Visual C++ Complex Mathematical Signal Generator

C. Dughir[1]

[1] *"Politehnica" University of Timisoara, Electronics and Telecomunications, V. Pârvan 2A, 30022, Timisoara, Romania, +40256403365, ciprian.dughir@etc.upt.ro*

***Abstract** –* This paper present a complex mathematical signals generator that can generate a combination of mathematical signals at the output of a PC sound card. The user can describe complex mathematical functions to be generated. The signal generator proposed is very simple to use and don't require complex and expensive hardware to run.

I. Introduction

In the testing process of different parts of an analysis system, very complex input signals are needed. In this case, a simple function generator is useless. A complex mathematical signal generator is very expensive, and in most cases the signals they provide is not 100% in concordance with the signal needed by specific applications. In this paper is presented a signal generator who can generate an indefinite number of combinations of basic mathematical functions (see Table 1) and send these signals to the output of a sound card. In many applications, the 5 kHz maximum signal frequency is far enough (e.g.: electrocardiography). The goal is to build a simple and cheap signal generator useful for many users and at an affordable price even for small experiments.

II. The solution proposed

The solution proposed in this paper use a PC sound card to generate the complex shape signals. These signals are mathematically computed by a special software component. The user can specify the mathematical function combinations to be generated by the application (see figure 1).
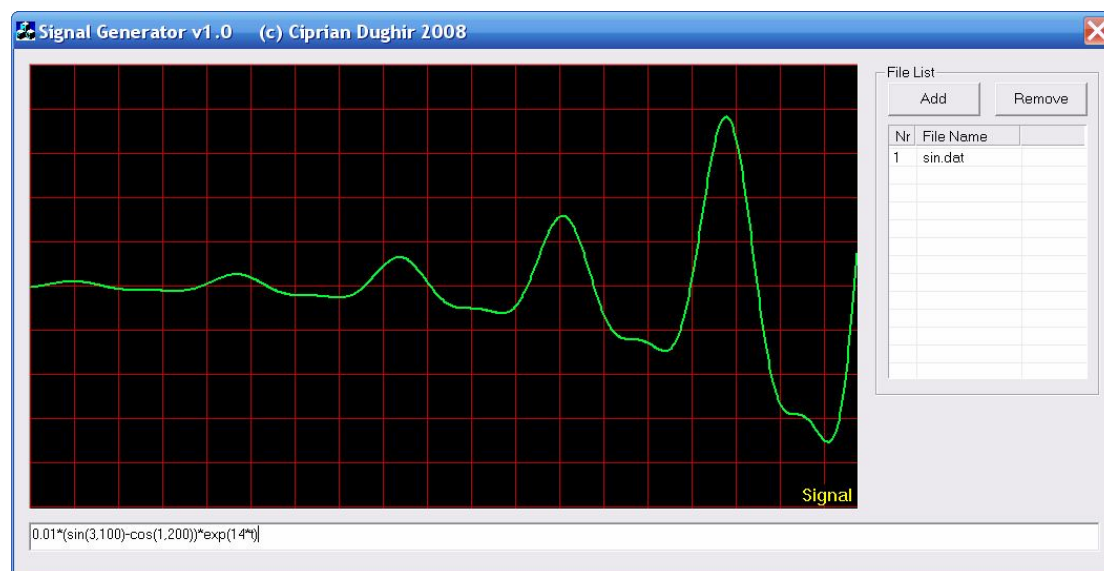

Fig. 1. Application window of the signal generator

The signal generator application window contains a graph displaying the generated signal, an edit box to input the mathematical function describing the desired output signal, and a list box of files with pre-recorded signals to be generated. Once introduced in the edit box, the mathematical formula is immediately interpreted and the corresponding output signal is generated. If an error is encountered or the user inputs an invalid formula, a message box with the error is displayed.

If the user needs to generate signals pre-recorded in a file (generated with other software or

captured from real medium), with specific disturbances or with specific shapes, the signals can be generated by the application without problem. The user can add up to 1000 different input signals from files (limited by the list box component capability), the generator output engine mixing them by applying the desired operand between the signals. In order to obtain the correct shape of the output signal the user must know the sampling frequency of each input signal.
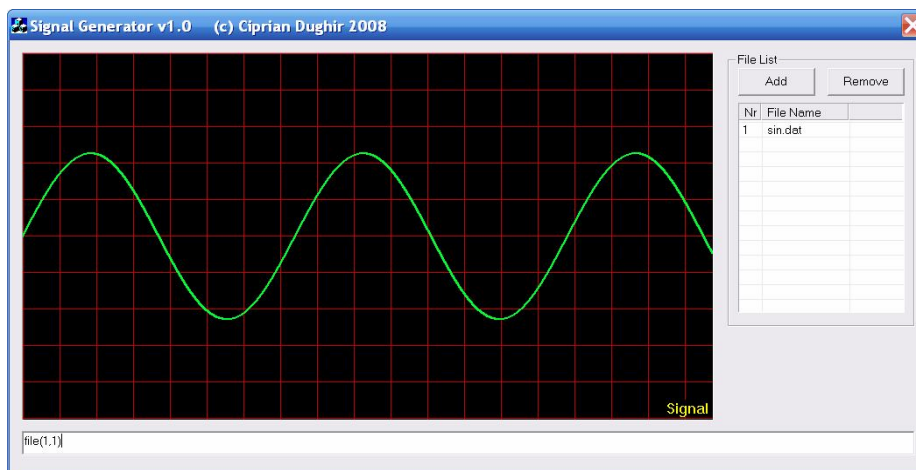

Fig. 2. Generation of a sine wave read from a file

In figure 2 is presented the generation of a sinus waveform read from a file generated using Matlab 7 software.

In the evaluation process of the global function specified by the user, each mathematic functions and operators are separated from the global function, and then are interpreted. The function interpreter extracts each basic mathematic function and computes the value of the function at the current moment of time. The operators are applied to the results. If any of the parameters of the function is missing, the engine presumes that parameters are equals to zero. The functions interpreter has a set of few basic instructions and operators. These are simple mathematic functions and operators. All known mathematic operators are available to the user. The basic functions which the engine can interpret are presented in the following table:

Table 1. Correspondence between the mathematic functions and function interpreter syntax

Mathematic function	Interpreter Syntax
sin(x)	sin(amplitude, frequency, offset)
cos(x)	cos(amplitude, frequency, offset)
tan(x)	tan(amplitude, frequency, offset)
ctn(x)	ctn(amplitude, frequency, offset)
sqrt(x)	sqrt(amplitude, x, offset)
pow(x,y)	pow(x, y, amplitude, offset)
rand(x)	rand(amplitude, offset)
sinh(x)	sh(amplitude, frequency, offset)
cosh(x)	ch(amplitude, frequency, offset)
tanh(x)	th(amplitude, frequency, offset)
ctanh(x)	cth(amplitude, frequency, offset)
asin(x)	asin(amplitude, frequency, offset)
acos(x)	acos(amplitude, frequency, offset)
atan(x)	atan(amplitude, frequency, offset)
actn(x)	actn(amplitude, frequency, offset)
sign(x)	sign(x)
abs(x)	abs(x)
log(x)	log(x)
ln(x)	ln(x)
exp(x)	exp(x)
t (time)	t

The function interpreter can process user defined functions. The user defined functions are mathematical formulas build using the basic mathematic functions, recognized by the function interpreter, grouped together in a file. This is like an auxiliary library for the application. The file must have the extension fnc. If any other function than presented in Table 1 is found in the global function specified by the user in the main window of the application, the interpreter search in the current directory of the application a file named like the unknown function. If such file is found, and if the syntax is correct, the function will be internally replaced with the formulas contained in the file.

The signal generator can generate only periodical signals, the output signals being sampled at 44.1kHz (16bit).
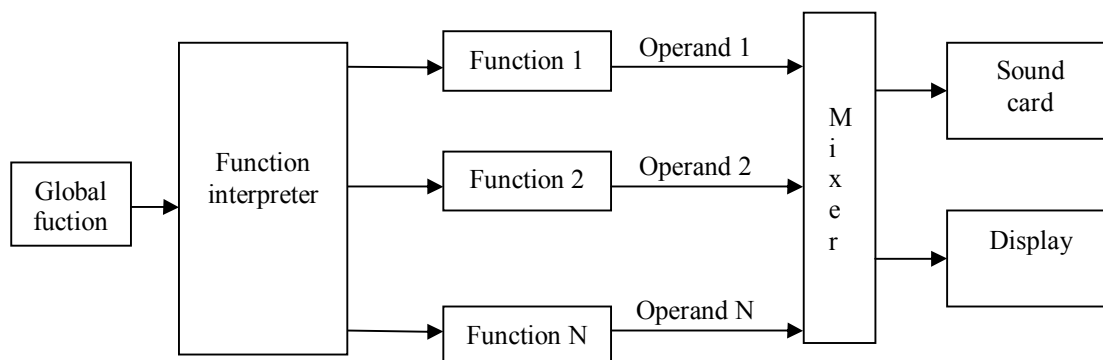

Fig. 3. Signal generator application diagram

The application was written in Microsoft Visual C++ 6, using MFC libraries [1][2]. The sound generation engine is modular and if, in the future, a National Instruments signal generation card will be added, only the sound generator class must be replaced in order the application to function properly. A simple wrapper class must be inserted to communicate properly with the National Instruments NiDaqMX library.

III. Experimental results

The generated signals at the output of the sound card was visualised with a National Instruments NI PCI 6254M acquisition board. In Figure 4 is presented a 50Hz sine wave with a short bi-exponential signal over imposed. The generated signals correspond with the signals specified by the user in the application window. The test was made on 8 different sound cards (Audigy 1, Audigy 2, SB Live 5.1, Sigmatel, ESS 688, ESS 1688, MAX and Turtle Beach). Due to some particularity of different types of sound cards existing on the market (the presence or the absence of the output capacitor), the minimum guaranteed frequency is 20Hz, and for some particular cases (Turtle Beach), the minimum frequency is 1Hz. In conclusion, the signal generator proposed can be used to generate signals with a 20Hz - 5kHz amplitude and user defined signal shapes.

IV. Conclusions

The signal generator can be used to generate signals at the output of the sound card, or in demonstrative purposes to simply visualize on the screen the desired function or combination of functions. In our days, almost any personal computer has a sound card, the solution proposed here being very cheap and easy to use for everyone. The user has no need to buy expensive signal generators or expensive signal generation systems in order to generate the desired waveforms. The signals provided by any sound card are between 0 and $\pm0.707V_{ef}$, being sufficient in most of the cases. A simple operational amplifier can be used in order to amplify the output signal [3][4].

The author use this signal generator to test his power network disturbance detector presented in [5]. Another utility of this signal generator is in the electro-medical applications class for presenting to the student different EKG waveforms, and other biomedical signals pre-recorded in files.

The advantage of the signal generator presented here over the Matlab based signal generators is the possibility to run this application on any computer better than 80486DX, with a minimum

amount of 8MB of RAM memory. The signal generators build on Matlab environment need a very huge amount of memory and better CPU's. The executable code of the applications was build with speed optimizations in order to reduce the CPU load. Being written in Visual C++ using static libraries option, the application doesn't need any additional files or libraries. With little code modifications, the application can be ported to Microsoft Windows Mobile operating system, running on portable devices.
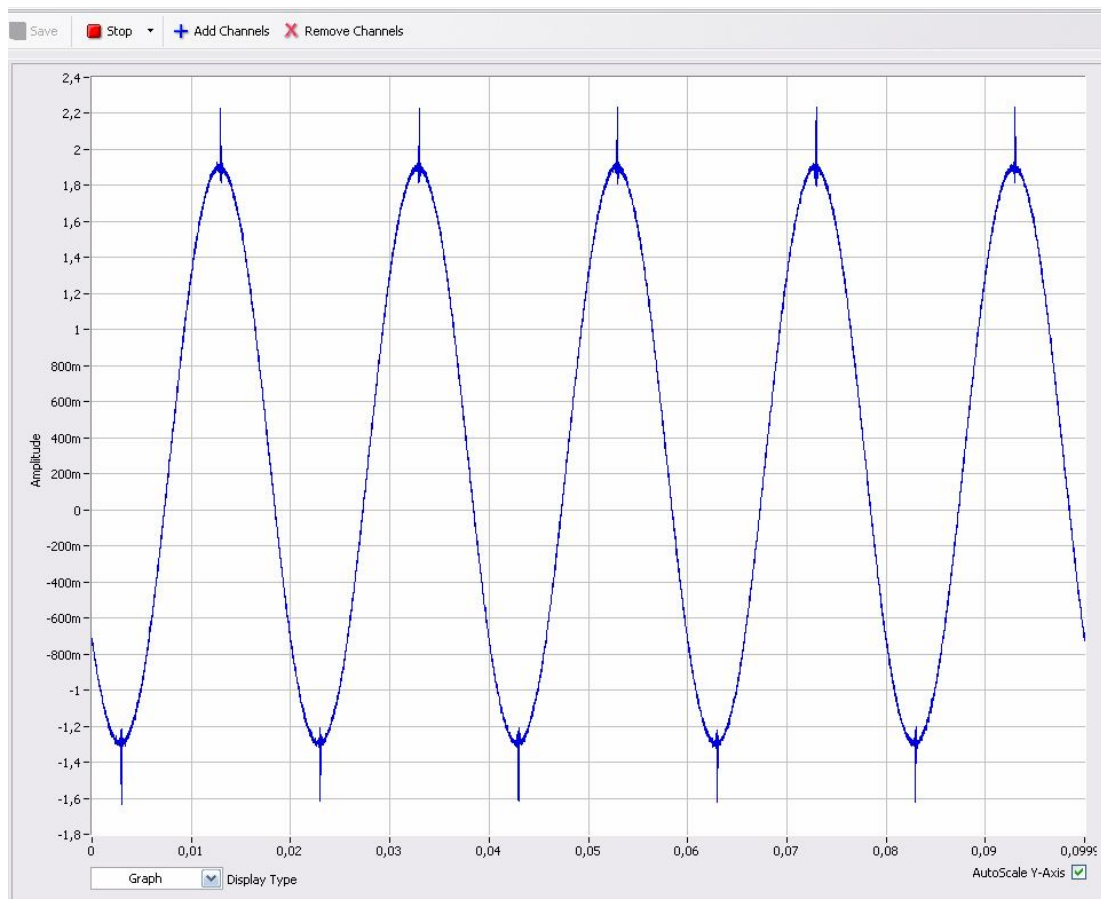

Fig. 4. Signal generator output signal captured with National Instruments Measurements and Automation Studio

References

[1] Microsoft, *Microsoft Developer Network*, Documentation DVD, October 2001
[2] C. Petzold, *Programming Windows*, 5th edition, Microsoft Press, 1998
[3] M. Ciugudean, *Circuite integrate liniare-Aplicaţii*, Editura Facla, Timişoara, 1986.
[4] G. Asch et collaborateurs, *Les capteurs en instrumentation industrielle*, Imprimeie Gauthier-Villards, France, septembre 1991.
[5] C. Dughir, "Detecting symmetrical disturbances in electrical power systems", pag. 27-31, *Doctor Etc 2005*, Timişoara, 22 September 2005.