Programming PC-Based Multi-Function Oscilloscopes: A User-Friendly Approach to Rapid Prototyping of Spectral Analysis Code

FABRIZIO RUSSO, FRANCESCO TRAVAIN
Department of Engineering and Architecture
University of Trieste
Via A. Valerio 10, I-34127
ITALY
rusfab@univ.trieste.it

Abstract: Personal computer (PC)-based multi-function oscilloscopes are compact and low-cost programmable instruments that typically combine a multi-channel data acquisition unit, a signal generator and a USB interface to the PC. Although general purpose functions for spectral analysis based on the Fast Fourier Transform (FFT) are usually provided by the manufacturer in the accompanying software, the development of code for particular needs still remains a critical issue. This paper shows how new or more specific algorithms for frequency domain measurements can be easily implemented and tested for this class of instruments without spending time on a long and expensive training with dedicated development environments. The proposed approach is based on a recently introduced software shell, called DSPrototyper, where user interface and basic operations (for signal acquisition, processing and display) are already built-in and new measurement functions can be easily added by users having only a basic knowledge of programming languages. Some application examples are discussed in the paper focussing on the spectral analysis of a periodic signal and the implementation of algorithms that do not adopt the conventional time-domain windowing to reduce the spectral leakage.

Key-Words: - Measurement systems, virtual instruments, digital signal processing, spectral analysis, digital oscilloscopes.

1 Introduction

Personal computer (PC)-based instruments have rapidly become the most versatile architecture for implementing and testing digital signal processing (DSP) algorithms into a measurement system [1-13]. An interesting example of this class of instruments is constituted by multi-function USB PC oscilloscopes. In these devices the measurement hardware (connected to the PC by means of a USB interface) is composed of two main modules: an arbitrary waveform generator and a multi-channel data acquisition unit. Since USB PC oscilloscopes can generate and acquire test signals, they can potentially address a large variety of measurement tasks, depending on the availability of specific software. In this paper we shall focus on the development of code for spectral analysis based on the Fast Fourier Transform (FFT) [14-15]. Indeed, frequency domain measurements are very important in many research and application areas such as energy metering, medical instrumentation, telecommunications, consumer electronics, etc. A large body of scientific literature is available on this subject [16-20]. It is expected that users with expertise in programming languages can easily produce the appropriate code for spectral analysis by exploiting software libraries and/or development kits provided by manufacturers. On the contrary, the development of dedicated software often represents a very compelling issue for users having a limited knowledge of computer programming or that cannot spend time (and money) on a long training with dedicated development environments.

Fig.1 – Example of multi-function PC oscilloscope.

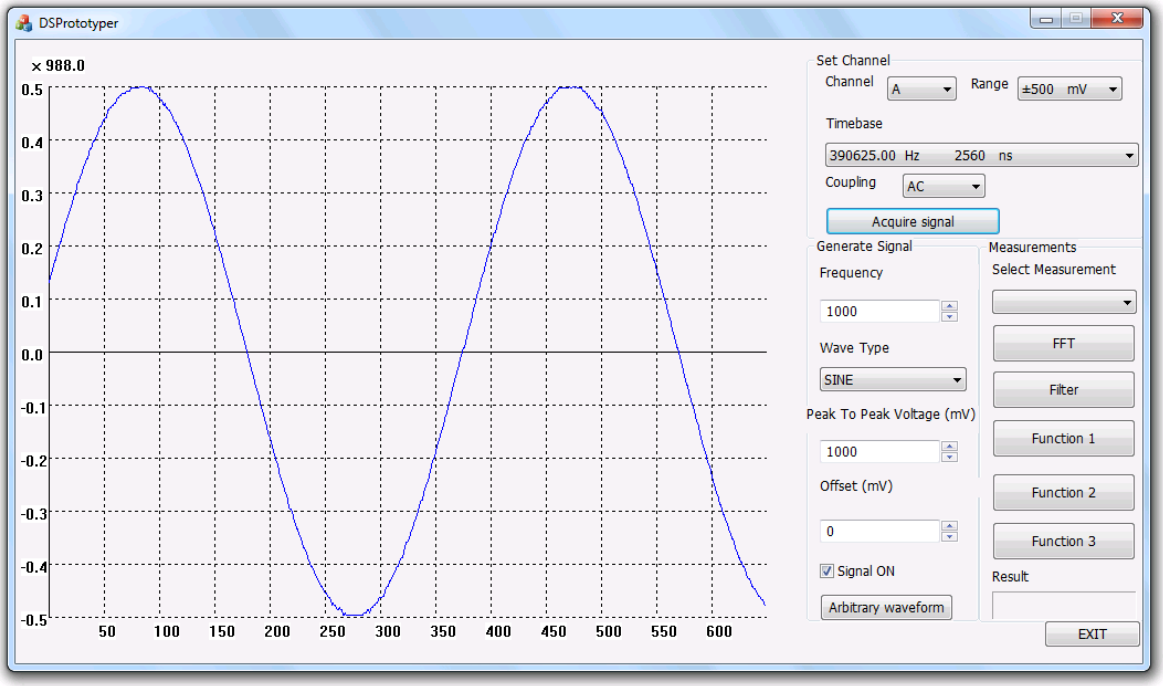

Fig.2 – Soft-panel of DSPrototyper.

In this paper we show how new or specific algorithms for frequency domain measurements can be easily experimented into a real environment by using a recently introduced software shell called *DSPrototyper* [1]. This software shell, written in C++ using Microsoft Visual Studio, provides a Windows-based user interface and a collection of basic operations for signal acquisition, processing and display. In this approach, the user can directly focus on new measurement functions and add them to the system. Only a minimal knowledge of C language is needed in order to write the source code of the desired algorithms in the empty bodies of some pre-defined C functions. As in our previous work, we chose the Picoscope 2204A (Fig.1), because it is a compact and low-cost programmable instrument and thus an excellent candidate for education purposes too [21-22]. The rest of the paper is organized as follows. Section 2 briefly reviews the built-in functions of DSPrototyper focusing on the generation of test waveforms, Section 3 describes the implementation of user defined algorithms for spectral analysis, and, finally, Section 4 reports the conclusions.

2 Main Operations of DSPrototyper

The soft-panel of *DSPrototyper*, with its Windows-based graphical user interface (GUI), is depicted in Fig.2. The main operations are summarized in Fig.3. A complete description of all the functions can be found in [1]. The operations that are specifically involved in the development and test of spectral analysis techniques are "arbitrary waveforms" and "user defined spectral analysis" (Fig.3).

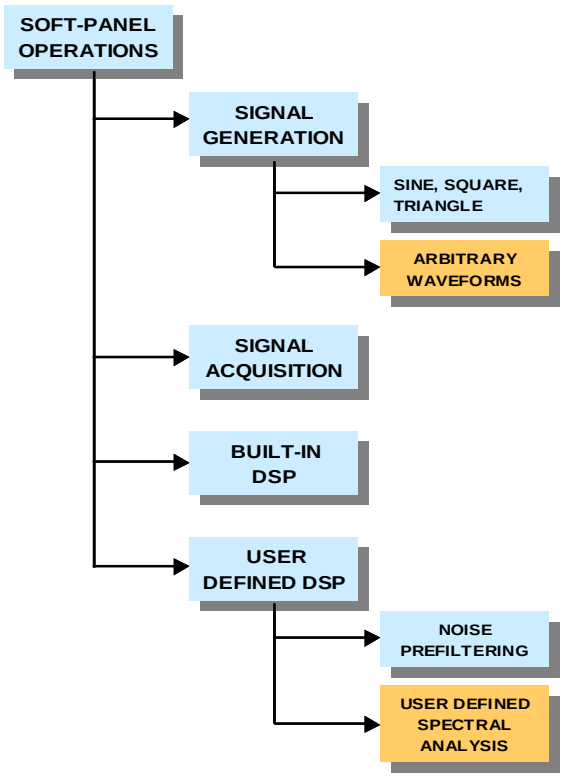

Fig.3 – Main operations of DSPrototyper.

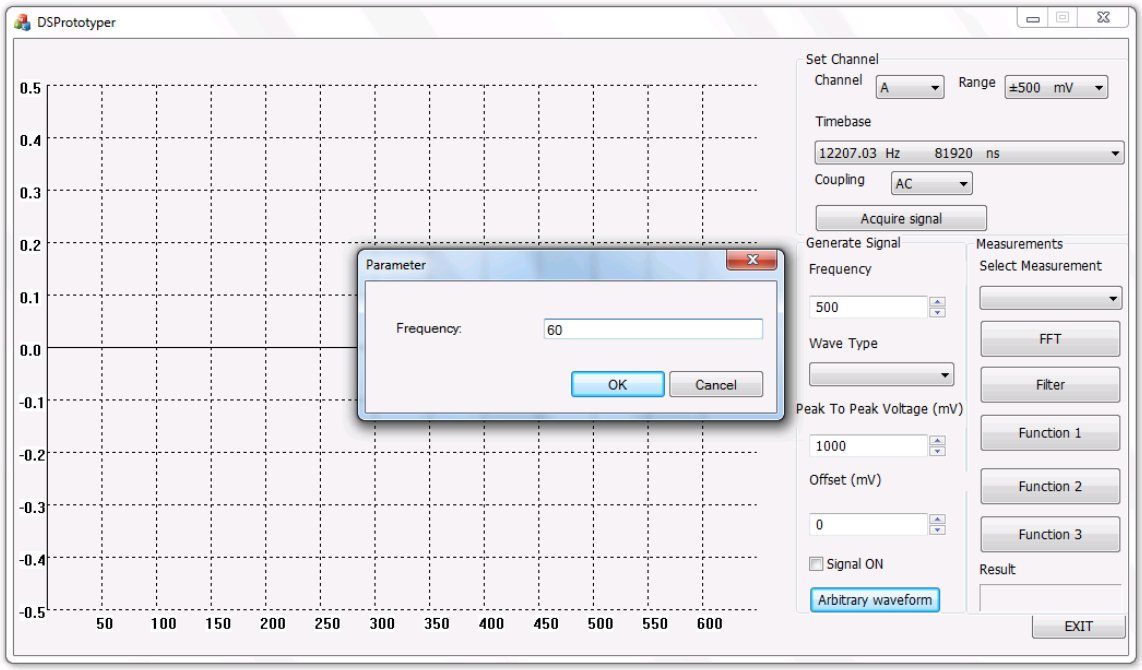

Fig.4 – Choice of fundamental frequency.

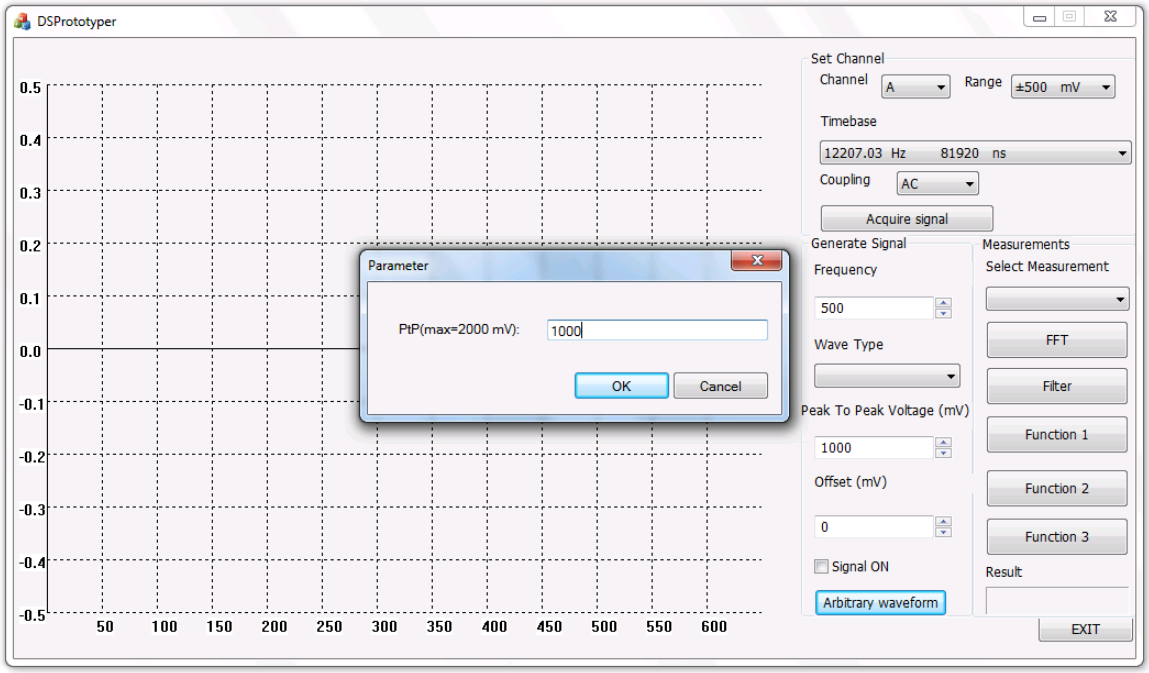

Fig.5 – Choice of peak-to-peak amplitude.

By clicking on the button “Arbitrary waveform”, the user can define a periodic signal. He/she can set the fundamental frequency, the actual amplitude, order and relative amplitude of additional harmonics. The generation of electrical signals whose harmonic content is known, is very useful to evaluate the accuracy of spectral analysis algorithms. As an example, let us suppose we want to generate a periodic waveform having fundamental frequency 60 Hz, total amplitude 1000 mV (peak-to-peak), a 5^{th} harmonic (with relative amplitude 80%), a 14^{th} harmonic (with relative amplitude 60%), and finally a 15^{th} harmonic (with relative amplitude 60%). The simple procedure is shown by Figs.4-8. Once the waveform data have been computed, they are passed to the digital-to-analog module of the instrument in

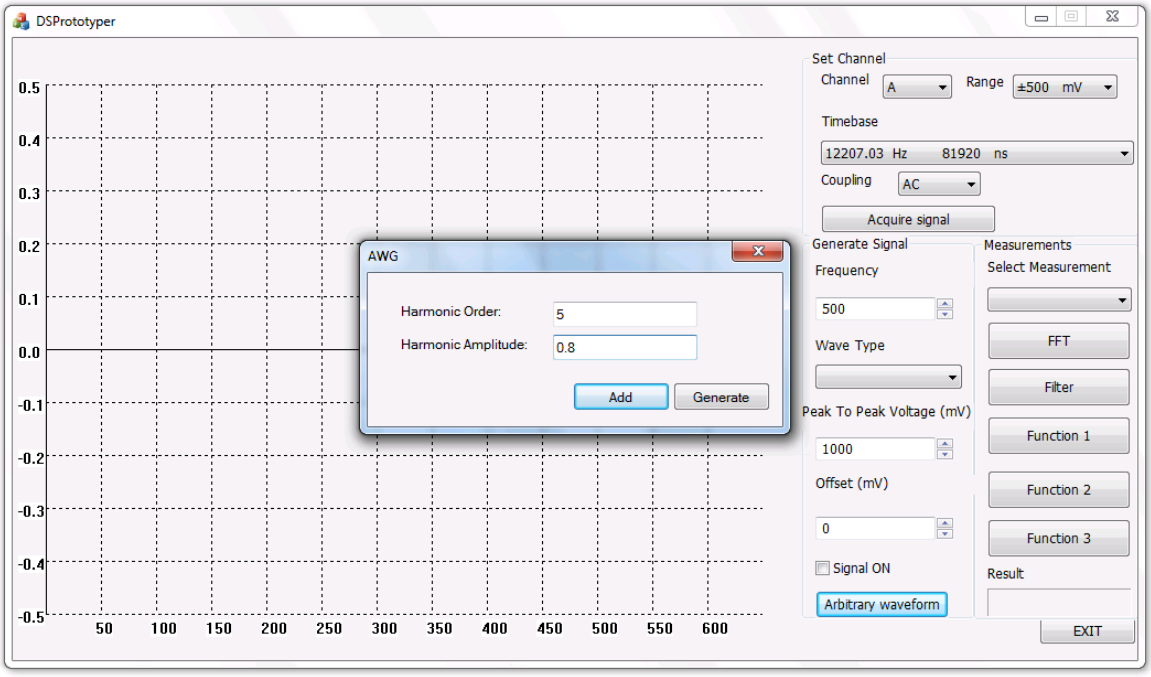

Fig.6 – Choice of harmonic order and relative amplitude (harmonic component #1).

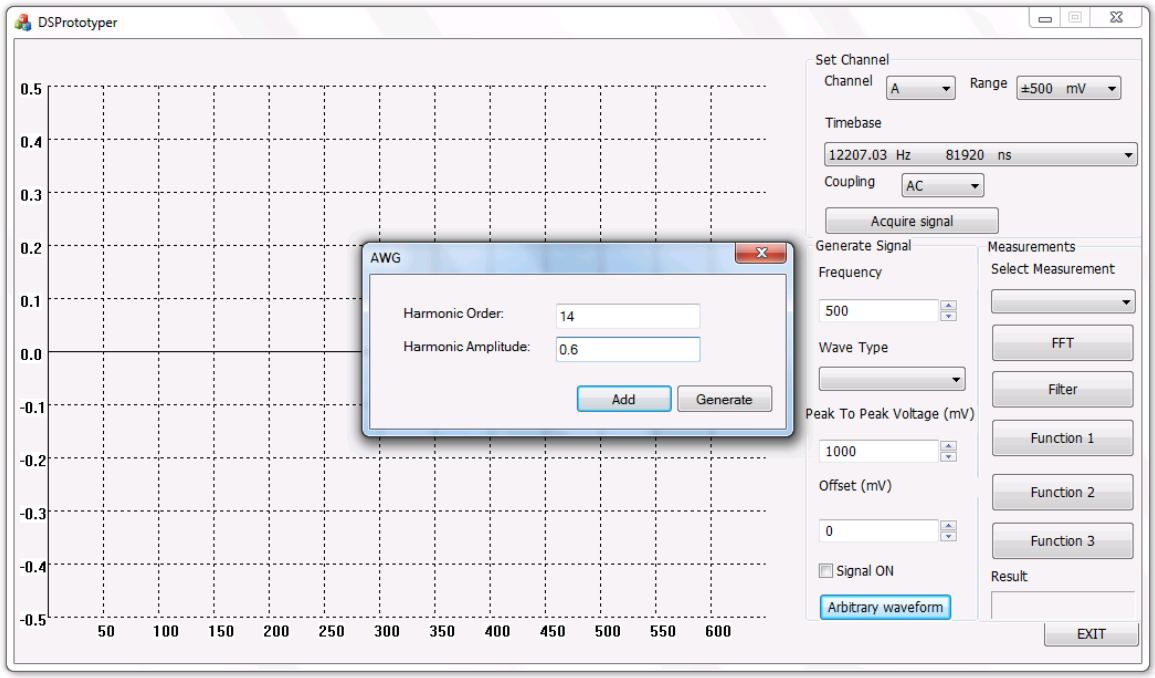

Fig.7 – Choice of harmonic order and relative amplitude (harmonic component #2).

order to produce the corresponding electrical signal. If we connect the generator output to the input channel A of the instrument, we can acquire the electrical signal and show it on the screen (Fig.9). Signals with adjacent harmonic components are a challenging issue for many spectral analysis algorithms. Many other examples of test signals can be easily assembled. DSPrototyper offers a group of built-in DSP functions devoted to FFT-based spectral analysis. In addition to the rectangular window, other choices are available for reducing the spectral leakage by resorting to time domain windowing. We shall describe in the next section how new algorithms can be implemented.

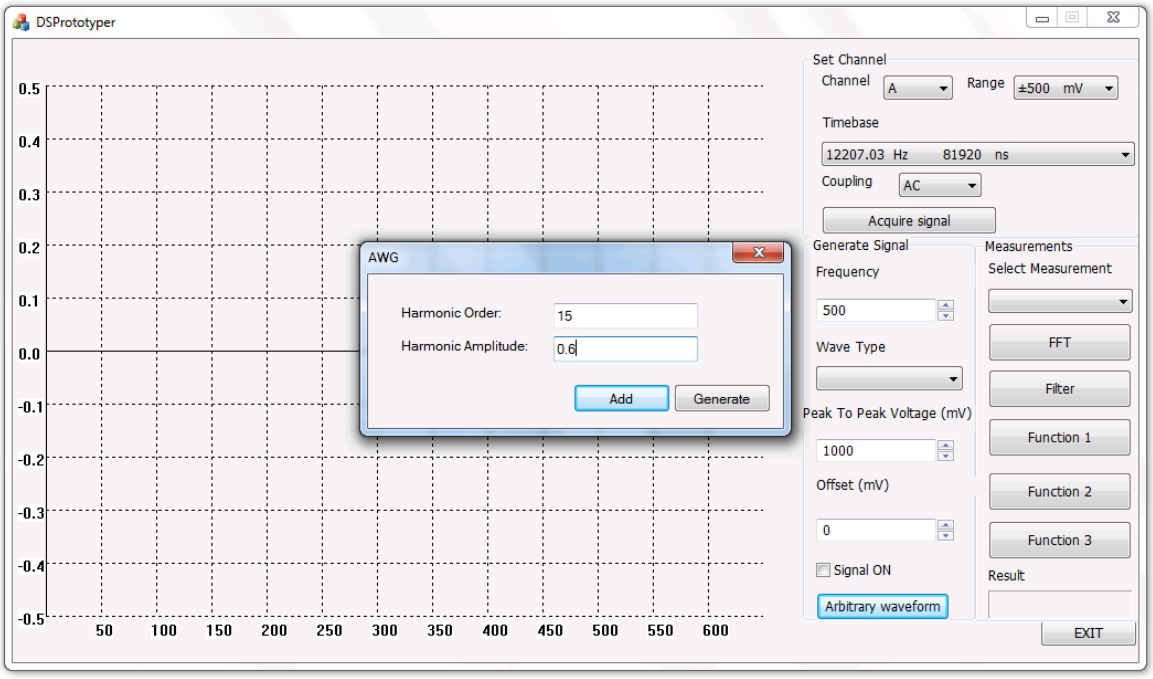

Fig.8 – Choice of harmonic order and relative amplitude (harmonic component #3).

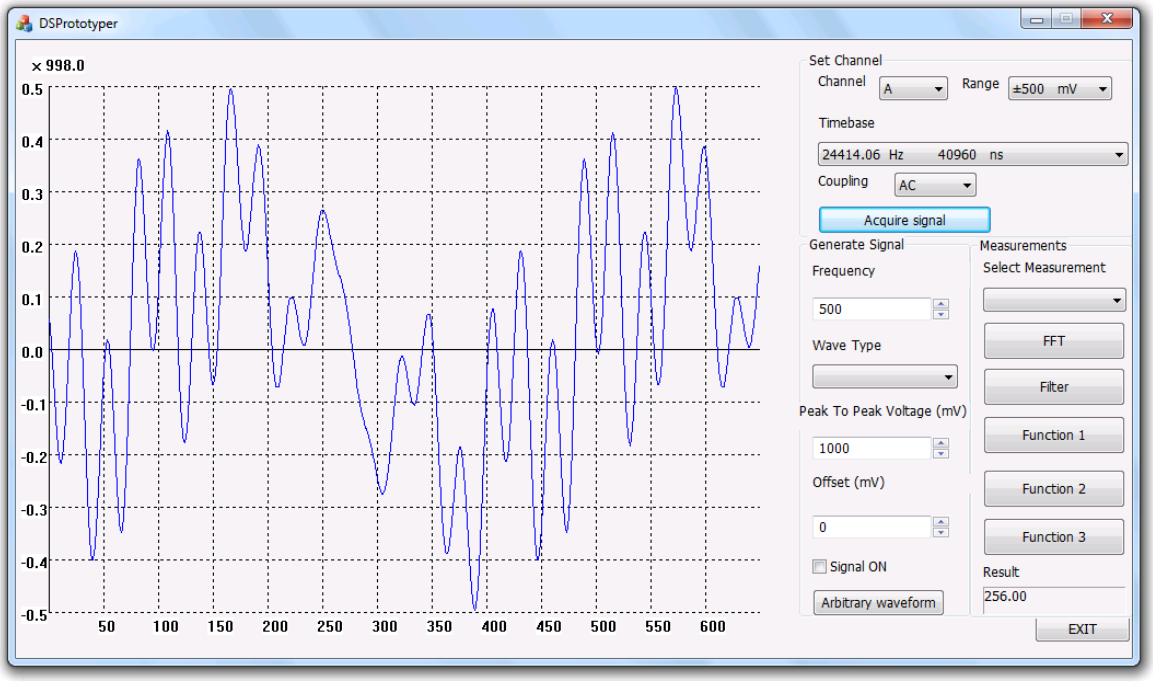

Fig.9 – Acquisition of the generated electrical signal.

3 User defined Functions for Spectral Analysis

The user-friendly environment allows the developer to test new algorithms for spectral analysis without wasting time with user-interface and hardware control problems. On the other hand, the already implemented FFT-based functions offer a comparison tool for evaluating the advantages of user defined techniques. By pressing the appropriate software buttons (Fig.10) three user defined functions (namely Function 1, Function 2 and Function 3) can be executed. More functions will be available in a future release of the software shell.

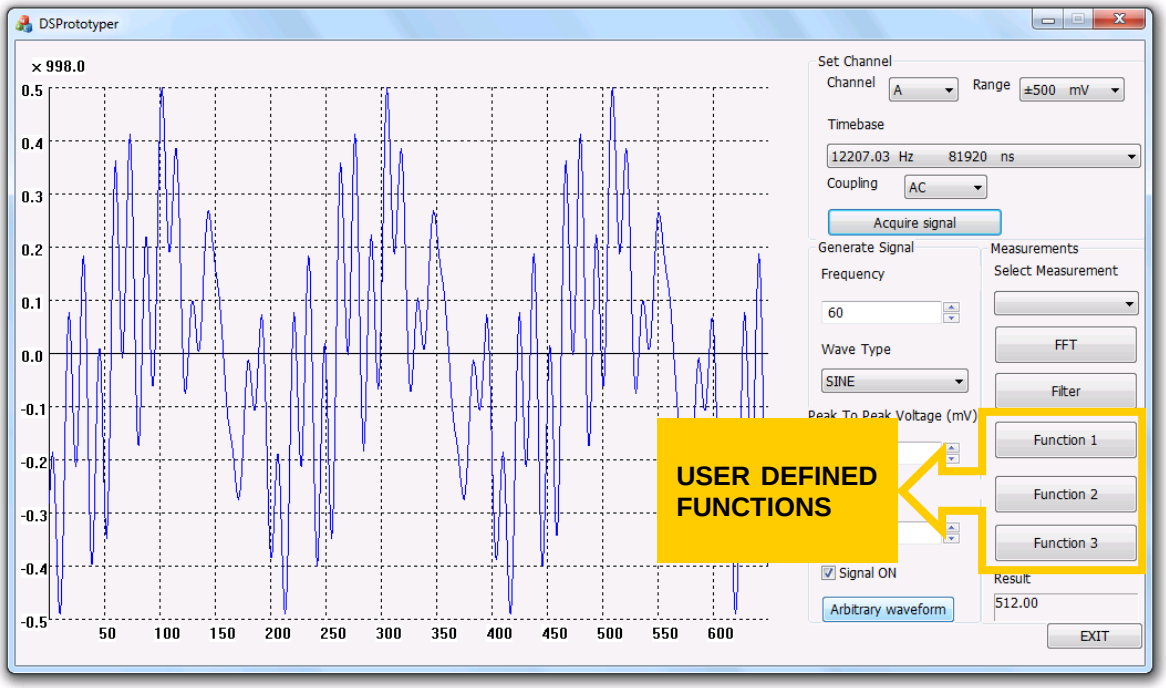

Fig.10 – Software buttons for calling the user defined functions.

DSPrototyper is a Visual Studio project written in C++: it includes a collection of source files as shown in Fig.11. It is worth pointing out that only a file, namely MOD_DSP.cpp, is devoted to the implementation of digital signal processing operations such as the user defined functions. All the remaining files are dedicated to the GUI and can be ignored by the user. The implementation of user defined algorithms is very simple and needs only a basic knowledge of C programming language. Indeed, even if MOD_DSP.cpp is formally a C++ file, it can include ANSI C source code too. The procedure is very simple and involves the following steps:

1) write the plain C source code of the desired algorithm into the empty body of one of the predefined functions in MOD_DSP.cpp;
2) rebuild the overall project, without interacting with any other file.

The input of parameter values is also very simple: it is managed by a predefined function, namely "InputParameterWithName", that opens a dialog window and asks the user about the necessary information. We shall focus here on FFT-based spectral analysis of a periodic signal and, in particular, on the implementation of algorithms that do not adopt the usual time-domain windowing in order to reduce the spectral leakage [23-24]. A possible strategy could resort to a preliminary measure of the period and then to a synchronization of the sampling frequency in order to achieve an integer number of data points per period (possibly equal to a power of two). If the sampling frequency cannot be finely adjusted, a slightly different approach should be adopted as described by the following step-by-step procedure:

1) obtain an estimate of the period length;
2) interpolate the samples with an equidistant grid with N samples per period, such that $N=2^n$;
3) compute the spectrum using an N-points FFT.

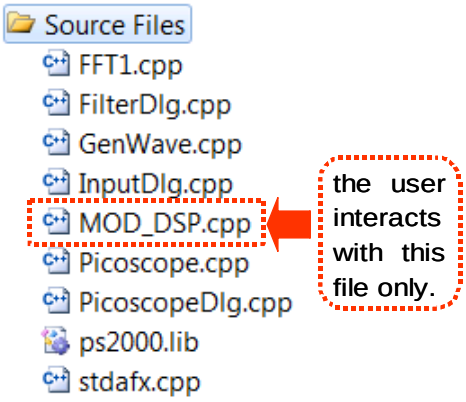

Fig.11 – Source file structure of DSPrototyper: MOD_DSP is the file that includes all the DSP code, including the user defined functions.

```
////////////////////////////// SPECIAL FUNCTIONS ////////////////////////////////
double Function1(){
    double funpar1=InputParameterWithName("BILATERAL - choose N ");
    double funpar2=InputParameterWithName("BILATERAL - choose Sr");
    double funpar3=InputParameterWithName("BILATERAL - choose Sd");
    Bilateral(funpar1,funpar2,funpar3,sigsize,isig,osig);
    return 1;}

double Function2(){
    return Measure_RMSvalue(isig);
}

double Function3(){
    double n,m;
    double funpar1=InputParameterWithName("Number of points ");
    n=funpar1;
    m=Measure_Period(isig);
    Reconstruct(isig,isigFFT,m,n);
    Execute_FFT(isigFFT,n);
    return n;
}
```

Fig.12 – User defined function 3 for spectral analysis.

The source code of the corresponding C functions is written in the MOD_DSP.cpp. If we want to call these functions, we only need to insert a few statements in the (originally empty) body of a specialized C function. For example, let us suppose that "Function3" has not been used yet. The first operation (Fig.12) asks the user for the number of FFT points and assigns this value to the variable n. The second operation estimates the number of samples m that corresponds to a period of the input signal (stored into the "isig[]" array). A double threshold technique is used for this purpose. Then, the samples are interpolated and the result is stored in the "isigFFT[]" array. Finally, these data are passed to the FFT algorithm for the computation of the spectrum. The waveform described in the previous Section is used as a test signal. By pressing the button "Function 3", the overall procedure for spectral estimation is executed (Figs.13-14). We can see that the adopted procedure yields satisfactory results. The measured fundamental frequency is f_1=59.8 Hz yielding a relative error lower than 0.4%. (More accurate results could be obtained by increasing the sampling frequency and then the number of samples per period). The spectral leakage is almost negligible and the adjacent spectral lines are well separated. For a comparison, the results given by classical rectangular and Hamming windowing techniques are reported in Figs.15-16. As expected, the spectral leakage is very annoying especially for the rectangular window. In all cases, the frequency resolution does not suffice (f_1=47.7 Hz) and the spectral lines cannot be distinguished. Clearly, many other signals and strategies for spectral analysis could be taken into account and tested using the proposed development environment.

4 Conclusions

DSPrototyper is a recently introduced software shell for rapid prototyping of DSP techniques in multi-function PC oscilloscopes. The target is the user who wants to focus on a measurement algorithm without wasting time and money with a long training with dedicated development environments. We have shown how user defined algorithms for spectrum estimation can be quickly implemented and tested. Only a minimal knowledge of C language is necessary in order to write the source code of the spectral analysis algorithms, whereas all other operations devoted to data acquisition and display are already built-in. On the other hand, real electrical signal (whose harmonic content is known) can be easily generated to test and possibly improve the spectral analysis strategy. It is expected that all the mentioned features make the software shell very useful for experimenting classical and novel spectral analysis techniques in education applications that are an important target of our work.

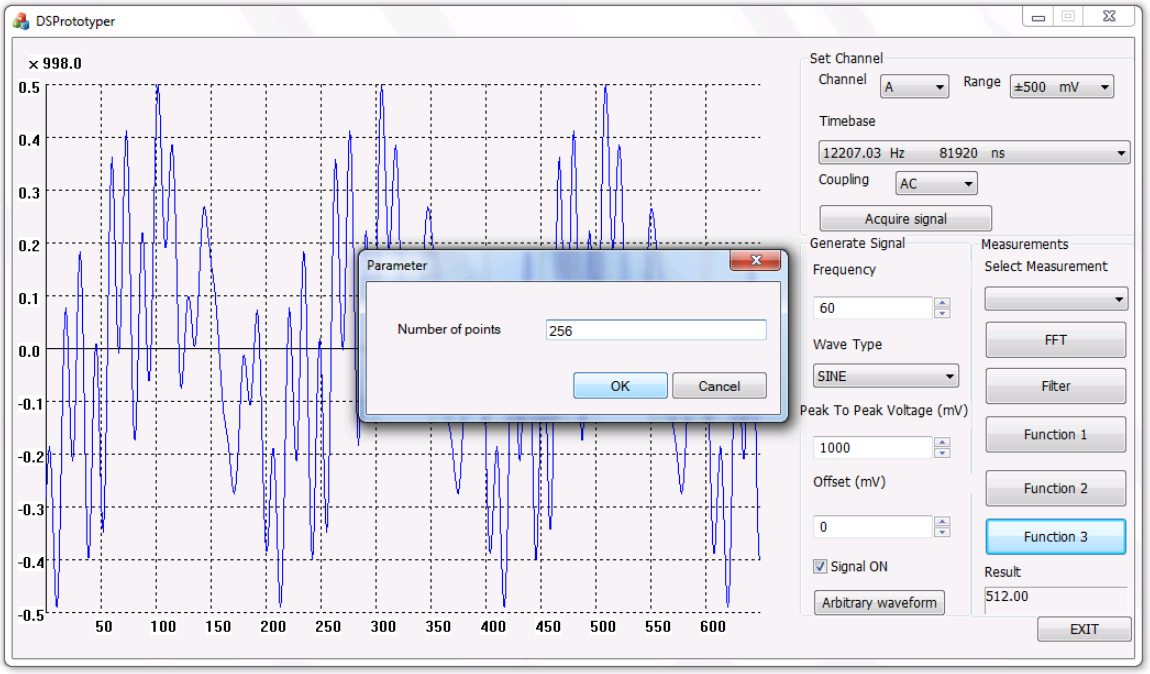

Fig.13 – Choice of the number of FFT points.

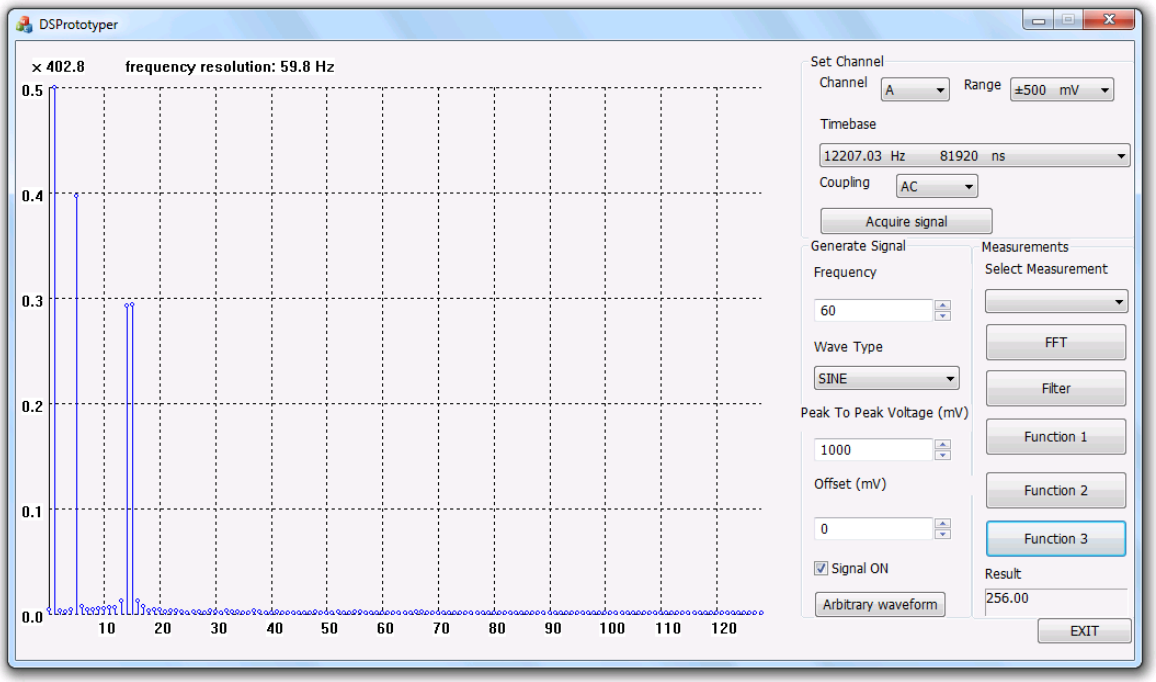

Fig.14 – Spectral lines yielded by the user defined algorithm.

References:

[1] F. Russo and F. Travain, "Rapid Prototyping of DSP Code for PC-Based Multi-Function Oscilloscopes", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp.462-470, ISSN 1998-4464.

[2] D. N. Vizireanu, "Quantised Sine Signals Estimation Algorithm for Portable Digital Signal Processor Based Instrumentation", *International Journal of Electronics*, vol.96, n.11, pp. 1175-1181, 2009.

[3] A. Lakshmikanth and M. M. Morcos, "A Power Quality Monitoring System: a Case Study in DSP-based Solutions for Power

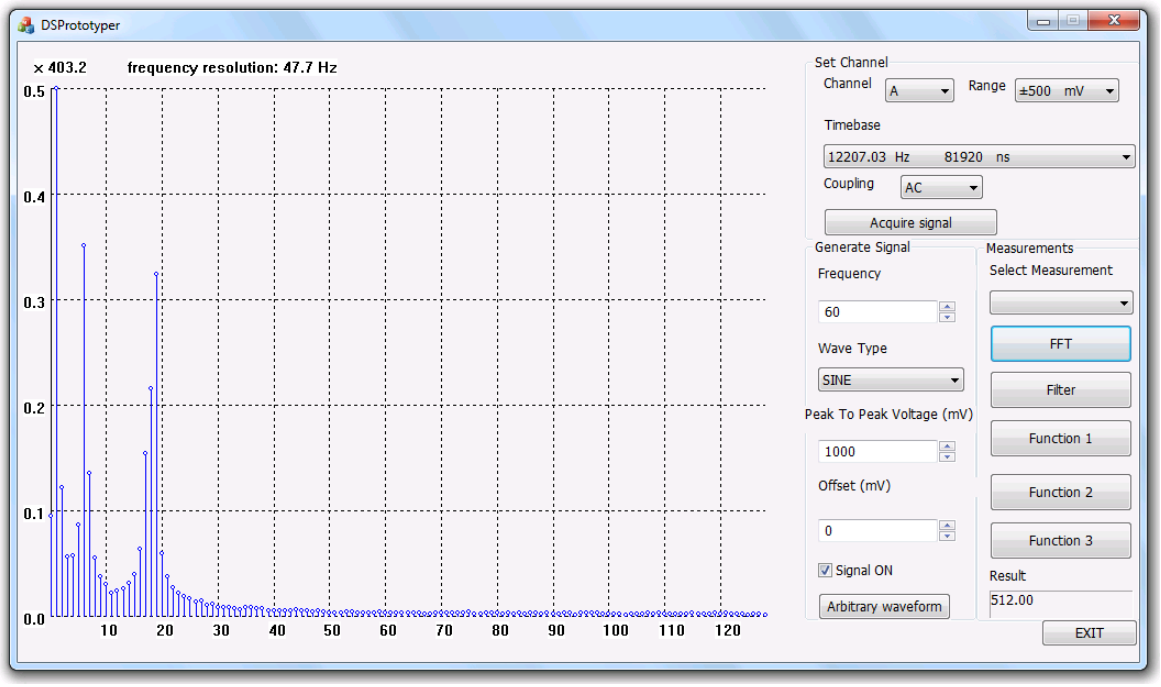

Fig.15 – Spectral lines yielded by the built-in function for spectral analysis (rectangular window).

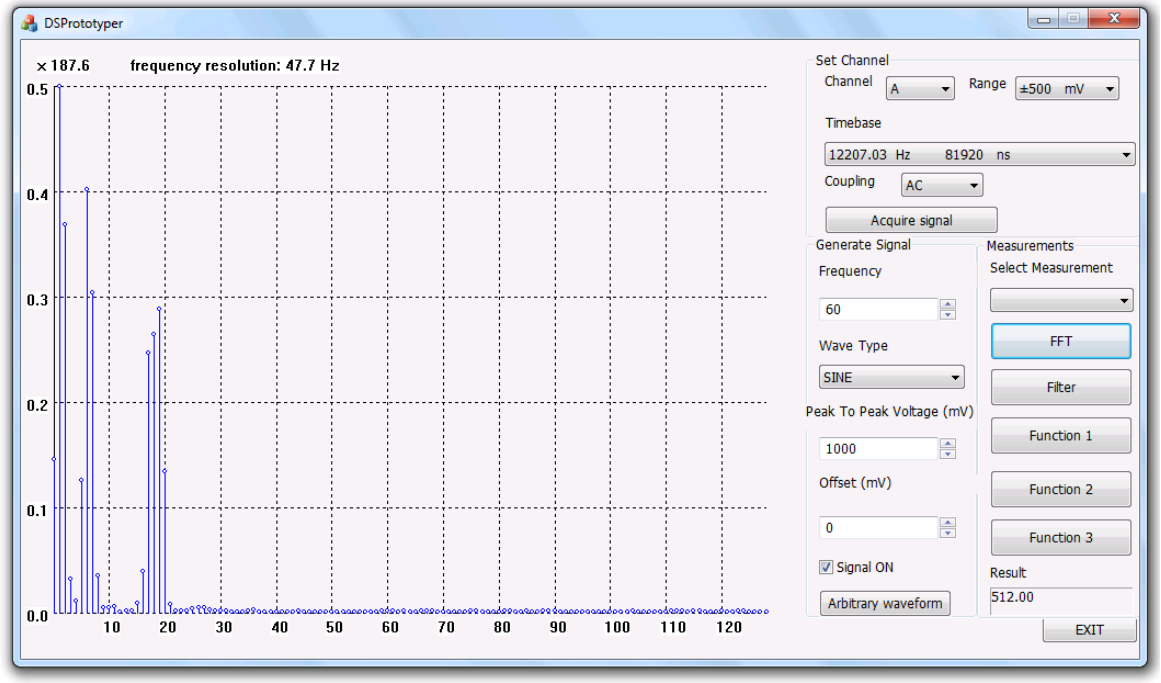

Fig.16 – Spectral lines yielded by the built-in function for spectral analysis (Hamming window).

Electronics", *IEEE Trans. on Instrumentation and Measurement*, Vol.50, n.3, pp.724-731, 2001.

[4] A. Lakshmikanth and M. M. Morcos, "A Power Quality Monitoring System: a Case Study in DSP-based Solutions for Power Electronics", *IEEE Trans. on Instrumentation and Measurement*, Vol.50, n.3, pp.724-731, 2001.

[5] M. B. Yeary, R. J. Fink, D. Beck, D. W. Guidry, M. Burns, "A DSP-based mixed-signal waveform generator", *IEEE Trans. on Instrumentation and Measurement*, Vol.53, n.3, pp.665-671, 2004.

[6] J. Mindykowski and T. Tarasiuk, "Development of DSP-based Instrumentation for Power Quality Monitoring on Ships", *Measurement* (Journal of the International Measurement Confederation), Elsevier, Vol.43, n.8, pp.1012-1020, 2010.

[7] O. Märtens et al., "DSP-Based Power-Quality Monitoring Device", Proc. 2007 IEEE International Symposium on Intelligent Signal Processing, WISP, 2007.

[8] M. Artioli, G. Pasini, L. Peretto, R. Sasdelli, F. Filippetti, "Low-cost DSP-based Equipment for the Real-time Detection of Transients in Power Systems, *IEEE Trans. on Instrumentation and Measurement*, Vol.53, n.4, 2004, pp.933-939.

[9] M. Brando et al. "A PC-Based Instrument for Automatic Monitoring and Control of a CPVT Power Plant", Proc. 20th IMEKO TC4 International Symposium, Benevento, Italy, September 15-17, 2014, pp. 40-44.

[10] F. Corrêa Alegria, E. Molino-Minero-Re, S. Shariat-Panahi, "Evaluation of a Four-point Sine Wave Frequency Estimator for Portable DSP based Instrumentation", *Measurement* (Journal of the International Measurement Confederation). Elsevier, Vol.45, n.7, 2012, pp.1866-1871.

[11] F. Adamo, G. Cavone, A. Di Nisio, A.M.L. Lanzolla, M. Spadavecchia, "A Proposal for an Open Source Energy Meter", Proc. IEEE Instrumentation and Measurement Technology Conference, I2MTC, Minneapolis, MN, USA, May 6-9, 2013, pp. 488-492.

[12] D. Bellan, "A Discrete Model of Jitter for Coarsely Quantized Waveforms", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp. 269-274, ISSN 1998-4464.

[13] F. Adamo, F. Attivissimo, G. Cavone, C. Guarnieri Calò Carducci, A. M. L. Lanzolla, "New Technologies and Perspectives for Laboratory Practices in Measurement Science", Proc. 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), May 11-14, 2015, Pisa, Italy, 2015.

[14] S. Rapuano and F. J. Harris, "An Introduction to FFT and Time Domain Windows", *IEEE Instrumentation & Measurement Magazine*, December 2007, pp. 32-44.

[15] E. Brigham, Fast Fourier Transform and Its Applications, Englewood Cliffs, NJ: Prentice Hall, 1988.

[16] F.J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform", *Proceedings of the IEEE*, Vol.66, N.1, 1978, pp.51-83.

[17] F.J. Harris and W. Lowdermilk, "Cost Effective, Versatile, High Performance Spectral Analysis in a Synthetic Instrument", IEEE Instrumentation & Measurement Magazine, August 2009, pp.8-13.

[18] C. Liguori, V. Paciello, A. Paolillo, and A. Pietrosanto, "Improving and Qualifying Spectrum Analysis Made by Digital Scopes", *IEEE Trans. on Instrumentation and Measurement*, Vol.56, N.6, 2007,pp.2411-2419.

[19] D. Bellan, "Estimation of Harmonic Power Components Affected by Frequency Instability", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp. 275-280, ISSN 1998-4464.

[20] F. Adamo, F. Attivissimo, A. Di Nisio, M. Savino, M. Spadavecchia, "A Spectral Estimation Method for Nonstationary Signals Analysis with Application to Power Systems", *Measurement* (Journal of the International Measurement Confederation) Elsevier, Vol.73, n.11, 2015, pp.247-261.

[21] https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview

[22] PicoScope 2000 Series Programmer's Guide: https://www.picotech.com/oscilloscope/2000/picoscope-2000-manuals

[23] J. Schoukens, Y. Rolain, G. Simon, R. Pintelon, "Fully Automated Spectral Analysis of Periodic Signals", IEEE Instrumentation and Measurement Technology Conference Anchorage, AK, USA, 21-23 May, 2002, pp.299-302.

[24] R. M. Hidalgo, J. G. Fernandez, R. R. Rivera, and H. A. Larrondo, "A Simple Adjustable Window Algorithm to Improve FFT Measurements" *IEEE Transactions on Instrumentation and Measurement*, Vol.51, N.1, 2002, pp.31-36.