Computer Physics Communications 182 (2011) 485–489

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

TimeSeriesStreaming.vi: LabVIEW program for reliable data streaming of large analog time series ☆

Fabian Czerwinski *, Lene B. Oddershede

Niels Bohr Institute, Blegdamsvej 17, 2100 Copenhagen, Denmark

A R T I C L E I N F O

Article history:
Received 2 July 2010
Received in revised form 10 August 2010
Accepted 16 October 2010
Available online 28 October 2010

Keywords:
Data acquisition
Data streaming
LabVIEW
TDMS
Optical tweezers

A B S T R A C T

With modern data acquisition devices that work fast and very precise, scientists often face the task of dealing with huge amounts of data. These need to be rapidly processed and stored onto a hard disk. We present a LabVIEW program which reliably streams analog time series of MHz sampling. Its run time has virtually no limitation. We explicitly show how to use the program to extract time series from two experiments: For a photodiode detection system that tracks the position of an optically trapped particle and for a measurement of ionic current through a glass capillary. The program is easy to use and versatile as the input can be any type of analog signal. Also, the data streaming software is simple, highly reliable, and can be easily customized to include, e.g., real-time power spectral analysis and Allan variance noise quantification.

Program summary

Program title: TimeSeriesStreaming.VI
Catalogue identifier: AEHT_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEHT_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 250
No. of bytes in distributed program, including test data, etc.: 63 259
Distribution format: tar.gz
Programming language: LabVIEW (http://www.ni.com/labview/)
Computer: Any machine running LabVIEW 8.6 or higher
Operating system: Windows XP and Windows 7
RAM: 60–360 Mbyte
Classification: 3
Nature of problem: For numerous scientific and engineering applications, it is highly desirable to have an efficient, reliable, and flexible program to perform data streaming of time series sampled with high frequencies and possibly for long time intervals. This type of data acquisition often produces very large amounts of data not easily streamed onto a computer hard disk using standard methods.
Solution method: This LabVIEW program is developed to directly stream any kind of time series onto a hard disk. Due to optimized timing and usage of computational resources, such as multicores and protocols for memory usage, this program provides extremely reliable data acquisition. In particular, the program is optimized to deal with large amounts of data, e.g., taken with high sampling frequencies and over long time intervals. The program can be easily customized for time series analyses.
Restrictions: Only tested in Windows-operating LabVIEW environments, must use TDMS format, acquisition cards must be LabVIEW compatible, driver DAQmx installed.
Running time: As desirable: microseconds to hours

1. Introduction

Precision experiments where data is acquired with high temporal resolution pose a challenge with respect to streaming and saving the data correctly onto a computer hard disk for further processing [1]. Within the nanoscience and biophysical communi-

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: czerwinski@nbi.dk (F. Czerwinski).
URL: http://www.nbi.dk/~czerwin (F. Czerwinski).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2010.10.019

ties, LabVIEW is often the program of choice for control of data acquisition and streaming [2–4]. Here, we present a highly reliable and efficient data streaming program in LabVIEW. The program is built into modular blocks with the goal of making the design comprehensible and easily compatible for further customization. Also, user-friendliness has been highly valued and we show how to use the program to stream time series data from two typical nanoscale experiments: One involving optical trapping assays [5], the other ionic current measurements through glass capillaries [6].

2. Program overview

2.1. Requirements

A time series can originate from a wide range of physical signals, such as temperature, voltage, current, etc. The time series, most often in the form of parallel voltage signals, enter the program through a number of channels of an acquisition card, building the interface between computers and experimental setups. Acquisition cards are available in a broad range for various tasks and quality requirements. We used National Instruments cards NI PCI-6251, NI PCIe-M6251, and NI PCI-M6040 [7]. They are coupled into TimeSeriesStreaming.vi by DAQmx, a LabVIEW-internal driver. As precondition, the acquisition card must be compatible with LabVIEW. This holds either for those that can be installed by National Instrument's *Measurement and Automation Explorer*, or for those supplied with a LabVIEW-compatible driver.

2.2. Main program

The main program is designed in a modular fashion to offer independent as well as interconnected control of different sources of analog signals. Further, it contains support for data-streaming protocols. The programming architecture combines horizontal modules (acquisition, queuing, streaming) with vertical programming patterns (sequential structure, parallel while loops, multicore processing) in order to assure negligible error rates and optional customization.

The different modules of the program are highlighted each by their background color in Figs. 1 and 2. The four modules deal with elements that concern *computer specifications* (yellow), *acquisition* (blue), *queuing* (red), and *streaming* (green). Each of the modules functions independently from the others as it communicates through well-defined programming patterns.

The usage of the program will be explained in Section 3.

2.2.1. Programming patterns

Multicore processing is the ability to distribute computational jobs over more than one core, i.e. one CPU. This feature has become available in recent versions of LabVIEW. In TimeSeriesStreaming.vi multicore processing is implemented by assigning each timed loop to a specific core. On the tested systems, the CPU load of an individual core never exceeded 20%. Optimal multicoring was ensured by core assignments (highlighted yellow in Fig. 2). It could also compensate for occasional interruptions by the Windows XP operating system.

Data acquisition must not be interrupted by waiting times during the streaming process. LabVIEW is optimized for *data flow control*. In TimeSeriesStreaming.vi this is achieved by transporting data packages between different loops exclusively through built-in queues (highlighted red in Fig. 2). The streaming loop is not executed when the queue is empty. This strategy has proven very powerful, as it allows both loops to run as quickly as possible without potential disturbance by waiting times.

Parallelizing allows for parallel execution of computational jobs. Data acquisition is done in one while loop, data streaming in a parallel loop. Very reliable *streaming* is achieved by the powerful data format TDMS (Technical Data Management Streaming, National Instruments). Using the primitive TDMS VIs allows for high performance streaming virtually with no limitation.

3. How to use the program

A hands-on introduction to TimeSeriesStreaming.vi is given in this section. The perspective user is guided through the modules in the program's front panel (Fig. 1). An experienced user could adjust the programming architecture in the block diagram at will (Fig. 2).

The only *computer specification* that must be set is the processor assignment (highlighted yellow). The user can choose to perform processor assignment on quad-, duo-, or single-cores; or simply choose automatic in which case the program will usually assign the highest ordered cores. However, if the user is aware, e.g., that the Windows operating system is utilizing certain cores, it might be beneficial to assign the cores manually.

The *acquisition* module (highlighted blue) controls mainly the settings regarding the acquisition card for a particular measurement. The desired scan rate [Hz] must be given in units of Hertz.

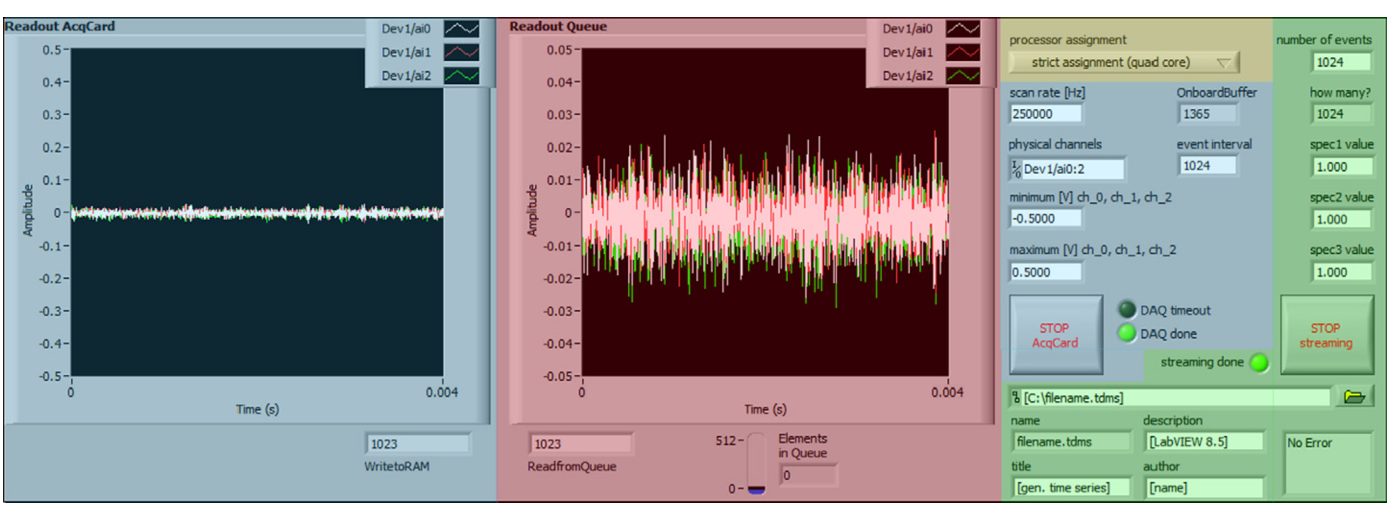

Fig. 1. Front panel of TimeSeriesStreaming.vi. In- and outputs of the four programming modules are highlighted by background color: *computer specifications* (yellow), *acquisition* (blue), *queuing* (red), and *streaming* (green). Details regarding the user defined settings are given in Section 3. A high-resolution version of Fig. 1 is available through http://www.nbi.dk/~czerwin/TimeSeriesStreaming.html. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

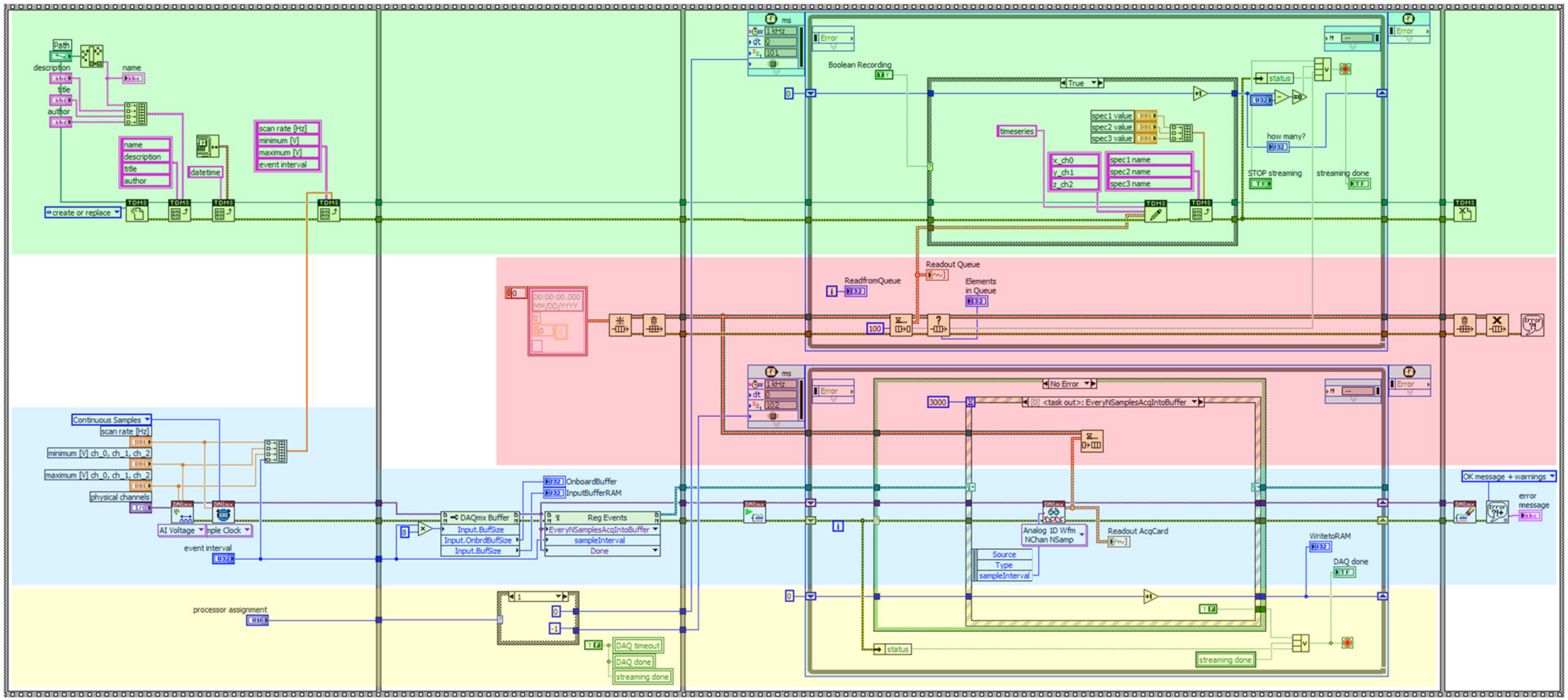

Fig. 2. Block diagram of TimeSeriesStreaming.vi. The four programming modules are highlighted by background color: *computer specifications* (yellow), *acquisition* (blue), *queuing* (red), and *streaming* (green). This diagram illustrates the modular architecture of the program where horizontal modules (acquisition, queuing, streaming) can be combined with vertical programming patterns (sequential structure, parallel while loops). A high-resolution version of Fig. 2 is available through http://www.nbi.dk/~czerwin/TimeSeriesStreaming.html. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Benchmark of TimeSeriesStreaming.vi as performed on a NI PCI-6251 connected to a Windows XP computer running LabVIEW 8.6. Each individual run took 4096 s. Scan rate, number of channels, and buffer per channel were set. Maximum filling, file size, and error rate were determined.

Scan rate (kHz)	Number of channels	Buffer per channel	Max filling of queue	TDMS file size (MB/s)	Error rate
1250	1	4096	0	9.6	<0.000
500	2	2048	0	7.7	<0.000
250	4	1024	0	15.3	0.004
100	3	1366	0	2.3	0.001
100	1	4096	2	2.3	0.001
22	6	683	5	6.0	0.003
22	3	1366	3	3.0	0.001
22	1	4096	3	1.0	0.001
10	1	4096	8	0.5	<0.001
1	1	4096	3	0.05	<0.000

The physical channels must be specified. `Dev1/ai0:2` denotes the signals input from Dev1 through the channels 0, 1, and 2 (e.g., x, y, and z coordinates of a recorded movement). The internal limits of recordable voltage signal minimum [V] and maximum [V] are set in order to optimize the resolution of the recorded time series. It is advisable to have these settings as close as possible to the extrema of the input time series, though, without cutting any of the data points.

Acquisition cards are equipped with an on-board buffer of a certain size indicated by OnboardBuffer. event interval sets the number of data points in each interval. If this number does not exceed the limit given by OnboardBuffer, the data are optimally passed onto the memory. Therefore, it is recommendable to try to keep event interval smaller than OnboardBuffer.

On left side of the front panel, a graph displays the output of the acquisition card in Volts for all specified channels. The graph Readout AcqCard shows the last package passed to the memory. The values of minimum [V] and maximum [V] shall be set as vertical axis limits. At any time, the acquisition can be aborted by hitting STOP AcqCard. This will halt the execution of looped functions of the DAQmx driver. The indicators DAQ timeout and DAQ done light up, if one of these two reasons terminates loop and therewith the program.

The graph Readout Queue (highlighted red) displays the last package of data passed through the *queue* to the streaming module. Utilizing a user-specified voltage interval here allows for an on-screen check, e.g., for whether data could be exposed to drift. The indicator Elements in Queue reveals the queue's filling level.

Streaming and data storage are controlled by the settings highlighted in green. With number of events the user specifies the total number of event intervals to be acquired. Hence, the total number of data points will be:

$$\text{total number of points} = \text{number of events} * \text{event interval}.$$

how many? counts the processed number of events. The entries spec1 value, spec2 value, and spec3 value are numerical values. They will be stored in the resulting TDMS file in the header to the recorded data. It can regard values the user wishes to keep with the data (e.g., laser power, particle dimensions, etc.).

The exact path for the TDMS file with the recorded data must be given. Therefore, replace `[C:\filename.tdms]` by an appropriate entry. Note that an already existing file will be overwritten by default. The program automatically returns the file name and stores it in the header of the TDMS file. title, description, and author can be defined by the user and are also stored in the file header.

If the total number of data points has been streamed, or if the STOP streaming button has been pressed, the program terminates and the indicator streaming done lights up. All data recorded up to that point is available in the user defined TDMS file. Also, TDMS files could be access already during streaming, e.g., from inside another VI.

Potential error messages are shown in the bottom right box.

4. Data format and benchmarks

4.1. Data format

The data recorded with TimeSeriesStreaming.vi is stored in a TDMS file. TDMS is an open-source file format developed by National Instruments. It is a binary format optimized for data streaming, thus, it handles dynamically increasing files correctly. Also, one can access the file already during acquisition. There are three ways to access TDMS: Directly in LabVIEW using the primitive TDMS VIs, by third-party plug-ins, or by the program *Diadem* (National Instruments). For the latter method, it is essential to use *Diadem* version 11.1 or higher. Detailed information on how to use Diadem or how to export TDMS files into third-party products can be found under http://www.ni.com/tdms. Third-party plug-ins exist, e.g., for *Matlab*, *OpenOffice*, and *Excel*. *Origin* can open TDMS files directly.

4.2. Benchmarks

We tested the performance and stability of TimeSeries Streaming.vi on three different computers designated for data acquisition. Their specifications are made available through the program summary URL [8]. Since LabVIEW only provides acceptable performance on machines operating Windows, we were limited by the choice of the operating system. All tested CPU architectures (quad-, duo-, and single-core) proved to yield very good acquisition. Though, the highest level of customizability applied to multicore processors. We tested the acquisition cards NI PCI-M6040, NI PCI-6521, and NI PCIe-6521. Table 1 represents the benchmarks for the NI PCI-6251 card with thermal noise as input. Scan rate, number of channels, and buffer per channel were set. Then the program ran for 4096 s and maximum filling and file size were determined. An error was counted when two consecutive 16-bit digits were exactly identical. The error rate was defined as number of errors divided by the number of data points.

In addition, we simulated basically all voltage acquisition devices with the *Measurement and Automation Explorer* and performed benchmarks with TimeSeriesStreaming.vi using the input from the simulated device.

5. Examples of program applications

Two brief examples of how to use TimeSeriesStreaming.vi are described here. In addition, the program has already been used to reliably stream large time series from experiments involving optical trapping of micron-sized polystyrene spheres [5], gold nanorods [9], and quantum dots [10]. Furthermore, we implemented an improved calibration protocol for optical tweezers that made use of the main programming features introduced here [11].

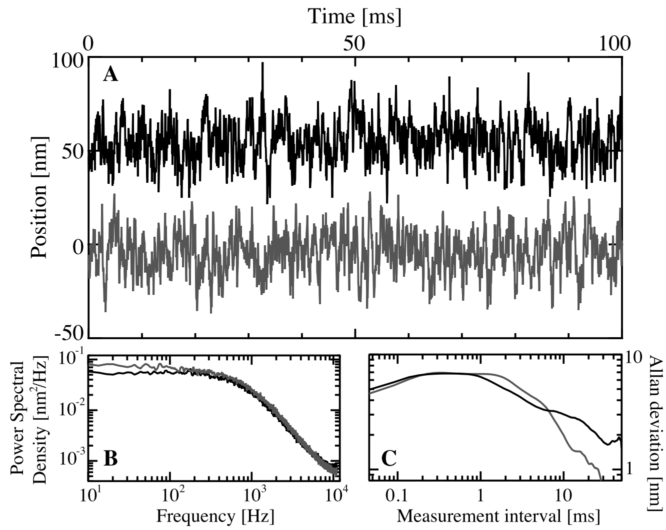

Fig. 3. Example of data acquisition and time series analyses applied to an optical trapping assay. (A) Positional time series of an 800 nm polystyrene sphere which is harmonically trapped. Its lateral positions *x* (grey) and *y* (black, offset 50 nm) are recorded by a photodiode detection system. The power spectral density (B) and the Allan deviation (C) of the lateral positions visited.

5.1. Position recording in optical tweezers

The development of TIMESERIESSTREAMING.VI was prompted by a need to stream large amounts of data from optical trapping assay to a hard disk. The goal was to analyze the noise by means of accuracy measurements [5]. Therefore, positions of a trapped microsphere were recorded at sampling frequencies of up to 100 kHz in the order of hours. A short time series is plotted in Fig. 3(A).

Experimental details are provided in Ref. [5]. The positions were sampled using a photodiode detection system yielding an output in Volts, which were reformulated in terms of metric distances by a calibration factor. Fig. 3(B) shows the positional power spectrum, which, when properly analyzed, gives information about the calibration factor as well as the strength of the optical trap [12]. A different type of time series analysis, Allan variance analysis, is excellent for quantifying noise in optical trapping assays [5,11], in particular in the low frequency regime, which is not possible through normal variance or power spectral analysis. Fig. 3(C) shows the Allan deviation of the same trace quantifying the exact accuracy for various measurement intervals. For this type of analysis it is crucial to have long overall measurement time series and reliable streaming of the data onto the hard disk, a requirement met by TIMESERIESSTREAMING.VI. The modular fashion of the program enables straight forward implementation of similar types of calibration or noise quantification routines.

5.2. Ionic current through glass capillary

The translocation of molecules through solid-state nanopores has drawn a lot of attention in recent years due to the enormous potential they hold for parallel screening of biomolecular solutions [13]. Also, glass capillaries with a diameter of 60 nm could be used to detect DNA folding [6]. Here, we used TIMESERIESSTREAMING.VI to stream the ionic current measured onto the computer hard disk. A sketch of the experiment is shown in the lower right of Fig. 4. The measured current values are plotted, and in the upper left there is a zoom-in to illustrate that data acquisition was done at a very high rate (1.25 MHz). In this experiment, the error rate was zero. Hence, time series can be analyzed for events happening within sub-milliseconds, the timescale relevant for protein translocation through a nanopore [14], or for events happening on the order of minutes, a typical timescale for drift, as also visible in Fig. 4.

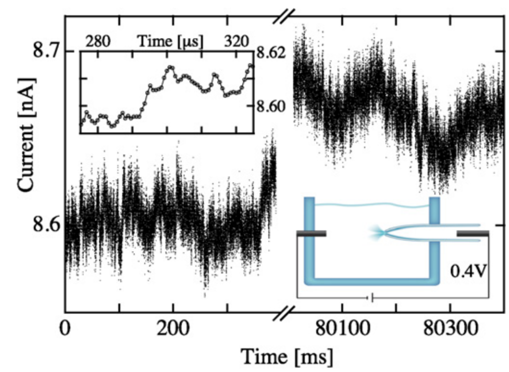

Fig. 4. Time series from measurements of monovalent-ionic current through a glass capillary (diameter 60 nm), a sketch of the experiment is shown in the lower right. The inset in the upper left is a zoom-in on the time axis, showing that the acquisition card's limit of 1.25 MHz sampling can be achieved.

6. Summary

We developed a LabVIEW program that reliably streams large amounts of data correctly onto a computer hard disk. The program was checked on several individual platforms and showed to perform with a very small error rate. The program is made in a modular fashion with the aim of making it user-friendly and easily customizable. As an example, we showed how to acquire the positions of an optically trapped sphere and how this time series data can be further analyzed. We also demonstrated how the program streamed time series of ionic current measurements. As the program easily handles a broad range of analog inputs, there is a wide range of applications, particularly in biophysical nano-scale experiments as pointed out in the two examples. The source code is freely available at through the CPC Program Library and under the standard CPC license agreement [15].

Acknowledgements

We thank Lorenz Steinbock for data acquisition and help on the ionic current measurements, and Oliver Otto as well as Jesper Tholstrup for comments on the manuscript. Also, we acknowledge financial support through the KU excellence program.

References

[1] K.C. Neuman, S.M. Block, Rev. Sci. Instrum. 75 (2004) 2787.
[2] T.M. Hansen, S.N.S. Reihani, L.B. Oddershede, M.A. Sørensen, Proc. Natl. Acad. Sci. 104 (2007) 5830.
[3] J. van Mameren, K.C. Vermeulen, F. Gittes, C.F. Schmidt, J. Phys. Chem. B 113 (2009) 3837.
[4] M. Mahamdeh, E. Schäffer, Opt. Express 17 (2009) 17190.
[5] F. Czerwinski, A.C. Richardson, L.B. Oddershede, Opt. Express 17 (2009) 13255.
[6] L.J. Steinbock, O. Otto, C. Chimerel, J.L. Gornall, U.F. Keyser, Nano Lett. 10 (2010) 2493.
[7] Specifications of National Instrument's acquisition cards are available through, http://www.ni.com/dataacquisition/.
[8] Computer specifications during benchmarks are made available through the program summary URL, http://www.nbi.dk/~czerwin/TimeSeriesStreaming.html.
[9] F. Czerwinski, A.C. Richardson, C. Selhuber-Unkel, L.B. Oddershede, Proc. SPIE 7400 (2009) 740004.
[10] L. Jauffred, M. Sletmoen, F. Czerwinski, L.B. Oddershede, Proc. SPIE 7762 (2010) 776226.
[11] M. Andersson, F. Czerwinski, L.B. Oddershede, J. Opt. (2010), in press.
[12] K. Berg-Sørensen, H. Flyvbjerg, Rev. Sci. Instrum. 75 (2004) 594.
[13] C. Dekker, Nature Nanotech. 2 (2007) 209.
[14] S.W. Kowalczyk, A.R. Hall, C. Dekker, Nano Lett. 10 (2010) 324.
[15] The standard CPC license is posted at http://cpc.cs.qub.ac.uk/licence/licence.html.