Journal of Computer and Communications, 2018, 6, 1-13
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Development of a Measurement Software for the Characterization of WORM Devices for Novel Memory Storage Applications

Mirko Congiu[1*], Miguel H. Boratto[1], Paride Pica[2], Carlos F. O. Graeff[1,3]

[1]POSMAT—Post-Graduate Program in Materials Science and Technology, School of Sciences, UNESP—São Paulo State University, Bauru, SP, Brazil
[2]Department of Chemistry-Piazzale Aldo Moro, University of Rome "La Sapienza", Rome, RM, Italy
[3]Department of Physics, School of Sciences, UNESP—São Paulo State University, Bauru, SP, Brazil
Email: *mirko.congiu@unesp.br

How to cite this paper: Congiu, M., Boratto, M.H., Pica, P. and Graeff, C.F.O. (2018) Development of a Measurement Software for the Characterization of WORM Devices for Novel Memory Storage Applications. *Journal of Computer and Communications*, **6**, 1-13.
https://doi.org/10.4236/jcc.2018.69001

Received: August 14, 2018
Accepted: September 1, 2018
Published: September 4, 2018

Abstract

We hereby propose a software solution to perform high quality electrical measurements for the characterization of WORM (write-once read many), a new generation memory device which is being intensively studied for non-volatile data storage. The as-proposed software is completely based on .NET framework and sample C# code. The paper performed a relevant measurement based on this software. Working WORM devices, based on a polymeric matrix embedded with gold and copper sulfide nanoparticles, have been used for test measurements. The aim of this paper is to show the main steps to develop a fully working measurement software without using any expensive dedicated software.

Keywords

WORM, Memristor, Electrical Measurements, Memory Devices

1. Introduction

One of the new challenges of modern technology is the improvement of the data-storage devices. The progress of silicon-based devices is approaching to its physical limit (~10 nm) [1]. In this respect, new incoming technologies are being studied all over the world such as memristors (MEMs) for non-volatile memories and write-once read-many (WORM) for non-volatile memory storage. MEMs were described by Chua *et al.* in 1971 "the missing element" in the circuit's theory, filling the missing relationship between the charge $Q(t)$ and the current flux $f(t)$ [2]. Until it's realization in 2008 by the HP labs [3], MEMs have

remained as a theoretical object. Belonging to MEMs, resistive switching memories (RRAMs) represent an interesting and multidisciplinary topic of modern scientific and technologic research. For instance, RRAMs may found application in different fields of technology such as memory storage [4] [5], computing and logic operations [6] [7], neuromorphic computing applications [8] [9] and analog circuits [10]. Basically, a RRAM is can be described as a resistive element whose resistance can be switched by means of a voltage pulse [3]. Normally, a RRAM can exist in two states: the high resistance state (HRS) and the low resistance state (LRS). The value of the resistances of the above-mentioned states can be defined as R_{ON} and R_{OFF}. This kind of elementary cell-unit can operate as a Boolean logic switch returning (0) if the state returns R_{OFF} value and (1) when R_{ON} [6]. As the RRAM is a reversible state device, the information is stored over time as a non-volatile memory similarly to the widespread used flash memories. Among with MEMs, WORM (write-once read-many) represents another class of modern devices, which are being intensively studied nowadays. The latter, differently from MEMs, is a non-volatile memory (read only ROM). The purpose of a non-volatile memory is to keep the information and ensure that the latter can be read many times [11] [12]. Basically, a WORM is a two-terminal device with an insulating film placed between two metallic electrodes. The information can be written by applying a voltage pulse which induces the formation of a stable conductive bridge (short-circuit) between the two electrodes. So the device, at the initial state, shown a huge value of resistance and, after the writing process, the resistance become very low due to the short-circuit formation [13]. In a similar way to MEMs, WORMs have two resistance states, one with high resistance and the other one with a low resistance allowing to store analogic values (0 - 1). The characterization of a WORM requires some fundamental steps which require a collection of precise and reliable set of methods. One of the first operations is the cyclic voltammetry measurement (CV), in which the current flowing through the devise is measured as a function of a cyclic voltage-sweep between two limit voltages. This characterization allows seeing the resistance transitions happening into the device due to the application of an electronic field. Subsequently, the device should be studied in pulsed-voltage regime, to verify that the short-circuit (data writing) can be performed using square pulses. This last analysis should be done by trying successive pulses of different intensity and duration sweeping the voltage in a well-defined range, defined through CV scans. One of the main problems in beginning a research project on MEMs and WORMS is the lack of dedicated measurement software. In fact, most of the research group must spend a lot of time to build its own analysis tools. A very common solution is the use of commercial visual programming software such as VEE (Keysight) or LabView (National Instruments™). The latest combine a visual-programming interface to directly access to the main function of measurement instruments. Both allow the creation of measuring application with data-saving and analysis built-in functions along with other professional stuffs. However, the choice of this kind of solutions requires a large initial money investment due to the software license

purchasing and the multi-level training courses to learn the visual-programming language. In fact, these softwares work with a graphical-dataflow interface which simplifies the programming operations, however it does not exempt the research staff from learning the basic mechanisms of programming (loops, arrays, variables, data handling etc.). Another limitation of these softwares is the impossibility of build a stand-alone software, executable on a generic computer. Both the mentioned software allows to build executable files which look-like a stand-alone-programs however the latest require a specific run-time environment installed on the hosting computer or a higher and more expensive software license (e.g. LabView pro). The as-mentioned commercial solutions are reliable and offer a wide range of programming tools and can be chosen after a careful consideration of advantages, disadvantages and costs involved. In this paper we want to show another programming approach, which uses freeware resources to build a powerful stand-alone software based on two freeware solutions: Microsoft Visual Studio and .NET environment, programming with the widespread C# language. Our approach requires a little bit higher effort in the programming phase however it is free, reliable and can be considered to build your own software, designed taking into consideration the experimental necessities and executable on a generic computer requiring just the measurement instruments drivers (normally distributed for free). Especially, we want to show how to wrap the original driver of a commonly used source-meter equipment family (Keithley 24xx). The proposed software has been tested on real WORM samples.

2. Methods

The software was developed using Microsoft Visual Studio Community 2017 (V. 15.7.2), keithley 2400C Source Meter (SM) was used as the measurement interface using the driver Ke24xx, compatible with all the 24xx instrument series. The connection between the SM and the computer was performed using a USB/GPIB interface 82357B by Agilent Technologies®. Real WORM samples were prepared using a modified method from our research group, based on copper sulfide nanocrystals [14] dispersed in a polymethylmethacrylate (PMMA) matrix embedded with gold nanoparticles capped with biphenyl groups [15]. As the device preparation method is beyond the scope of the paper and is under further investigation it will not be discussed here.

2.1. Software Design

The first step of the development of this solution is the planning of the fundamental elements necessary to perform a complete characterization of the memory device. As discussed in the previous paragraph, the initial characterization should be based on a sweep-voltage analysis. This tool is extremely powerful because it allows seeing the switching process induced by the application of a sweeping voltage. In physical terms, a CV consists of the generation of a triangular wave with a defined amplitude (voltage limits), a certain frequency and a

duration. During the application of this voltage wave, the equipment performs repeated measurements of the flowing current and stored the data. Several SM such as Keithley, have a built-in sweep function which can be accessed from the frontal panel or from remote (driver). This function allows defining the voltage limits, the number of points and the measurement speed. However, this kind of sweep cannot show real-time results because the data are sent to the remote unit (PC) just at the end of the sweep. In this kind of measurement, the voltage is increased linearly as a function of the time and the current is samples as fixed intervals, once the voltage reaches one of the limit value, the direction of the sweep is changed until the next limit voltage and the operation can be repeated in cyclic regimen. Another approach to perform current *vs* voltage (IV) measurement is the staircase voltammetry. In this technique, the voltage is increased in a series of steps just like a stair. Each step-voltage is maintained over a certain time and the current is sampled before the next step [16] [17]. In most SMs this function can be implemented just by remote operation (software).

The entire program can be schematized as shown in **Figure 1**. A generic measurement program should have a module containing a graphic user interface (GUI, **Figure A1**) in which the user inserts the experimental parameters such as: the desired function, the values of voltages, compliance current (maximum allowed current), the number of points and all the necessary information. These parameters should be checked by a function which verifies the compliance of the input values in accordance to the instrument specification, to avoid problems during the measurement. When the check is completed, and the experiment is validated, the program calls methods from the external dynamic-link library (dll) of the instrument, *i.e.* the driver, to start the communication and run the

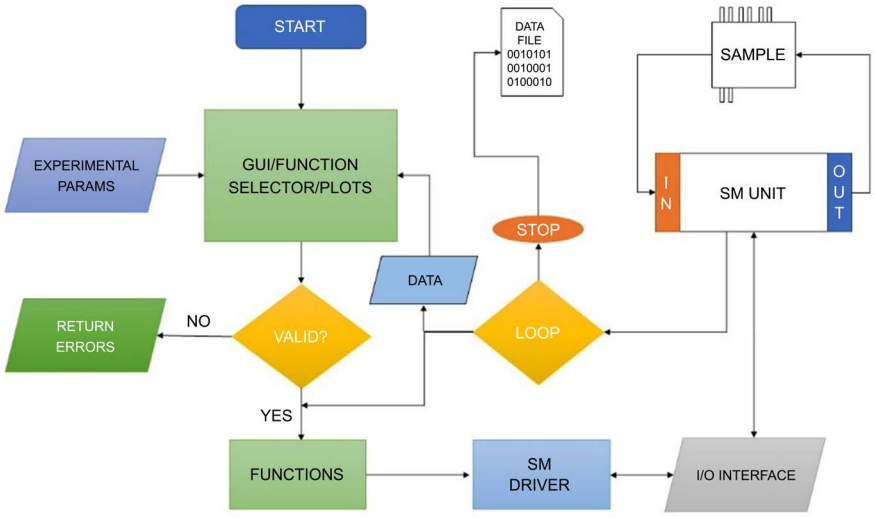


Figure 1. Flowchart representation of the measurement software. The experimental parameters are defined by the user through the GUI and, after a validity check, the selected function is executed by the driver-wrapper module built-in in the software. The I/O interface sends the information to the SM and the measurements are read and visualized in the GUI. At the end of the loop the measurements are stored into a data file.

selected functions. After the realization of the measurements, the software stores the data into a datafile.

2.2. Driver Wrapping

The core element of the as proposed software is the driver communication module. The driver of the equipment can be represented as a collection of functions which can be called by an external application. In our case, the first step is to reference the specific dll in the C# project and create the instrument as an object specifying the virtual software application architecture (VISA) address (e.g. Ke24xx instr = new Ke24xx ("GPIB0::15::INSTR", false, true)). The two bool values, following the VISA are respectively the option which resets the equipment and the instrument handling to start a communication session. More information about the parameters requested in the method can be found on the instrument driver manual. When the object instrument is declared into the software, all the front-panel function can be called from the main class Ke24xx. The following lines represent the commands used to set the equipment in the voltage source mode:

```
Ke24xx instr = new Ke24xx("GPIB0::15::INSTR", false, true);
instr.reset();//reset the equipment
instr.ConfigureSourceMode(Ke24xxConstants.SourceVoltageFunction,
Ke24xxConstants.SourceFixedMode, Ke24xxConstants.SourceDcShape,
Ke24xxConstants.AutoRange);
```

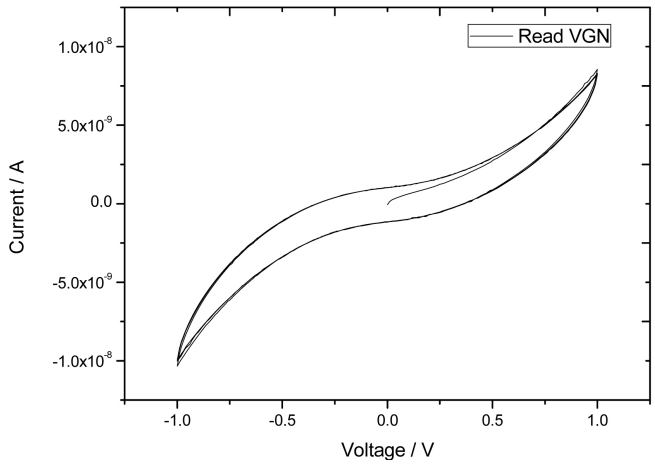
In this mode, the equipment works as a voltage source and is ready to apply a voltage and measure a current in a specified current range or in "Auto Range" mode. The voltage (V, double) can be applied to the sample after enabling the digital output:

```
instr.EnableSourceOutput(true);
instr.ConfigureSourceLevel(Ke24xxConstants.SourceVoltageFunction, V, 0);
```

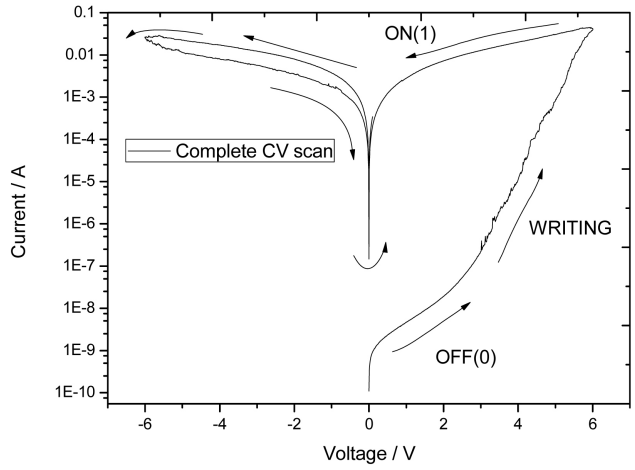
These functions can be included in a generic loop which increases the applied voltage in the selected range by a specific step and time. More source code examples can be found into the supplementary information (SI).

2.3. WORM and RRAMs Measurement Methods

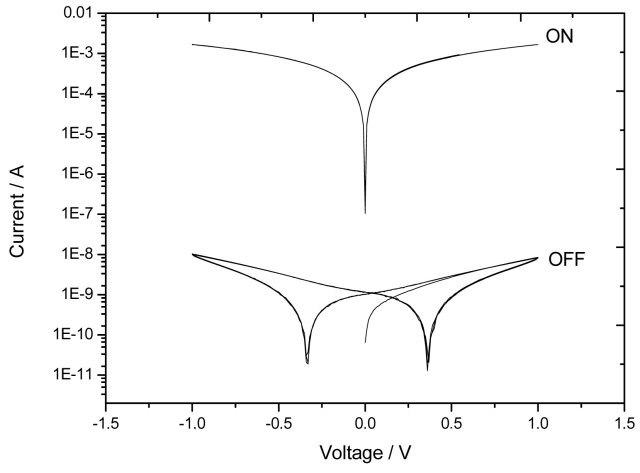
As discussed in the introduction, the first characteristic to be investigated is the resistance-change transition to identify our device as a WORM or a RRAM (if the transition is reversible). So, the first measurement should be a staircase CV or a sweep between two defined voltages. The following **Figure 2** shows an example of a resistance transition in a real WORM sample. In **Figure 2(a)** one can see the current vs voltage response of a virgin device, showing a shape characteristic of a capacitor. When the voltage is sweep between 1.0 and −1.0 V no sharp transitions are visible in the graph. This confirms that the device is stable under the applied bias, thus the reading voltage can be chosen into the selected range. To induce the resistance transition (writing process), we have extended the scanning range from −6.0 to 6.0 V, as shown in **Figure 2(b)**.



(a)



(b)



(c)

Figure 2. Cyclic voltammetry of a typical WORM based on PMMA/CuS/AuBi blend. The scans were performed from 0.0 V up to the voltage limits defined for the experiment: ±1.0 V for the reading measurements (a and c) and ±6.0 for the writing operation (b). First, the virgin (VGN) device was characterized in the reading range showing a capacitive behavior (a). When the voltage sweep is extended (b) a sharp increase in the flowing current sets the device in its ON state, which results stable over the rest of the scan.

In the region between 3.0 and 4.8 V, the current shows a higher noise as well as a sharp increase due to the formation of the short-circuit which brings the device from it OFF state to the ON state. In binary terms this transition can be understood as the writing of the bit "1" replacing the bit "0". Once the device is set to its ON state which shown a current ratio (ON/OFF) of about 10^3 considering the current of the virgin (VGN) device (**Figure 2(c)**). After the wiring operation, the device shows the typical CV response of a resistor without any capacitive effect. This can be due to the breaking of the insulating layer.

From CV characterization it is possible to identify the resistance switching behavior of the device under investigation. From this graph it is possible to identify the voltages in which such transition occurs. This is a key point of our characterization method. Notice that, for direct informatics applications, the switching operations are performed by pulsed voltages (square wave). For this reason, the next measurement module was designed to perform voltage pulses in a specific voltage range, to determine the switching voltage, the pulse duration and the minimum number of pulses to switch the device ON [18] [19]. **Figure 3** represents a schematic view of the pulsed voltage measurement module (PVMM) proposed in this software. To perform this measurement, the user defines some fundamental parameters such as: the limit voltages (V1 and V2); the number of points in which the range will be split; the reading voltage (V_R) in which the sample is read; the number of readings (N_R), the number of pulses (N), pulses duration (T_1) and the delay between each pulse (T_2).

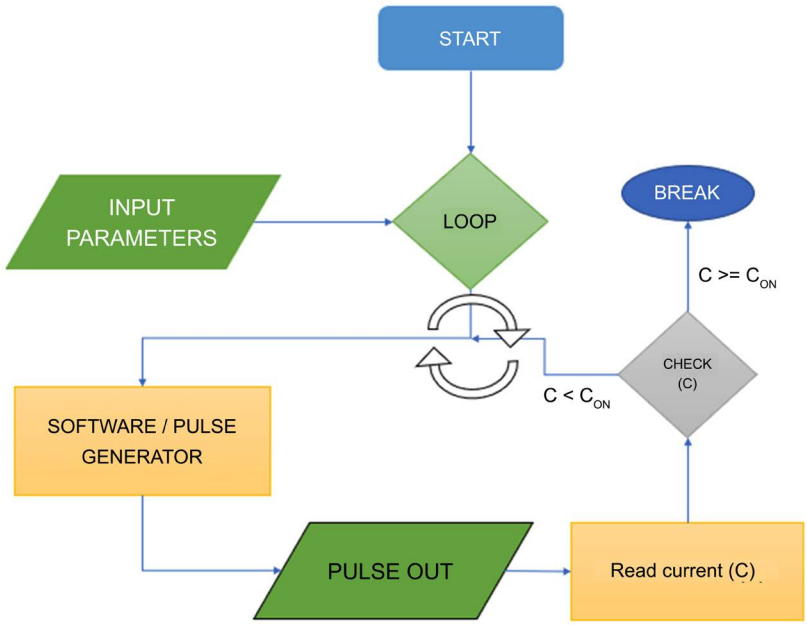


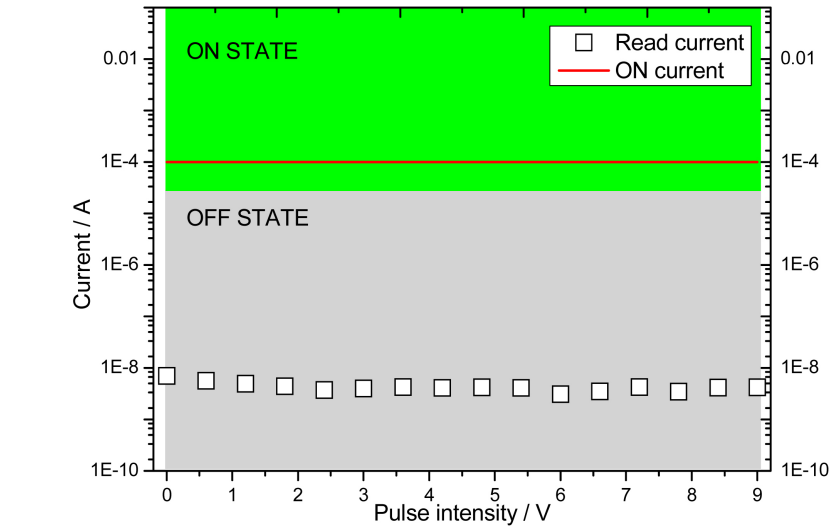
Figure 3. Flowchart representation of the pulse voltage measurement module. This function applies square voltage pulses to the sample. The pulse voltage is increased by a fixed value and is ranged into a specific voltage range. After the application of the pulses, the software read the current (C) value and check if it is higher than a fixed current value which identifies the ON state.

The voltage (intensity) of the square pulse is ranged between the values V1 and V2. Then, the software performs a current measurement and acquires this value storing it into a variable (C). This value is then compared with the expected current's value of the ON state of the device (C_{ON}). If the read value C is lower than C_{ON} the voltage is increased and a new cycle is started; else the loop is stopped and the last applied voltage is defined as the ON switching voltage. Also, this function allows optimizing the number of pulses and the duration. The core of this function, in a typical C# namespace, should consist in a loop (for) defining the number (i) of repeated iterations corresponding to the desired number of voltage points. In each cycle the software applies 0.0 V during a certain delay time, then applies the voltage during the pulse duration time and finally applies 0.0 V again. Notice that, if one needs to apply multiple pulses, the function must be structured as a loop (voltage points) of loops (multiple pulses). At the end of the pulses, the reading voltage is set, and the current is sampled, and its average value is plotted in a graph (current *vs* pulse voltage). The example code can be found in **Appendix A**. After the measurement of the current, the variable corresponding to the applied voltage is increased by a fixed value and the cycle is repeated. The results of a typical switching experiment are reported in **Figure 4**. In this example, the pulse intensity was investigated between 0 and 9 volts using two different pulse duration times: 30 and 1000 ms.

In **Figure 4(a)**, square dots represent the average current values (n = 10) measured after the application of 1 pulse with an intensity ranging from 0.0 to 9.0 V with a duration of 30 ms. As shown by the plot, these experimental condition does not induce the OFF to ON transition. In fact, in the whole voltage range, the current remains below C_{ON}. Notice that it should be useful to introduce a tolerance threshold instead of precisely define C_{ON} as a strict barrier between the OFF and the ON state (gray and green regions). In the successive measurement, the dwell time of the pulse was extended to 300 ms (**Figure 4(b)**) and, after 4.0 V, the device shown the irreversible switch transition maintaining the ON state at higher voltages.

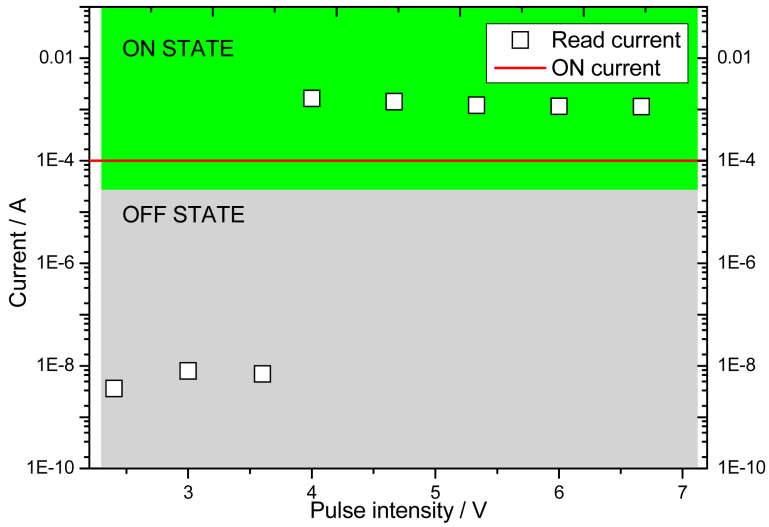
3. Conclusion

We proposed a software structure based on .NET framework, C# language and freeware resources for the characterization of resistance switch devices. In this work we have shown the main functions of the original driver of keithley measurement instruments (Ke24xx) but the same concepts could be traduced to work with other equivalent equipment. Besides the illustration of the main software procedure we have proposed a measurement routine for the characterization of WORM devices including both current *vs* voltage curves and square voltage pulse characterization. Our initial results demonstrate that the use of expensive software suites for visual virtual instrument programming could not be compulsory for the realization of an efficient measurement program. A basic knowledge of C# and Visual Studio allows the user to directly perform a link



Pulse duration = 30 ms; number of pulses = 1; Reading voltage = 0.3 V

(a)



Pulse duration = 1000 ms; number of pulses = 1; Reading voltage = 0.3 V

(b)

Figure 4. Square pulses measurements performed on WORM samples using the PVM module of the software. The current is sampled ten times after the application of the square pulse. Different dwell time was studied: 30 ms (a) and 300 ms (b). Notice that, when the dwell time is extended up to 300 ms the device is set in its ON state between 3.5 and 4.0 V.

between the software and the equipment using the original instrument driver as an external reference.

Acknowledgements

This work was supported by FAPESP (proc: 2013/07396-7); PNPD/CAPES, CAPES 024/2012 Pro-equipamentos; Department of Chemistry, University of Rome Sapienza and Project "TornoSubito 2017" supported by Lazio Region of

Italy, with FSE (FondoSocialeEuropeo, ID code: 9266). *In memory of Congiu Massimo* (1966-2018).

Author Contributions

1) Dr. Mirko Congiu: Software development, debugging and testing on real WORMs samples.

2) Dr. Miguel Henrique Boratto and Dr. Paride Pica: Data analysis and sample preparation.

3) Dr. Prof. Carlos Frederico De Oliveira Graeff: Coordination of the research group activities, discussion of the obtained results.

All the authors contributed to the writing of the paper.

Conflicts of Interest

No conflict of interest was reported by the authors.

References

[1] Tang, S., Tesler, F., Marlasca, F.G., Levy, P., Dobrosavljevic, V. and Rozenberg, M. (2016) Shock Waves and Commutation Speed of Memristors. *Physical Review X*, **6**, Article ID: 011028. https://doi.org/10.1103/PhysRevX.6.011028

[2] Chua, L. (1971) Memristor—The Missing Circuit Element. *IEEE Transactions on Circuit Theory*, **18**, 507-519.

[3] Strukov, D.B., Snider, G.S., Stewart, D.R. and Williams, R.S. (2008) The Missing Memristor Found. *Nature*, **453**, 80-83. https://doi.org/10.1038/nature06932

[4] Waser, R. and Aono, M. (2007) Nanoionics-Based Resistive Switching Memories. *Nature Materials*, **6**, 833-840. https://doi.org/10.1038/nmat2023

[5] Yang, Y., Choi, S. and Lu, W. (2013) Oxide Heterostructure Resistive Memory. *Nano Letters*, **13**, 2908-2915. https://doi.org/10.1021/nl401287w

[6] Borghetti, J., Snider, G.S., Kuekes, P.J., Yang, J.J., Stewart, D.R. and Williams, R.S. (2010) "Memristive" Switches Enable "Stateful" Logic Operations via Material Implication. *Nature*, **464**, 873-876. https://doi.org/10.1038/nature08940

[7] Strukov, D.B. and Likharev, K.K. (2005) CMOL FPGA: A Reconfigurable Architecture for Hybrid Digital Circuits with Two-Terminal Nanodevices. *Nanotechnology*, **16**, 888-900. https://doi.org/10.1088/0957-4484/16/6/045

[8] Ohno, T., Hasegawa, T., Tsuruoka, T., Terabe, K., Gimzewski, J.K. and Aono, M. (2011) Short-Term Plasticity and Long-Term Potentiation Mimicked in Single Inorganic Synapses. *Nature Materials*, **10**, 591-595. https://doi.org/10.1038/nmat3054

[9] Pickett, M.D., Medeiros-Ribeiro, G. and Williams, R.S. (2013) A Scalable Neuristor Built with Mott Memristors. *Nature Materials*, **12**, 114-117. https://doi.org/10.1038/nmat3510

[10] Driscoll, T., Quinn, J., Klein, S., *et al.* (2010) Memristive Adaptive Filters. *Applied Physics Letters*, **97**, Article ID: 093502. https://doi.org/10.1063/1.3485060

[11] Möller, S., Perlov, C., Jackson, W., Taussig, C. and Forrest, S.R. (2003) A Polymer/Semiconductor Write-Once Read-Many-Times Memory. *Nature*, **426**, 166-169. https://doi.org/10.1038/nature02070

[12] Fang, J., Wang, Q., Yue, X., Wang, G. and Jiang, Z. (2015) A WORM Type Polymer Electrical Memory Based on Polyethersulfone with Carbazole Derivatives. *High*

Performance Polymers, **28**, 1183-1191. https://doi.org/10.1177/0954008315621122

[13] Smith, S. and Forrest, S.R. (2004) A Low Switching Voltage Organic-on-Inorganic Heterojunction Memory Element Utilizing a Conductive Polymer Fuse on a Doped Silicon Substrate. *Applied Physics Letters*, **84**, 5019. https://doi.org/10.1063/1.1763632

[14] Congiu, M., Albano, L.G.S., Nunes-Neto, O. and Graeff, C.F.O. (2016) Printable ReRAM Devices Based on the Non-Stoichiometric Junction CuS/Cu_{2-x} S. *Electronics Letters*, **52**, 1871. https://doi.org/10.1049/el.2016.2901

[15] Fontana, L., Bassetti, M., Battocchio, C., Venditti, I. and Fratoddi, I. (2017) Synthesis of Gold and Silver Nanoparticles Functionalized with Organic Dithiols. *Colloids Colloids and Surfaces A*: *Physicochemical and Engineering Aspects*, **532**, 282-289. https://doi.org/10.1016/j.colsurfa.2017.05.005

[16] Christie, J.H. and Lingane, P.J. (1965) Theory of Staircase Voltammetry. *Journal of Electroanalytical Chemistry* (1959), **10**, 176-182. https://doi.org/10.1016/0022-0728(65)85021-5

[17] Hai, B., Tolmachev, Y.V., Loparo, K.A., Zanelli, C. and Scherson, D. (2011) Cyclic versus Staircase Voltammetry in Electrocatalysis: Theoretical Aspects. *Journal of The Electrochemical Society*, **158**, F15-F19. https://doi.org/10.1149/1.3512914

[18] Liu, C.Y., Shih, Y.R. and Huang, S.J. (2013) Unipolar Resistive Switching in a Transparent ITO/SiO_x/ITO Sandwich Fabricated at Room Temperature. *Solid State Communications*, **159**, 13-17. https://doi.org/10.1016/j.ssc.2013.01.008

[19] Whitcher, T.J., Woon, K.L., Wong, W.S., *et al*. (2016) Interfacial Behavior of Resistive Switching in ITO-PVK-Al WORM Memory Devices. *Journal of Physics D*: *Applied Physics*, **49**, Article ID: 075104. https://doi.org/10.1088/0022-3727/49/7/075104

Appendix A

The following C# code belongs to the pulse voltage measurement module of the main program (Async Void). Notice that just the core functions are reported without mentioning other code lines such as the variables declaration. The core consists of a loop in which the software applies the square wave pulse(s) and performs a current measurement and increase the variable cVolt which represents the voltage to be applied in the next cycle. The initial value of cVolt will be equal to the lower limit of the selected voltage range and will be increased (cVolt += step) up to the upper limit of the voltage range. The variables onDelay and onDuration represents respectively the delay before the voltage application and the pulse duration while the variable millis represents the overall experiment time, which could be also substituted by a timer object (System.Timers.Timer). The measurement of the current is performed by performing 10 measurements to obtain an average value. For this reason, at the beginning of the cycle the variable "currents" is declared as 1D array. To perform the measurement, the Read() method can be called specifying which value is requested (Ke24xxConstants.ReadCurrent), an out integer (numread) to count the number of the acquisition (in case of multiple read) and a 1D array in which the data will be dumped (readBuffer). The data are plotted directly into a chart object (chart 1).

```
for(inti = 0; i<= numberOfVoltagePoints; i++)//loop of voltage points
                        {
double[] currents=newdouble[10];
//apply 0 volts
instr.ConfigureSourceLevel(Ke24xxConstants.SourceVoltageFunction, 0, 0);
for (inti = 0; i<numberOfPulses); i++)//loop of multiple pulses
                                {
awaitTask.Delay(onDelay); //delay before the pulse
millis += onDelay; //increase the time count variable
instr.ConfigureSourceLevel(Ke24xxConstants.SourceVoltageFunction, cVolt, 0);
awaitTask.Delay(onDuration);
millis += onDuration;
//apply 0 volts agai-
ninstr.ConfigureSourceLevel(Ke24xxConstants.SourceVoltageFunction, 0, 0);
                                }
//reading, 10 measurements of current
for (inti = 0; i< 10; i++)
                                {
double[] readBuffer = newdouble[20];
instr.ConfigureSourceLevel(Ke24xxConstants.SourceVoltageFunction, read-
Volt, 0);
awaitTask.Delay(90);
instr.Read(Ke24xxConstants.ReadCurrent, outnumread,readBuffer);
```

```
    awaitTask.Delay(10);
instr.ConfigureSourceLevel(Ke24xxConstants.SourceVoltageFunction, 0, 0);
                                        chart1.Series[0].Points.AddXY(millis, readBuf-
fer[0]);
                                        chart1.Series[2].Points.AddXY(millis, cVolt);
                                        currents[i] = readBuffer[0];//dump the value
    millis += 100
                    }
    //calculate the average value of the current
    doubleaverage = (currents.Sum() / currents.Count());
    //increase the voltage for the next pulse(s)
    cVolt += step;
    }
```

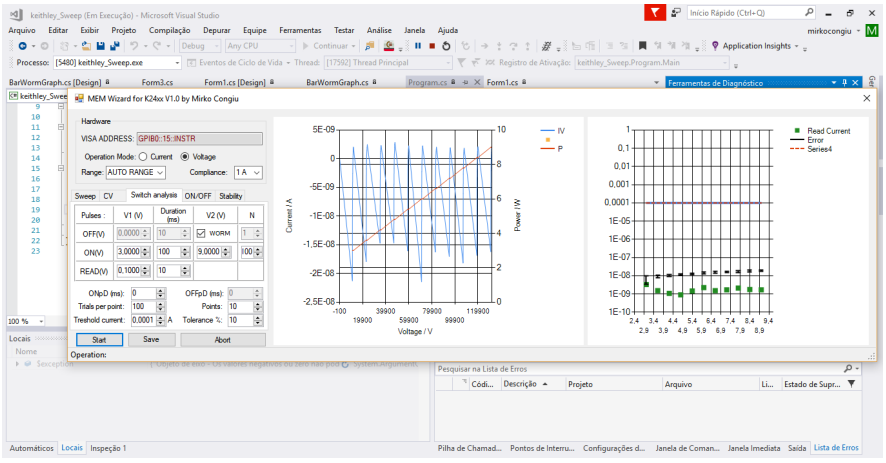


Figure A1. Screenshot of the software developed using the code functions presented in this work during a routine debugging operation in Visual Studio 2017. The graphical user interface (GUI) allows the user to input all of the necessary parameters for the experiment.