MATLAB Toolboxes for A/D Converters Characterization

F. Adamo, F. Attivissimo and N. Giaquinto

Laboratory for Electric and Electronic Measurements
Department of Electrics and Electronics (DEE) – Polytechnic of Bari
Via E. Orabona 4, 70125 Bari - Italy
Phone: +39 080 5963318 Fax +39 080 5963410
e-mail: [adamo, attivissimo, giaquinto]@deemis06.poliba.it

I INTRODUCTION AND MOTIVATIONS

The considerable progress achieved both in the fields of the electronic technologies and digital signal processing methods is modifying the idea of measuring and measurement devices and the concepts of auto-diagnosis, self-testing and auto-calibration, bringing to a new concept of measuring instrumentation. This evolution leaves the classical viewpoint of the device devoted to the measure of a single quantity towards the idea of a computerized data-processing system which acquires a large amount of data by which it is possible to find information necessary to determine the desired quantities.

With these aims, the employment of the computer and the optimization of signal-processing techniques represent the correct approach to obtain a drastic reduction of costs and an increasing flexibility in dedicated applications. With a view to obtain higher performance in terms of accuracy, flexibility and speed in estimating the variables of interest it is fundamental to qualify the uncertainty of the analog-to-digital (A/D) converters, data acquisition plug-in boards and, in general, instruments and systems based on digital recording of waveforms must be characterized.

A lot of works and documents deal with this matter, but many limitations must still be overcome; even the most authoritative technical document about this field ([1]-[2]) give a wealth of useful definitions, test methods and mathematical analyses, but do not produce a complete qualification of the A/D converter.

Besides the scientific literature abounds of test methods that analyze particular effects or error phenomena in digital recorders and it is not uncommon to run into puzzling questions and even contradictions when trying to arrange them together. Finally, we can assert that a complete *behavioral model* which takes into account *all* error phenomena of A/D converters simply cannot be found.

With these aims, *it is necessary the use of a common and univocal standard framework to analog-to-digital converters test*; this solution, as stressed by some researchers ([3]-[4]), would be very profitable and desirable.

The purpose described above requires the choice of a suitable programming language which must be very usual, computationally powerful and user-friendly; even if many other software package are available, the MATLAB program seems a very favourable choice ([3]).

II GPIB4MATLAB & NIDAQ4MATLAB TOOLBOXES: AN OVERVIEW

The toolboxes presented here by the authors, named *GPIB4Matlab* and *NIDAQ4Matlab*, may represent a common platform that can permit fast and simple implementation of all the tests and the algorithms described in the Standard or proposed by the researchers around the world.

Particularly they have already permitted various metrological characterization of different digital devices: A/D converters, digital boards, computerized data-processing system and so on. Moreover they can be used to easily control both a large variety of Automatic Test Equipment (ATE) and Data acquisition (DAQ) boards.

The platform used to produce this framework was a desktop PC with Windows 98 operating system so that a simple and user-friendly access to ISA/PCI bus devices, PXI systems and all instruments with a GPIB interface is possible.

The fundamental characteristics of the toolboxes proposed are:

- simple and efficient software structure (see figures 1 and 2);
- every function returns a clear and complete status indicator (a MATLAB's structure) of the performed operation; the data reported in this structure permits to search and apply the necessary corrections to the procedure. For example, the following lines show what is the content of this structure when we call the NIDAQ4Matlab's function DAQ_Rate:

```
» [status, timebase, sampleInterval,
actRate] = DAQ_Rate(1035)
```

```
status = Sender: 'DAQ_Rate'
           Code: 0
           Name: 'noError'
    Description: 'Function successfully
executed.'

    timebase = 1
    sampleInterval = 966
    actRate = 1.0352e+003
```

To be clear, the function DAQ_Rate returns some parameters required to configure and start any DAQ procedure.

- proper operation over many MATLAB's versions (starting from 4.0);
- their use is immediate and do not require any additional programming skills to the user different than the simple ones necessary to implement ordinary algorithms;
- it is possible (and simple) to extend the actual implemented functionalities; to do this only basics C language programming skills are required;
- the project can be converted in an *Open Source* form and every researcher can contribute to its advance.

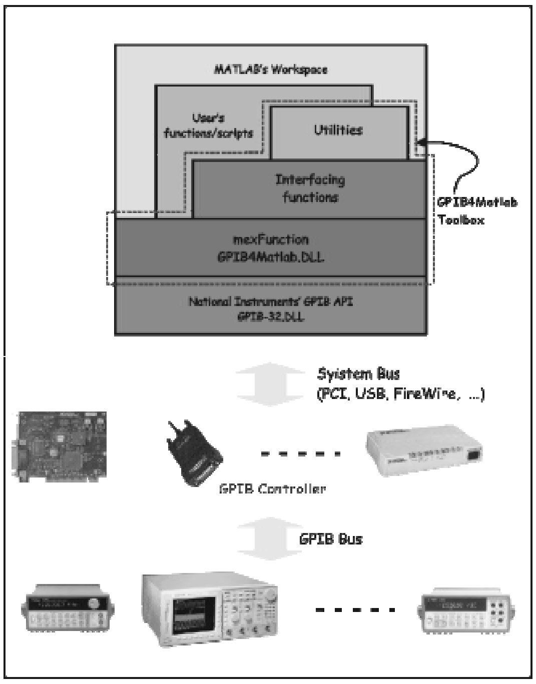


Fig. 1 - GPIB4Matlab block diagram.

Obviously there are also many aspects of these toolboxes that can limit their applicability especially with respect to the commercially available counterparts (i.e. the MathWorks' Data Acquisition Toolbox® and Instrument Control Toolbox®); the three main drawbacks are:

- They are not commercial products and, consequently, they do not have a guaranteed support;
- They (at the moment) are not provided of a printed manual; there is only a complete on-line help in the typical style of MATLAB's functions;
- They are not intended to be *interoperable* with DAQ and GPIB hardware different than National Instruments' products. These toolboxes were originally developed to respond to precise research needs of the authors and when the MathWorks' products were not available yet; at that time (2nd semester 1999) the DAQ devices and GPIB controllers hardware available in the authors' laboratory were only provided by National Instruments.

III GPIB4MATLAB: SOME DETAILS

The software structure of *GPIB4Matlab* is depicted in Fig. 1. From this figure it follows the great simplicity of the implementation and, consequently, the run-time effectiveness of the toolbox.

The careful implementation of all of the functions of NI's GPIB API (Application Programming Interface, i.e. the main software interface between user's applications and the underlying hardware) has permitted the use of GPIB4Matlab to program and control various stand alone instruments with a GPIB interface as well as more complex ATE systems composed of various instruments connected between them and at a common GPIB backbone or even connected to an Ethernet LAN with a proper protocol converter.

As it appear from the Fig. 1, the data exchange between the MATLAB's workspace and the instruments connected to the GPIB bus can occur in three different (and totally mixable) ways:

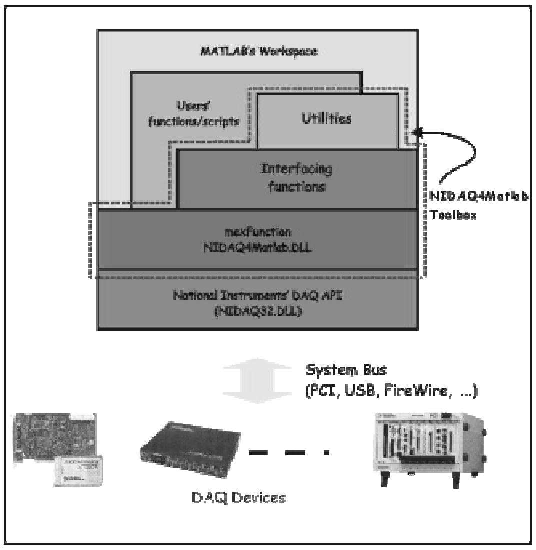
1. we can use a *direct call* to the mexFuncton GPIB4Matlab.dll (a Windows' dynamic linking library compiled according to the MathWorks' specifications), which is also the main component of GPIB4Matlab;
2. we can call one of the functions contained in the *interfacing functions* block, all of which are m-files with a very simple structure and the same name of the corresponding NI's GPIB function; the only purpose of these functions is to simplify the draft of the program code. Moreover the similarities between the MATLAB's instruction set and program structures with that of the C language, permits an easy translation of many useful example code enclosed to the NI's GPIB products. This process is

simplified by the availability of a MAT file incorporating all of the NI-GPIB predefined constants.

3. we can use the functions contained in the *Utilities* block. In this block there are many useful functions (e.g. there is a function to initialise the data exchange with an instrument, a function to send a query or a command to a previously initialised instrument, etc.). There are also some useful functions that may be used to send commands and queries *directly* to the instruments from the MATLAB's command line *without the need of any other preliminary initialisation*; with these functions it is possible to program and query the instruments interactively from the command line.
As an example of the great flexibility achievable with the use of these functions, the following is the command we can use to identify an IEEE488.2 compliant GPIB instrument:

```
» [status, id] =
DirectQuery_GPIB_Instrument(0, 1, 0,
'*IDN?', T3s, 1, 0, 'char')
```

With the previous command we obtain the instrument's identifier directly in a string in the MATLAB's workspace (note that `T3s` is one of the constant defined by NI for its GPIB products and is used to instruct the appropriate GPIB function to wait for the answer for a maximum of three seconds). Obviously we can use the same syntax to query the instrument about any other internal parameter accessible by an instruction of its own set. As an example this is the query we can use with a LeCroy's LT262 digital oscilloscope to obtain the vertical offset of its first channel:



MATLAB's Workspace
Users' functions/scripts
Utilities
Interfacing functions
NIDAQ4Matlab Toolbox
mexFunction NIDAQ4Matlab.DLL
National Instruments' DAQ API (NIDAQ32.DLL)
System Bus (PCI, USB, FireWire, ...)
DAQ Devices

Fig. 2 - NIDAQ4Matlab block diagram

```
» [status, C1_Vo] =
DirectQuery_GPIB_Instrument(0, 1, 0,
'C1:VERTICAL_OFFSET?', T3s, 1, 0, 'char',
'str2double')
```

In this case the variable `C1_Vo` contains the value of the required offset directly as a double data type number, and any other type cast is unnecessary; all the work is done in the background by `DirectQuery_GPIB_Instrument`.

IV NIDAQ4MATLAB: SOME DETAILS

As it appear at a first glance, the software structure of NIDAQ4Matlab (Fig. 2) is almost identical to that of GPIB4Matlab; obviously there are only formal analogies between these two products: the code on which they are based is completely different.

In this case the NIDAQ4Matlab's functions create an interface layer between the NIDAQ's API (NIDAQ-32.DLL dynamic linking library) and the MATLAB's workspace.

Once again, as in the case of GPIB4Matlab, the similarities between the MATLAB's instruction set and that of the C language permits the almost immediate translation of many useful example code available with NI's DAQ products.

Obviously the maximum flexibility reached when one use the powerful functionalities offered by many MATLAB's advanced toolboxes in conjunction with specific functions of the NI's DAQ products "imported" in MATLAB thanks to NIDAQ4Matlab.

For example, the implementation of FIR/IIR digital filtering applications with filters designed by MATLAB under user control, with sampling frequencies up to 100 kHz and data block sizes down to 4096 samples/block has been simple and has produced the results in real time thanks to the Double Buffering capabilities offered by NI's DAQ API.

The flexibility achievable with NIDAQ4Matlab (often in conjunction with GPIB4Matlab) has been experimented in almost two years of applied research of the authors in the field of dynamic characterization of ADCs; excluding some (absolutely marginal) adjustments, there was not the need for any modification of the software structure of the two toolboxes.

V NIDAQ4MATLAB & GPIB4MATLAB VS. MATHWORKS' PRODUCTS

The MATLAB Toolboxes presented here are substantially different (in many aspects) from their commercially available counterparts from MatWorks Inc. (Instrument Control Toolbox [5] and Data Acquisition Toolbox [6]).

Obviously there are many features of the latter that make them highly sophisticated products; one of these features is the support offered to different hardware products (currently the whole range of National Instruments' DAQ and GPIB products, some products from Agilent Technologies and the standard audio boards for PC/Windows platform).

However what seems to be only an advantage could become a serious problem when these toolboxes must be used to implement some complex algorithm that require the use of some advanced and specific functionality of an hardware product; in fact the homogeneity imposed by the MathWork's toolboxes in the access procedures can easily cause the loss of control over these specific functionalities. As an example, we can see ([6]) that the user of a NI's multifunctional DAQ product does not have access to the digital counters of the board because this is a functionality not supported by all of the manufacturers.

The solution proposed by MathWorks to this kind of problem as well as to the problem of extension of support to hardware from others OEMs (i.e. the *Data Acquisition Toolbox Adaptor Kit*), in the author's opinion, does not help the majority of researcher and scientists involved in the Measurements field, simply because their use requires advanced programming skills (operating knowledge of C++, of Microsoft's COM architecture and of the Active Template Library).

The toolboxes proposed here, on the contrary, are based on an *open source code, very simple and easily extendable*; only simple C language basics skills are required. It is worth to underline that, besides these very important characteristics, one other major advantage of these toolboxes over the corresponding ones proposed by the MathWorks Inc., is that they can be installed and correctly used with almost all versions of MATLAB, starting from 4.0 and ending to the current version (6.1).

VI CONCLUSIONS

Two originally and independently developed MATLAB toolboxes dedicated to the control of GPIB instrumentation and data acquisition hardware have been presented. They can be two useful tools for all the scientists involved in the Measurements field and aiming to obtain direct access to the measurement hardware from the MATLAB workspace.

They have some significant advantage over the commercially available counterparts from MathWorks but also some drawbacks.

VII REFERENCES

[1] IEEE Standard 1057 for Digitizing Waveform Recorders, Dec. 1994.

[2] IEEE Standard 1241 for Terminology and Test Methods for Analog-to-Digital Converters, Draft, May 1999.

[3] J. Markus, I. Kollar, Standard Framework for IEEE-STD-1241 in MATLAB, *Proceedings of IMTC/01 Conference,* Budapest, May 2001, Hungary, pp. 1847-1852.

[4] J. J. Blair, Sine-fitting software for IEEE standard 1057 and 1241, *Proceedings of IMTC/99 Conference*, Venice, May 1999, Italy, pp.1504-1506.

[5] MathWorks Inc., Instrument Control Toolbox User's Guide
http://www.mathworks.com/products/instrument

[6] MathWorks Inc., Data Acquisition Toolbox User's Guide
http://www.mathworks.com/products/daq