA and MCU. With the high speed frame transmission the embedded PC becomes superfluous and has been omitted. The replacement of the DAQ, embedded PC and the MCU with the entry level version of the RIO (sbRIO 9623) has therefore resulted in a cost saving. The code developed for this device may be installed on higher performance cRIOs if more local processing of data is required. The implementation of a reconstruction algorithm on the RIO is currently being considered. RIO System Architecture

National Instruments offer a wide range of data acquisition systems. The DAQ range includes PCI bus, USB and Ethernet based systems but control of the IO is via instructions from code executed on the PC and is not strictly "real time", particularly under the Windows operating system. The PXI range, which is primarily targeted at automated test and measurement, although suitable for our application was not considered for physical size and cost considerations. A PXI version [5] of a tomography system was, however, successfully implemented using LabVIEW. The NI RIO range incorporates local processing with either USB or Ethernet connectivity to a computer and offers compact single board versions specifically for OEM applications. The development work described in this paper was done on the sbRIO-9623 as the primary objective was to get a viable low cost solution for the ERT data acquisition.

The general architecture of the RIO devices is shown in Fig 1 and photographs of both the low cost single board version, which is targeted at OEM applications, and the more advanced chassis based cRIO device in Fig 2.

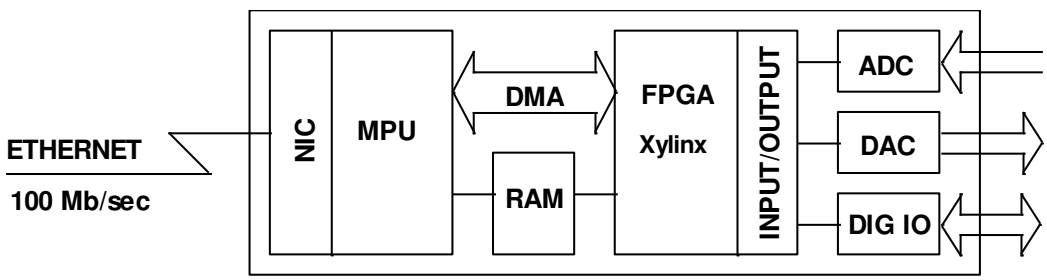

Fig. 1. National Instruments RIO system architecture

The RIO incorporates both a microcontroller and FPGA. The FPGA services all IO devices and can also be programmed to process the data. In this case all the timing functions (at uS resolution) and multiplexer control is implemented on the FPGA. The frame data is passed to the MCU via the DMA channel and transmitted using TCP/IP via the NIC. The processors used in the various RIO devices range from Freescale MCUs on the single board systems to more powerful PowerPC chips on the chassis versions. In our case the MCU is the rate limiting component in transmission speed. It is believed that the PowerPC would be capable of running the reconstruction algorithm in addition to the data acquisition and transmission functions. The cRIOs are also available with 1Gb/s NICs and therefore significantly higher network data transfer rates can be expected. The actual sustained data transfer rates achievable is dependent on the transmission protocol used and the implementation of code to transfer data between the FPGA and the MCU. The optimal structuring of this code, using the high level LabVIEW functions, has a significant effect on the frame capture rate. A major benefit of the NI devices is that they are fully supported by the LabVIEW software which is widely used in scientific and engineering applications. The code developed for the RIO architecture is written in three sections (PC, MCU, FPGA) and tasks allocated to appropriate devices to optimise system performance.

Fig. 2. Single board RIO (LHS) targeted for OEM applications and the more powerful cRIO (RHS).

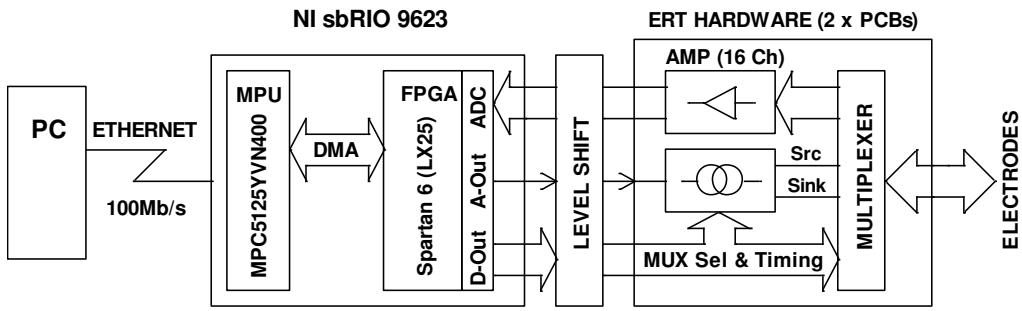

Fig. 3. ERT system configuration using National Instruments sbRIO 9623 IO device with ERT hardware which comprises of 16 channel amplifier, current source and multiplexer system.

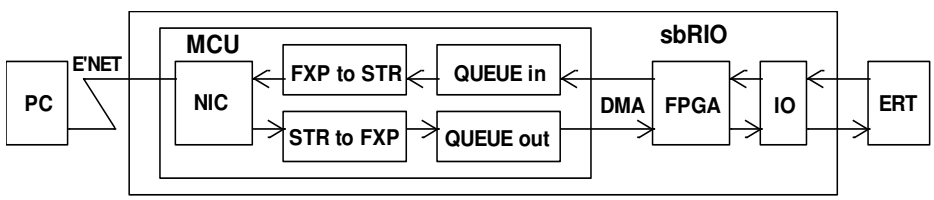

Fig. 4. Data flow between the ERT hardware, FPGA, MCU and PC.

2. Configuration for ERT application

The PC, sbRIO-9623 and ERT hardware are connected as shown in Fig. 3 and the data flow between the sections in Fig. 4. The level shift board between the RIO and the ERT system is to accommodate bi-polar voltage readings for the ADC (when opposite measurement strategy is used) and converts digital logic levels between 3,3V and 5.0V. Details of the ERT electronics and system operation have been published [1] and a full discussion of various multiplexing options (including strategies for 3D data sets) described [6]. This implementation of the system produces identical data sets to that of the instrument mentioned above.

The Freescale 32 bit MPC5125 MCU installed on the sbRIO is specified to run at a clock speed of 400MHz but only operates reliably in this configuration at 40MHz. This problem is related to timing of the DMA data transfers. In this application the MCU was found to be the bottle neck in moving data to the PC via the NIC incorporated in the processor. Full specifications of the MCU may be found in the Freescale data sheet MPS5125

3. Choice of Communications Architecture

Although the NIC on the sbRIO 9623 is specified at 100Mb/s the actual rate achieved, using packet based protocols, for sustained data transfer is significantly less and depends on several factors such as the protocol used (UDP, Network Streams, TCP/IP, etc). UDP is high speed but there is no acknowledge of successful data packet transfer and would only be reliable on a small dedicated network. "Network Streams" is a technique

specific to LabVIEW which has benefits in large data acquisition systems but again is low speed. TCP/IP is relatively fast packet based transmission protocol with acknowledge and data transfer is guaranteed lossless, even on high traffic networks. It has been found that actual structuring of the user written code which transfers the data between the FPGA and MCU and subsequent transmission is critically important in optimising the speed of sustained data transfer rates. The implementation described below achieved an order of magnitude increase in speed compared to earlier versions of the code and reliably achieves the transmission rate target of 1000 frames/second. This rate has been shown to be adequate for applications such as velocity profiling using cross correlation techniques, [7].

When implementing bi-directional data transfer, required for feed back control, the data rates are significantly reduced (150 frames/second). This speed reduction could, however, be largely compensated for by local processing (e.g. frame averaging) on the RIO's MCU which would significantly reduce the amount of data to be transferred to the remote PC.

4. LabVIEW Code Structure

The ERT LabVIEW application software is coded as separately compiled modules for the FPGA, MCU and PC. as shown in Fig. 5 below. The FPGA controls all IO required to acquire the frame data from the ERT hardware and passes it to the MCU via the DMA channel. The MCU reads the data and transmits it to the PC via the NIC for plotting, reconstruction and logging for off line processing.

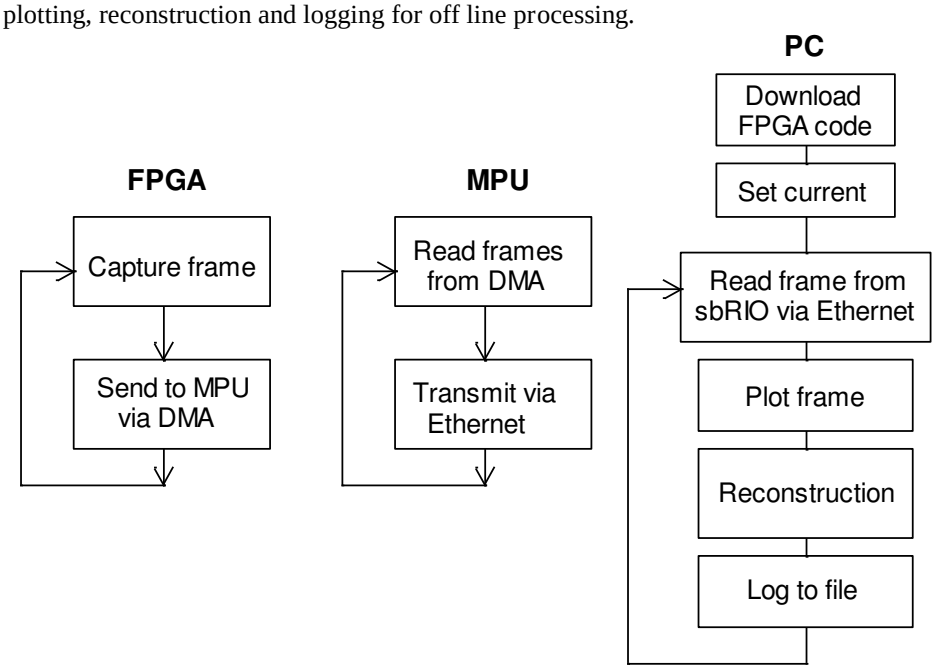

Fig. 5. LabVIEW code modules for the FPGA, MCU and PC.

4.1. FPGA

The function of the FPGA code is to generate the timing pulses and control the multiplexers required by the ERT hardware in order to acquire data frames. This involves specifying the electrode sequencing, current pulse timing and ADC reading. Code modules for different measurement strategies may be installed at run time. This code emulates the functions performed by the Freescale GB60 MCU in the previous version of the instrument which has been previously described in detail, [1]. The USB DAQ reads all 16 ADC channels and transmits them in the order of measurement (1..16). The PC code had to discard readings from measuring electrode pairs common to the excitation electrodes and rearrange the data to produce the data frame.

An improvement in this code is that only the valid ADC readings (13 of the 16 channels for adjacent pairs) are transmitted to the PC and in the correct sequence, i.e. the typical adjacent "U curve" plot from the adjacent pairs sequence is directly received by the PC (no sorting required). The acquired data frame is reduced from 256 to 208 readings and thus reduces the frame measurement time. This results in the capture rate increase from 850 to 1000 frames/second. Specific versions of the FPGA code may be created to implement various measurement strategies (e.g. adjacent pairs, opposite pairs, etc.) and may be loaded at run time.

4.2 MCU

In this case the MCU's function is to extract the data from the DMA channel, re-structure it, and implement the TCP/IP protocol for transmission of the data frames to the PC via the NIC. At a frame rate of 1000 frames/second the sbRIO's processor is working at maximum capacity and is the limiting factor in frame transmission rate. The PowerPCs installed in "C" series RIO's would enable local processing of data (e.g. frame averaging and possibly reconstruction) to be carried out in addition to the TCP/IP communications at a higher transmission rate of 1Mb/s. The averaging would reduce network traffic and enable feedback control at a higher loop rate.

4.3. PC

After initial communication with the RIO to initiate operation the PC monitors the Ethernet port and reads the data frames as they become available. The data may be plotted, reconstructed for real time visualization or stored for later off line processing

5. Code Implementation

The flow-sheets for the sbRIO and PC code modules are presented in Figs 5 and 6 below. They are intended as a broad outline of the sequence of operations performed and will facilitate the understanding of the actual code which is available on the web site (http://instruments.uct.ac.za/tomo/LVsource),

5.1. RIO code

Fig. 5 schematically represents the code that is executed by the FPGA and MCU on the sbRIO. Each of these execute their own software. These code modules (VIs) execute simultaneously and independently. Data transfer between the modules is via the DMA channels (NI's recommended method) and the transfer routines insure no buffer overflows and hence reliable operation. Synchronization between the FPGA and MCU is achieved by the handshake inherent in the DMA transfers between these modules.

5.1.1. FPGA software

The software on the FPGA is structured so that it that executes as a linear cyclic process. The LabVIEW code that is compiled to run on the FPGA does not permit the creation of dynamically sized arrays, that is, arrays in which the size (and hence the memory allocation) is variable. This restriction is inherent to the way that FPGAs operate and the way that arrays are implemented on them. Since FPGA code ultimately reduces to a set of interconnecting logic gates and look-up tables, the compiler has to set a fixed array size and interconnections can only be made or broken during compilation, not dynamically during program operation.

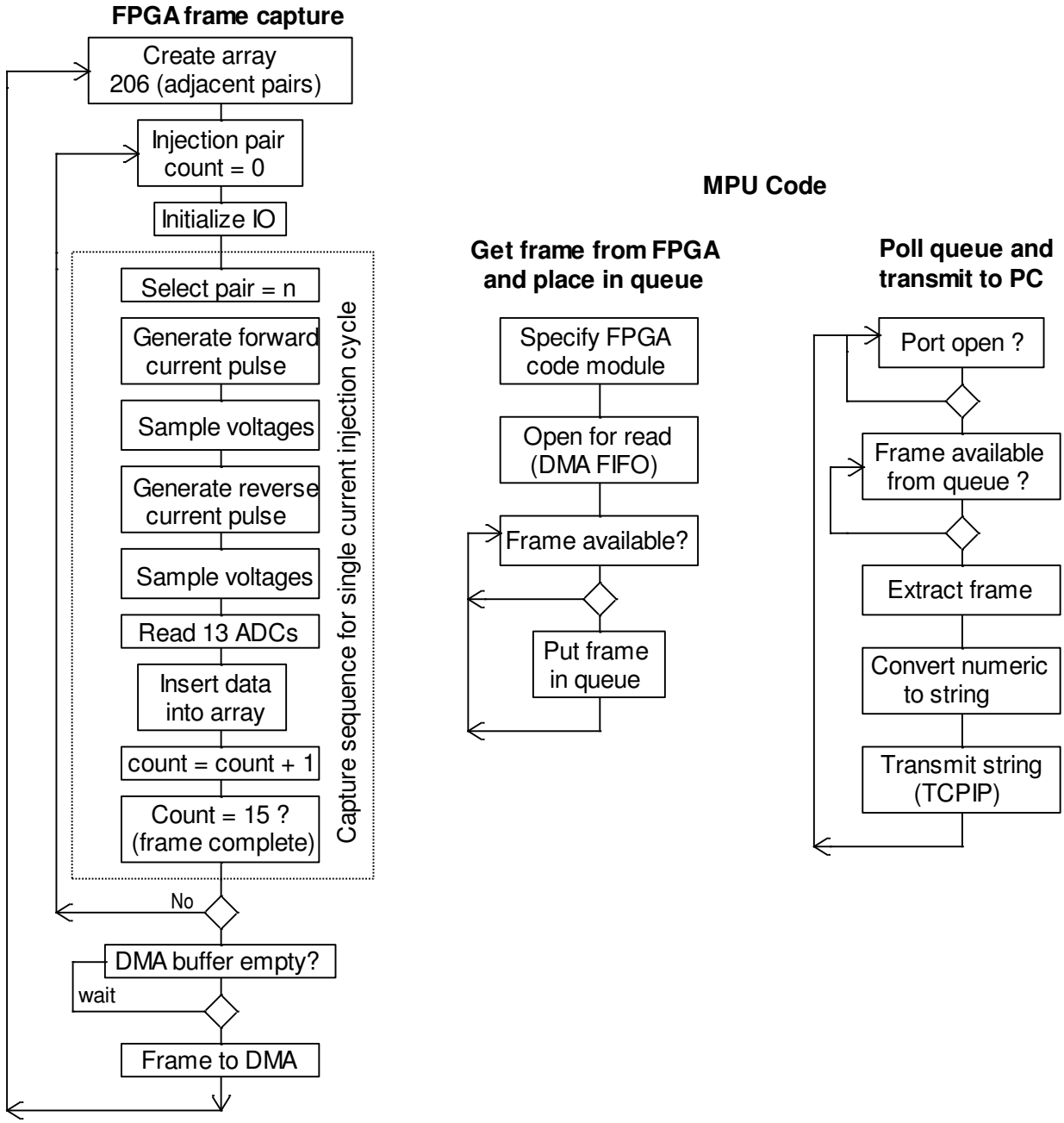

Fig. 5. RIO code is implemented on the FPGA and MCU in three independent LabVIEW VI's

Therefore, we create an empty array of 208 elements (for adjacent or opposite pair measurements) at program. initialization. This number will always remain constant, regardless of which current injection strategy is used, as one always needs to discard the data points measured common to the current injection electrodes. At this point the counter for the frame acquisition sequence is initialized (count = 0). After initialization the current pulse and ADC read sequence is repeated 16 times until a complete frame is collected.

Next, an unsigned short integer is created. This integer acts as a counter and its value determines the current injection sequence and ADC read order. It is reset at the start of each loop, as each iteration of the program loop constitutes a single frame of data. A nested loop runs and increments the counter. Each iteration of the nested

loop constitutes $^{1}/_{16}^{th}$ of a frame. Only 13 ADC inputs are read in any of these nested loop iterations, as the data at the remaining ADC inputs is common to the injection pair and thus not relevant in the frame reconstruction.

A novel feature of this implementation is the ability to sample the ADC lines in a non-sequential fashion. This, in turn, allows us to obtain a full frame in order, meaning that the need to shuffle and discard data points is eliminated. This not only makes the program operation more efficient, but also reduces the traffic on the Ethernet link by approximately 20%, which translates directly to an increase in frame capture speed by this amount.

At the end of each of these nested loops, the 13 acquired ADC values are inserted into the appropriate space in the 208 element array created at program start. Once the array is full (i.e. a full frame has been captured), the software probes a DMA FIFO buffer that has been established between the FPGA and the MCU. If this buffer is empty, the data is transferred to the buffer, at which point it is available to the MCU for extraction. If the DMA FIFO buffer is not empty, the FPGA code pauses until such time as it is empty. This acts as a synchronisation method for the code running on the MCU and the FPGA – if the MCU takes longer to extract data from the DMA than what it takes the FPGA to insert it, the FPGA will wait for the MCU. Conversely, should the MCU extract data faster than the FPGA supplies it, the MCU will wait for the FPGA. This technique ensures that no buffers can fill up and overflow and has proved reliable.

Experimental results have shown that the maximum speed of operation increases as DMA write operations decrease. For this reason, we transmit an entire ERT data frame as a single DMA transfer operation as opposed to $^{1}/_{16}^{th}$ of a frame at a time, which is programmatically much simpler and uses approximately 90% less FPGA space. Although using more FPGA resources and adding to program complexity, this approach more than doubled the frame rate achieved. A larger FPGA (available on compact RIOs) would allow us to read multiple frames and average them before writing them into the DMA buffer, thereby increasing speed further.

5.1.2. MCU software

The code residing on the microprocessor contains two loops running simultaneously in a so-called “Producer-Consumer Architecture.” One loop acquires frames from the DMA FIFO buffer mentioned above and places them in a queue. The other loop reads the data from the queue and transmits it via a TCP/IP link to the PC. Having these distinct operations running in simultaneous independent loops allows us to take advantage of threading, which significantly increases the rate at which the data is acquired and transmitted to the PC.

The loop responsible for obtaining data from the FPGA (see Fig. 5) functions as follows: The FPGA code module is referenced and examined for an established DMA FIFO buffer. The MCU then polls this DMA FIFO buffer and waits until a frame is present in the buffer, it is then read and placed in the queue for transmission via TCP/IP. This process repeats until program termination.

The loop responsible for transmitting the frames to the PC (see Fig. 5) functions as follows: Query the port on the PC to determine whether it is open. Continue doing this until the port is open. If the port is open, extract a frame from the queue, if the frame is present. If a frame is not present on the queue, continue polling the queue until a frame is present. An idiosyncrasy of the LabVIEW TCP/IP transmit function is that only string data may be transmitted. If we transmit the frame data as an ASCII representation of the data, we are increasing the amount of transmitted data by a factor of 8, due to the way that ASCII strings are represented. For this reason, the data is “flattened” to string, that is, the binary representation of the frame is “forced” to represent a string. The ASCII value of this string will appear to be garbage, but all data will be contained within it. This string is then transmitted via TCP/IP and the loop restarts.

5.1.3. PC software

The software on the PC consists of three parallel processes, as is shown schematically in Fig. 6. Each of these processes is running in its own loop: The first loop acquires a frame from the Network Interface Card (NIC), the second loop queues the data and the final "while loop" logs the data to disc. This structure is an example of what NI refers to as “Producer-Consumer Architecture”, i.e. separate acquisition and processing of data.

The loop responsible for obtaining data from the NIC and placing it in the "log queue" functions as follows: The relevant TCP/IP port is opened on the NIC. A frame is read and inserted into a queue of strings. The loop responsible for inserting data into the log queue operates as follows: The queue of strings is read and "unflattened" back into numeric data. This data is then queued into a queue of FXP array, and also plotted as U-curves. The loop responsible for logging the data to disc simply reads the queue of FXP arrays and writes this data to disc.

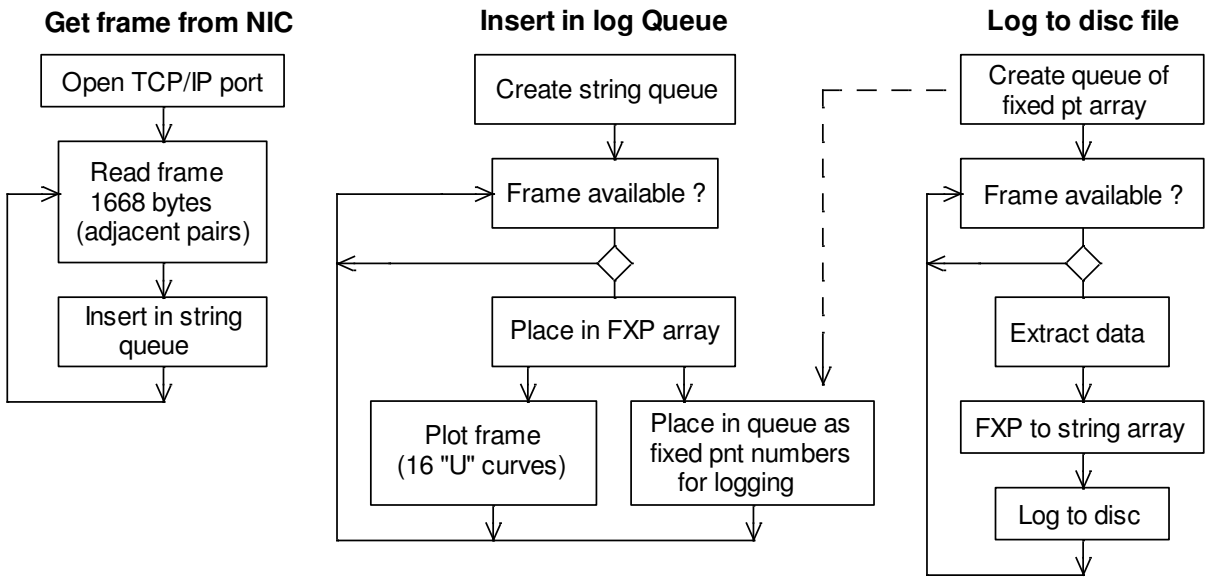

Fig. 6. PC code as implemented in three "while loops"

6. Data Sets

The multiplexer selection and current pulse timing pulses generated by the FPGA have been verified to be identical to those generated by the previous version of the system by examination using an oscilloscope. The details of these timing functions are available elsewhere, [1]. The un-calibrated data sets (typical example in Fig. 8) are therefore also identical to those previously obtained. These data sets are calibrated by scaling them to coincide with a data set generated by the forward model used in the reconstruction algorithm. The calibration constants (obtained from zero and maximum current readings) are applied to all subsequent raw data frames and then fed to the reconstruction algorithm.

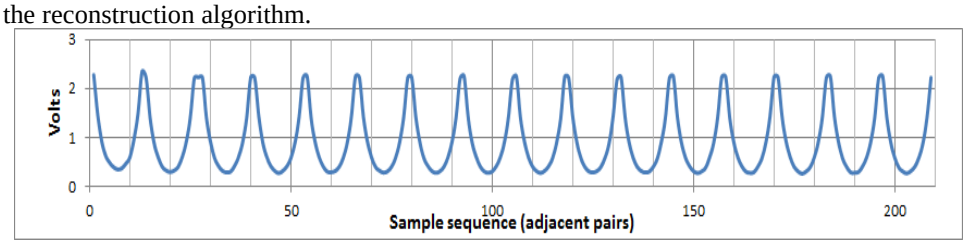

Figure 8. Typical "U curve" plot of un-calibrated data recorded by the system

6. Conclusions

The data acquisition system, based on an industry standard IO system, has been demonstrated to acquire frames from a 16 electrode system ERT system and transmit them to a PC via an TCP/IP network at 1000 frames/second on a 100Mb/s network. The LabVIEW code developed for this application has been placed in the public domain and complements the previously published ERT hardware design and provides a complete

solution to low cost ERT measurements. The LabVIEW approach allows researchers primarily involved in the application of ERT measurements to modify code for specific applications. The code structure lends it's self to adaption for real time feedback control.

The software is more robust than previous implementations as there are no operating systems overheads on the RIO. The instrument is also simpler to manufacture as the complete instrument has been reduced to two circuit boards plus the RIO. Software to control the instrument now resides directly in the RIO, rather than on a complex interaction of code installed on the GB60 MCU, USB DAQ and the embedded PC. The network linked operators computer is now primarily used for data storage, image reconstruction and analysis. On-line image reconstruction may be incorporated in the PC operating software and data recorded to disc files for processing using EIDORS, [8]. or external user written code. A 2D online reconstruction algorithm is currently under development and will be added to the open-source set of tools when available. The configuration described lends itself to feedback control either implemented on the RIO or the remote PC. All software for both the RIO and the PC has been developed using the National Instruments® LabVIEW® 2011 SP1 programming environment. The LabVIEW open source code could easily be modified for specific applications by the users. Replacing the sbRIO with a more advanced version of the RIO with a larger FPGA and more powerful processor would open up the possibility of local reconstruction and feedback control to implemented independently of the external PC.

Acknowledgements

Jacques Cilliers, of NI South Africa, for his advice on the optimization of data transmission procedures.

References

[1] Randall, E.W., Wilkinson, A.J., Duggin, K.E., Hauslaib, K.H., 2010, “An improved design for a Current Pulse Electrical Resistance Tomography System,., In the proceedings of 6th World Congress on Industrial Process Tomography, Beiging, China.

[2] Randall, E.W., Long, T.M., Salkinder, J., 2007, “System Design for the Control of a Tomography Instrument via an IP link”, In the proceedings of 3rd World Congress on Industrial Process Tomography Norway

[3] Byars, M., Pendleton, J.D., Goodeve, D., 2003. "A new High-Speed Control Interface for an Electrical Capacitance Tomography System", Proceedings of the 3rd World Congress on Industrial Process Tomography, Banff.

[4] Haselman, M., Miyaoka, T., Lewellen, S., Hauck, S., “FPGA-Based Data Acquisition System for a Positron Emission Tomography (PET) Scanner, ACM/SIGDA Symposium on Field-Programmable Gate Arrays, 2008.

[5] Sankowski, D., 2012, “Dual-mode ECT/ERT measurement system using NI PXI interface for multiphase control”, In the Proceedings of 6th International Symposium on Process Tomography, Cape Town.

[6] Wilkinson, A.J., Randall, E.W., Long, T.M., Collins, A, The Design of an ERT System for 3D Data Acquisition and a Quantative Evaluation of its Performance, MST/215961/FEA/6681, UCT, 4/5/2006

[7] Long, T.M., Wilkinson, A.J., Randall, E.W., Sutherland, A.P.N., 2007, ”An on online real time velocity profiling system using electrical resistance tomography”, In the proceedings of 5th World Congress on Industrial Process Tomography, Bergen, Norway.

[8] Polydorides, N.P., Lionheart, W.R.B., 2002, ‘A MATLAB toolkit for three dimensional electrical impedance tomography: A contribution to the EIDORS project’, Meas. Sci. Technol. 13, pp. 1871-1883.