3

A Matlab® Approach for Implementing Control Algorithms in Real-Time: RTWT

Andres Hernandez, Adrian Chavarro and Robin De Keyser
Ghent University, Dep. Electrical energy, Systems and Automation, Technologiepark 913, B-9052 Gent, Belgium

1. Introduction

The literature about real-time systems presents digital control or computer controlled systems as one of its most important practical applications. However, it is very difficult to find in these textbooks real-time control aspects (Gambier, 2005). It seems to be more natural that these applications should be treated as part of digital control courses. In spite of that, control system literature rarely includes extensively the real-time subject and it does normally not pay much attention to real-time implementation aspects. Nevertheless, in practice there is the requirement for the design of control algorithms which run in the specified time without detriment to quality and functionality.
Thanks to the improvement in some software products, new control algorithms can be designed and tested in real life practical applications very quickly with excellent quality, giving a new optic to the control engineering courses. Software like Matlab/Simulink with its RTW (Real Time Workshop) and the RTWT (Real Time Windows Target) give us the opportunity to work from an easy interface and produce good results, while one deals with time-critical applications.
This chapter attempts to give a guide for the implementation of real-time control systems, using the RTWT toolbox, as a practical tool for students in control engineering. A digital PID controller will be tested in a real-life application (Hernandez *et al.*, 2011), in order to present a description of the implementation procedure.
The outline of the paper is as follows: a brief introduction to the application problem is depicted in the next section. Definitions and characteristics of real-time systems are described in Section 3. Section 4 treats the implementation of real-time controllers using RTWT in Matlab®. Section 5 is devoted to the configuration of RTWT for our specific application as an example, including some experimental results. Final conclusions are drawn in the last section.

2. Application description: Lungs function test

Non-invasive lung function tests are broadly used for assessing respiratory mechanics (Northrop, 2002; Oostveen *et al*., 2003). Contrary to the forced maneuvers from patient side and special training for the technical medical staff necessary in spirometry and in body plethysmography (Pellegrino *et al.*, 2005), the technique of superimposing air pressure

oscillations is simple and requires minimal cooperation from the patient, during tidal breathing (Oostveen *et al.*, 2003). Among the air pressure oscillation techniques for lung function testing, the most popular one is that of Forced Oscillation Technique (FOT). FOT uses a multisine signal to excite the respiratory mechanical properties over a wide range of frequencies, usually between 4-48Hz (Oostveen *et al.*, 2003).
Using measurements of air pressure and air flow, it is possible to extract information regarding the human respiratory input impedance. However this is a linear approximation of a nonlinear system, hence the output will depend on the input's amplitude and frequency (Schoukens & Pintelon, 2001). It is therefore important to ensure that the desired signal to be applied at the patient's mouth will be delivered by the lung function testing device, without introducing distortions and nonlinear effects. Hence, a closed loop control system is necessary, to continuously monitor and correct the errors between the desired input signal and the one delivered by the device at the patient's mouth.
In practice, in order to send a sinusoidal signal of 50 Hz it is necessary to have a sample rate of at least 500 Hz, which means 10 samples per sinusoid period. The corresponding sampling time is 0.002 seconds, which can be delivered by the DAQcard 6024E used in this application. In this particular example, it is not possible to work with Matlab running in normal operation, because the delay for calculations in the closed loop is about 14ms, much higher than the desired sample rate. A solution to overcome this limitation consists in using RTWT to assign some resources of the system exclusively for this task, ensuring the desired sampling time.

3. Definitions and characteristics: Real-time systems

Nowadays, thanks to the computational and graphical power of modern computers, more flexible control systems including higher-level functions and advanced algorithms can be implemented successfully in real systems. Furthermore, most current complex control systems could not be implemented without the application of digital hardware; moreover these systems now contain not only physical components but also algorithms, which must be programmed, i.e. software is now included in the control loop. This leads to new aspects to take into account by designing control systems.
When one builds a control algorithm in any programming language, one normally assumes that sampling is uniform, periodic and synchronous. However, that is not realistic since the control algorithm also consumes some time producing a control or feedback delay (control or feedback latency), i.e. a delay between a sampling instant and the instant at which a control-signal value is applied to the actuator. Also the computational time of control algorithms can change from one sampling instant to other (e.g. hybrid controller with controller switching mechanism, event based controllers, adaptive controllers with on-line parameter update, etc.). This variation in the delay is called control jitter (according to the IEEE, jitter is "the time-related abrupt, spurious variation in the duration of any specified related interval") (Gambier, 2005)
It is important to clarify also some other aspects about the meaning of 'real-time', although it is a vast field and therefore a complete discussion about the topic is outside the scope of this document. Fast computing aims at getting the results as quickly as possible, while real-time computing aims at getting the results at a prescribed point of time within defined time tolerances. This idea explains how real-time is not just for fast systems, but for any control loop where a task must be achieved in a specific time.

4. Implementation of real-time controllers using RTWT[1]

4.1 Overview on RTWT

Real-Time Windows Target™ rapid prototyping software is a PC solution for prototyping and testing real-time systems. Real-Time Windows Target software uses a single computer as a host and target. On this computer, you use the MATLAB® environment, Simulink® software, and Stateflow® software (optional) to create models using Simulink blocks and Stateflow diagrams.

After creating a model and simulating it using Simulink software in *normal mode*, you can generate executable code using RTW and your C/C++ compiler. Then you can run your application in real time with Simulink in *external mode*.

Integration between Simulink external mode and Real-Time Windows Target software allows you to use your Simulink model as a graphical user interface for

- *Signal visualization* — Use the same Simulink Scope blocks that you use to visualize signals during a non-real-time simulation to visualize signals while running a real-time application.
- *Parameter tuning* — Use the Block Parameter dialog boxes to change parameters in your application while it is running in real time.

Typical uses for Real-Time Windows Target applications include

- *Real-time control* — Create a prototype of automotive, computer peripheral, and instrumentation control systems.
- *Real-time hardware-in-the-loop simulation* — Create a prototype of controllers connected to a physical plant. For example, the physical plant could be an automotive engine. Create a prototype of a plant connected to an actual controller. For example, the prototyped plant could be an aircraft engine.
- *Education* — Teach concepts and procedures for modelling, simulating, testing real-time systems, and iterating designs

4.2 Real time kernel

Real-Time Windows Target software uses a small real-time kernel to ensure a deterministic sampling rate in the application. The real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the PC clock as its primary source of time. Some important aspects regarding the kernel operation includes:

- *Timer interrupt* — The kernel intercepts the interrupt from the PC clock before the Windows® operating system receives it. The kernel then uses the interrupt to trigger the execution of the compiled model. As a result, the kernel is able to give the real-time application the highest priority available. To achieve precise sampling, the kernel reprograms the PC clock to a higher frequency. Because the PC clock is also the primary source of time for the Windows operating system, the kernel sends a timer interrupt to the operating system at the original interrupt rate.
- *Scheduler* — RTWT lets you to work with a single sample rate or with multiple/different sampling rates in your model. Each sampling rate is defined like a task and is clocked by a simple scheduler that runs the executable. The maximum

[1] Parts of the text has been subtracted from the "Real-Time Windows Target User's Guide", Copyright 1999 by The MathWorks, Inc. http://www.mathworks.com/products/rtwt/

number of tasks is 32, and faster tasks have higher priorities than slower tasks. For example, a faster task can interrupt a slower task.

- *Communication with hardware* — The kernel interfaces and communicates with I/O hardware using I/O driver blocks, and it checks for proper installation of the I/O board. If the board has been properly installed, the drivers allow your real-time application to run.
- *Simulink external mode* — Communication between Simulink software and the real-time application is through the Simulink external mode interface module. This module talks directly to the real-time kernel, and is used to start the real-time application, change parameters, and retrieve scope data.
  Opening a dialog box for a source block causes simulation to pause. While simulation is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow simulation to continue.

4.3 System concepts

Non-real time simulation

When you run your Simulink model using *normal mode*, Simulink software uses a computed time vector to step your model. After the outputs are computed for a given time value, the Simulink software immediately repeats the computations for the next time value. This process is repeated until it reaches the stop time.
Because this computed time vector is not connected to a hardware clock, the outputs are calculated in non-real-time as fast as your computer can run. The time to run a simulation can differ significantly from real time.

Real time execution

For real-time execution on your PC, you must use Simulink *external mode*, Real-Time Workshop code generation software, Real-Time Windows Target software, and a C/C++ compiler, to produce an executable that the kernel can run in real time. This real-time application uses the initial parameters available from your Simulink model at the time of code generation.
If you use continuous-time components in your model and create code with RTW code generation software, you must use a fixed-step integration algorithm. Based on your selected sample rate, RTWT software uses interrupts to step your application in real time at the proper rate. With each new interrupt, the executable computes all of the block outputs from your model.

Development process

With Real-Time Windows Target rapid prototyping software, one can use a desktop PC with the MATLAB environment, Simulink software, Real-Time Workshop code generation software, and Real-Time Windows Target software to:

1. *Design a control system* — Use the MATLAB environment and Control System Toolbox™ software to design and select the system coefficients for your controller.
2. *Create a Simulink model* — Use Simulink blocks to graphically model your physical system.
3. *Run a simulation in non-real time* — Check the behavior of your model before you create a real-time application. For example, you can check the stability of your model.

4. *Create a real-time application* — Real-Time Workshop code generation software creates C code from your Simulink model. The C/C++ compiler compiles the C code to an executable that runs with the Real-Time Windows Target kernel.
5. *Run an application in real time* — Your PC is the target computer to run the real-time application.
6. *Analyze and visualize signal data* — Use MATLAB functions to plot data saved to the MATLAB workspace or a disk.

Simulink external mode

External mode requires a communication interface to pass external parameters. On the receiving end, the same communications protocol must be used to accept new parameter values and insert them in the proper memory locations for use by the real-time application. In some Real-Time Workshop targets such as Tornado/VME targets, the communications interface uses TCP/IP protocol. In the case of a Real-Time Windows Target application, the host computer also serves as the target computer. Therefore, only a virtual device driver is needed to exchange parameters between the MATLAB environment, Simulink memory space, and memory that is accessible by the real-time application.
Signal acquisition — You can capture and display signals from your real-time application while it is running. Signal data is retrieved from the real-time application and displayed in the same Simulink Scope blocks you used for simulating your model.
Parameter tuning — You can change parameters in your Simulink block diagram and have the new parameters passed automatically to the real-time application. Simulink external mode changes parameters in your real-time application while it is running in real time.

Data buffer and transferring data

At each sample interval of the real-time application, Simulink software stores contiguous data points in memory until a data buffer is filled. Once the data buffer is filled, Simulink software suspends data capture while the data is transferred back to the MATLAB environment through Simulink external mode. Your real-time application, however, continues to run. Transfer of data is less critical than maintaining deterministic real-time updates at the selected sample interval. Therefore, data transfer runs at a lower priority in the remaining CPU time after model computations are performed while waiting for another interrupt to trigger the next model update.
Data captured within one buffer is contiguous. When a buffer of data has been transferred, it is immediately plotted in a Simulink Scope block, or it can be saved directly to a MAT-file using the data archiving feature of the Simulink external mode.
With data archiving, each buffer of data can be saved to its own MAT-file. The MAT-file names can be automatically incremented, allowing you to capture and automatically store many data buffers. Although points within a buffer are contiguous, the time required to transfer data back to the Simulink software forces an intermission for data collection until the entire buffer has been transferred and may result in lost sample points between data buffers.

4.4 Installation of the software RTWT

Once Matlab® is installed all Real-Time Windows Target software is copied onto your hard drive, but the Real-Time Windows Target kernel is not automatically installed into the operating system. You must install the kernel before you can run a Real-Time Windows

Target application. Installing the kernel configures it to start running in the background each time you start your computer. The kernel installation is done in the workspace by typing:

```
>> rtwintgt – install
```

You can also use the command **rtwintgt -setup** to install the kernel. The MATLAB Command Window displays one of these messages:

```
>> You are going to install the Real-Time Windows Target kernel.
Do you want to proceed? [y] :
```

or:

```
>> There is a different version of the Real-Time Windows Target kernel installed.
Do you want to update to the current version? [y] :
```

Type **y** to continue installing the kernel, or **n** to cancel installation without making any change. If you type **y**, the MATLAB environment installs the kernel and displays the message:

```
>> The Real-Time Windows Target kernel has been successfully installed.
```

If a message appears asking you to restart your computer, do so before attempting to use the kernel, or your Real-Time Windows Target model will not run correctly. After installing the kernel, verify that it was correctly installed by typing:

```
>> rtwho
```

The MATLAB Command Window should display a message that shows the kernel version number, followed by performance, timeslice, and other information.

```
>>Real Time Windows Target version 1.00 (C) The MathWorks, Inc. 1994-2010
Running on Multiprocessor APIC computer
MATLAB performance = 98.5%
Kernel timeslice period = 0.999 ms
```

Matlab specifies the performance of the running application on the actual PC and the used sampling time. It is desirable to execute your applications near 100% performance, is not recommended to use values of performance near to 50% because the switching execution time will decrease in the real time windows target in order to attend other programs in the Operative system.
Once the kernel is installed, you can leave it installed. The kernel remains idle after you have installed it, which allows the Windows operating system to control the execution of any standard Windows based application, including Internet browsers, word processors, the MATLAB environment, and so on. The kernel becomes active when you begin execution of your model, and becomes idle again after model execution completes.
The Real-Time Windows Target requires a C compiler which is not included in the installation in MATLAB. To choose the compiler to use it is necessary to type the following command in the workspace:

```
>> mex –setup
```

The following dialog will appear:

```
>> Would you like mex to locate installed compilers [y]/n? y
```

Select a compiler:

```
[1] Intel Visual Fortran 9.1 (with Microsoft Visual C++ 2005 linker) in
 C:\Program Files\Intel\Compiler\Fortran\9.1
[2] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2007b\sys\lcc
[3] Microsoft Visual C++ 2005 in
 C:\Program Files\Microsoft Visual Studio 8
[0] None
```

After you choose your compiler for instance, Compiler: 2, the following dialog will appear:

```
>> Please verify your choices:
 Compiler: Lcc-win32 C 2.4.1
Location: C:\PROGRA~1\MATLAB\R2007b\sys\lcc
>> Are these correct?([y]/n): y
Done . . .
```

After you confirm your choice typing *y* the process finish it. You can use any PC-compatible computer that runs Microsoft® Windows XP 32-bit, or Microsoft Windows Vista ™ 32-bit. Your computer can be a desktop, laptop, or notebook PC.

4.5 Hardware I/O boards

Real-Time Windows Target applications use standard and inexpensive I/O boards for PC-compatible computers. When running your models in real time, RTWT captures the sampled data from one or more input channels, uses the data as inputs to your block diagram model, immediately processes the data, and sends it back to the outside world through an output channel on your I/O board.

Real-Time Windows Target software provides a custom Simulink block library. The I/O driver block library contains universal drivers for supported I/O boards. These universal blocks are configured to operate with the library of supported drivers. This allows easy location of driver blocks and easy configuration of I/O boards. You drag and drop an universal I/O driver block from the I/O library the same way as you would from a standard Simulink block library. And you connect an I/O driver block to your model just as you would connect any standard Simulink block.

You create a real-time application in the same way as you create any other Simulink model, by using standard blocks and C-code S-functions. You can add input and output devices to your Simulink model by using the I/O driver blocks from the **rtwinlib** library provided with the Real-Time Windows Target software. This library contains the blocks depicted in figure 1.

The Real-Time Windows Target software provides driver blocks for more than 200 I/O boards. These driver blocks connect the physical world to your real-time application:

- Sensors and actuators are connected to I/O boards.
- I/O boards convert voltages to numerical values and numerical values to voltages.
- Numerical values are read from or written to I/O boards by the I/O drivers.

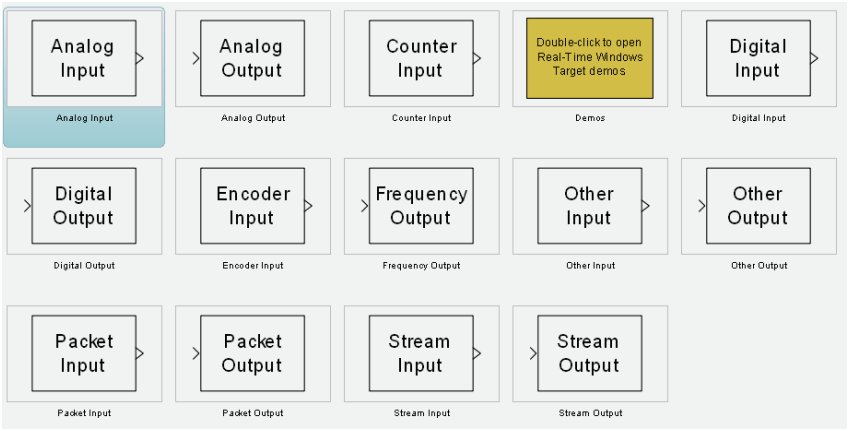


Fig. 1. Library Real Time Windows Target

5. Application of the real-time control in a lung function test device

By following the procedure described in section 4.3 the Simulink scheme will be implemented and configured. The computer characteristics used in this example are: Intel core duo processor of 1.73 GHz with 3Gb of RAM , Windows Xp 32 Bits, and expressCard to PCMCIA adapter.

5.1 Implementing the simulink model

The communication between the computer running Matlab and the FOT device is made by using the National Instruments DAQCard 6024E (which is recognized by Matlab and supported for real time applications). The corresponding Simulink model was developed in order to send and receive signals to/from the real FOT system, as depicted in figure 2.

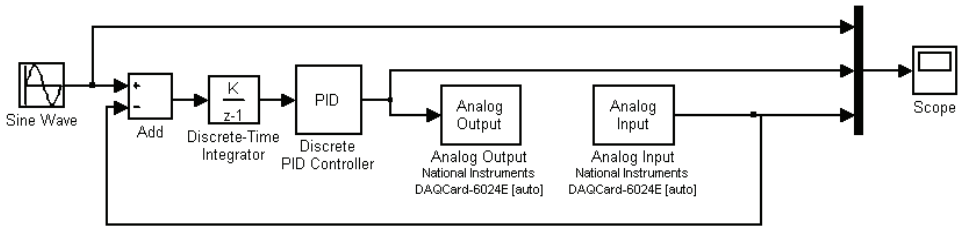


Fig. 2. Simulink model for the RTWT application

At this point it is recommendable to create a new folder in your current directory, because the compilation procedure creates several files and this will let you work in an orderly manner.

In this application our interest is to test a discrete PID controller; its parameters have been previously tuned, and its design will not be presented in detail.

Configuration of the simulation parameters

Once the model has been created, we must set the simulation parameters. By pressing the combination of keys '*Ctrl+E*' the configuration parameters window will appear (figure 3).

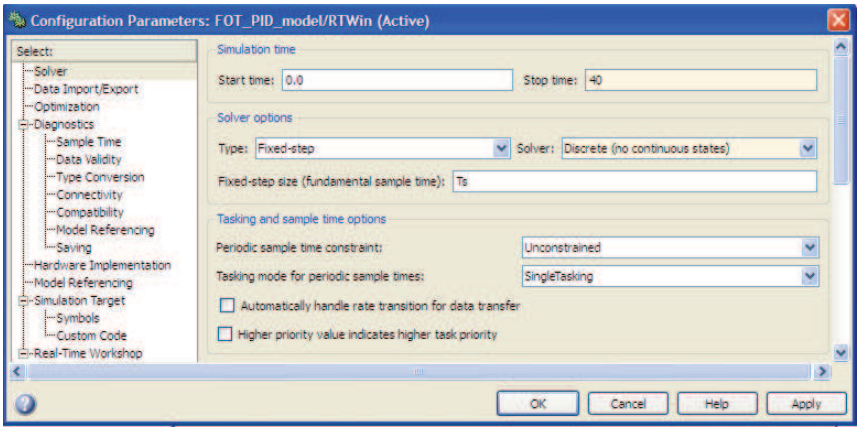


Fig. 3. Solver configuration

The first parameter to configure is the solver. We can choose the stop time of the simulation, between a fixed value or to run indefinitely by typing 'inf'. In this application a stop time of 40s was chose. In the solver options you can choose between variable or fixed step, in this application what we want is to guarantee a fixed sampling time, hence, we choose the type 'Fixed-step' and as solver the 'discrete (no continuous states)'.

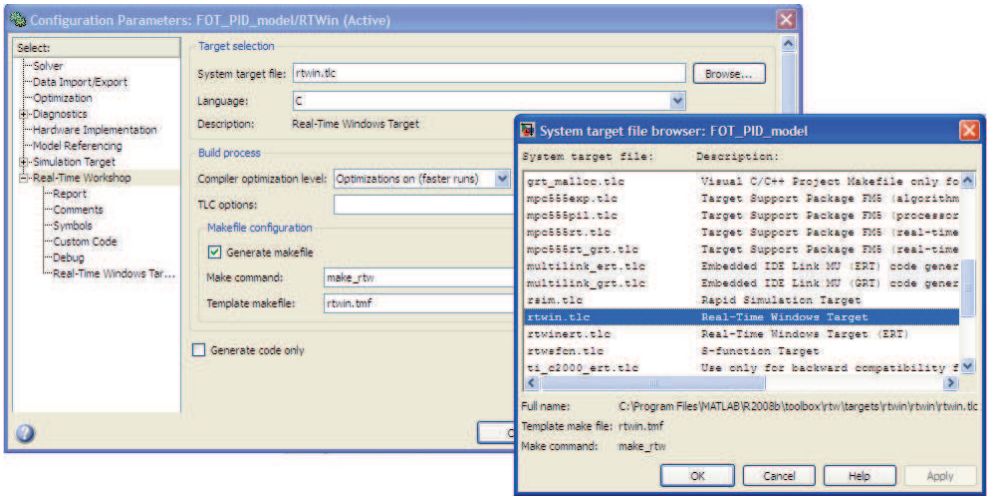


Fig. 4. Configuration System Target

The next step is to configure the target, for this we must select the option 'Real-time Workshop' as presented in figure 4. The first option to select is the system target file, there

are several options available when we press the 'Browse' button, however we must select 'rtwin.tlc' which is the Real-Time Windows Target. The language can be selected as C or C++, we choose C language by default. We accept these changes and return to our model in Simulink. At this point we can choose the simulation as external mode, as depicted in Figure 5. Remember to save your model by pressing the keys 'ctrl+S'

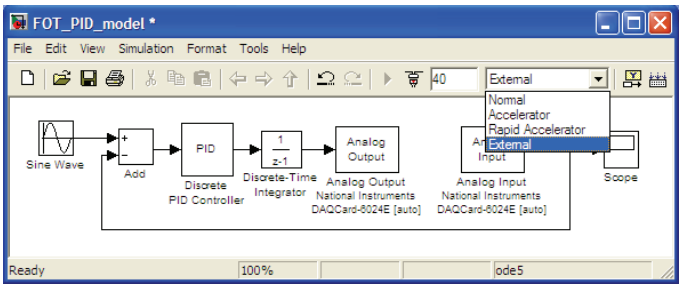


Fig. 5. Configuring the simulation in External mode

Configuring the analog input and output

After our simulation parameters has been configured, then we can continue with the process interfacing. By double clicking in the analog output block in our Simulink model, the configuration window will appear as depicted in Figure 6. In this window we must select our hardware board, in this case the National Instruments acquisition board DAQCard-6024E. The sampling time is selected as '*Ts*', which can be previously defined in the workspace as *Ts*=0.002. In this step also the output range can be configured, which is in our case from -10 V to 10 V. Some initial and final values can be established at this point, for the cases when we need that the DAQ board remains with some value after the simulation stops. It is possible to test our hardware to verify that there are not communication problems between Simulink and our external hardware. By pressing the '*Board Setup*' button a new window will appear, and by pressing the '*Test*' Button we can test all the inputs and outputs available in our board. To configure the analog input the same procedure must be followed, the only difference is that you'll not find the 'initial' and 'final' value parameters available in the analog output.

Discrete PID configuration

For this application we have tuned the parameters of a PID controller by means of the KCR algorithm (Hernandez *et al*, 2010); this procedure will not be described here, because our interest is to present how to use the Real-Time Windows Target toolbox.
The discrete PID controller used in this work can be found in the: SimPowerSystems/Extra Library/Discrete Control Blocks/Discrete PID Controller (Figure 7). This block implements a discrete PID controller, where the *Kp*, *Ki*, *Kd* and sampling time *Ts* parameters can be configured. There are also some other options available, e.g. the time constant for the derivative action or the constraints in the output, which have been selected as 1000 and [-1 1] respectively.

Scope configuration to display and save data

Until now, the simulation parameters, PID and I/O have been configured; nevertheless, another important issue to solve is how to save the data on our hard disk. By double clicking

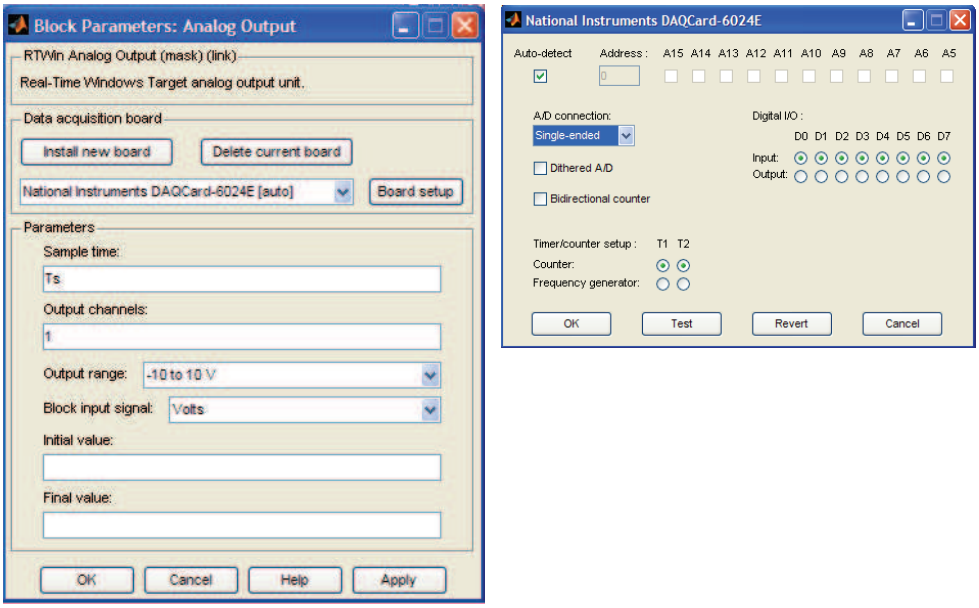


Fig. 6. Configuration window Analog input/output

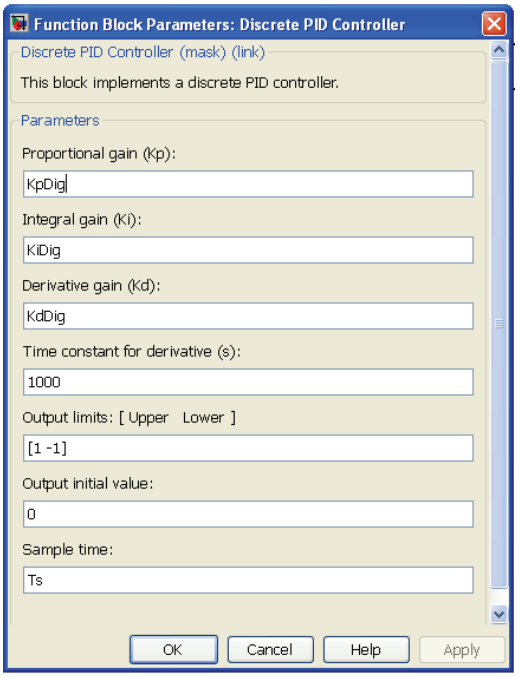


Fig. 7. Configuration Window Discrete PID

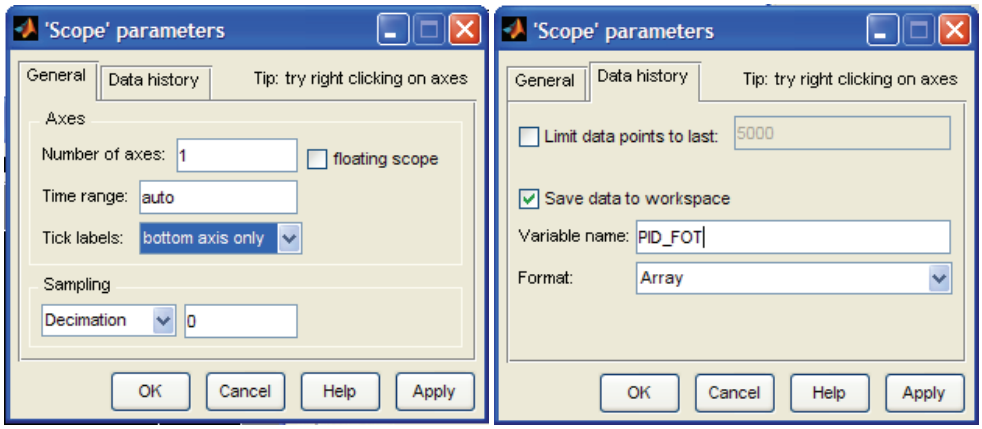


Fig. 8. Scope Configuration Window to display and save the data

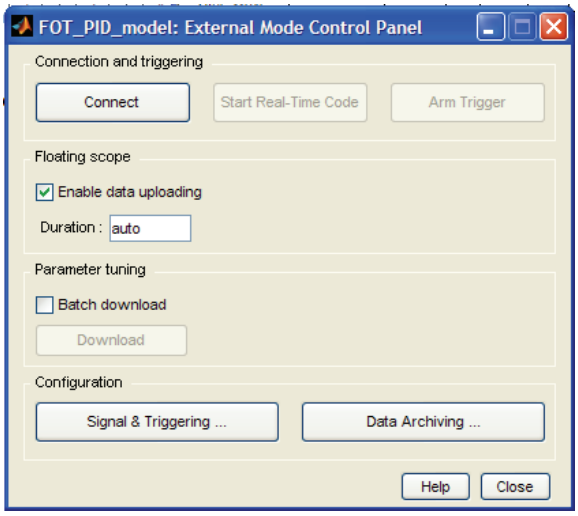


Fig. 9. External mode control panel

in the scope a new window will appear, there, the sampling will be chosen as '*Decimation*'=0, this is done to consider this block as an analog block because the triggering will not be done by this block. By selecting the next tap '*Data history*' (Figure 8-right), we must avoid to limit the data points and instead of this, we select save data to workspace. We type a name for the variable we want to save and then we choose array as data format and we press '*ok*' in all the windows.

Going back to our Simulink model, we choose in the toolbar the option Tools/External Mode Control Panel, a new window will appear as depicted in Figure 9.

The first step is to press the '*Signal & Triggering*' button; in this new window we must configure the trigger as '*manual*' and the mode in '*normal*'. The duration is the number of samples that you are going to simulate. A very important detail when we choose this value

is to know how much our sampling time is, how long our simulation will be and that Real-Time Windows Target takes zero as an extra value. By taking this into account it is possible to see that in 40s at 2ms sampling time, we need to save 2000 samples, however, taking into account the sample at time zero, finally we choose 2001 as parameter.
After choosing the signal and trigger options, we press the '*Data Archiving*' button; in this new window we have to enable '*archiving*' and then type de directory where we want to save our data and the name of the file (Figure 10). If we have more than one variable to save then an array will be saved in this address with the name we chose, the first column is always the time vector and the next columns each one of our variables. In this application we have used a mux block to put all the measured variables into one scope (see Figure 2), however it is also possible to have one scope for each variable.

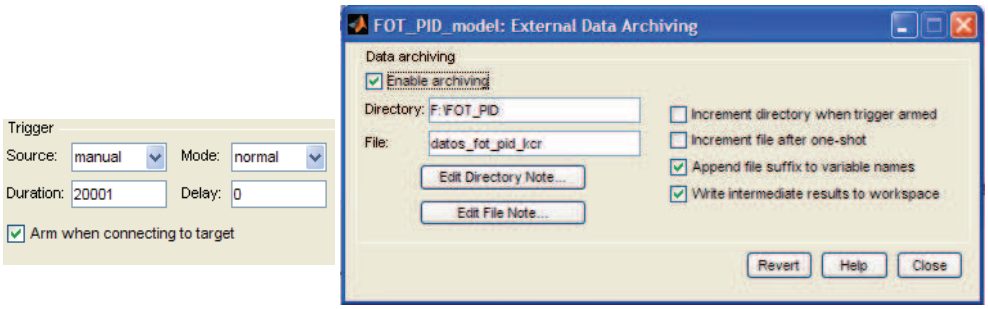


Fig. 10. External Signal & Triggering and External Data Archiving configuration window

Once the simulink model has been configured, then it has to be saved to accept all the changes and then compilate it, by using the combination of keys *ctrl+B*.
Once all procedures have been completed, then we can press the button '*Connect to Target*' and then '*Start Simulation*', to run the simulation as depicted in figure 11.

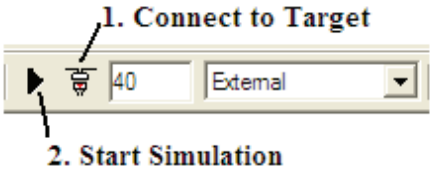
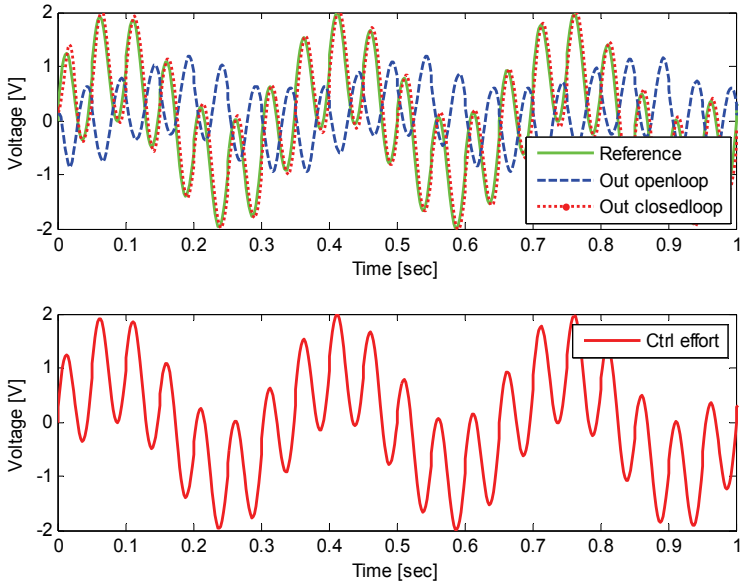


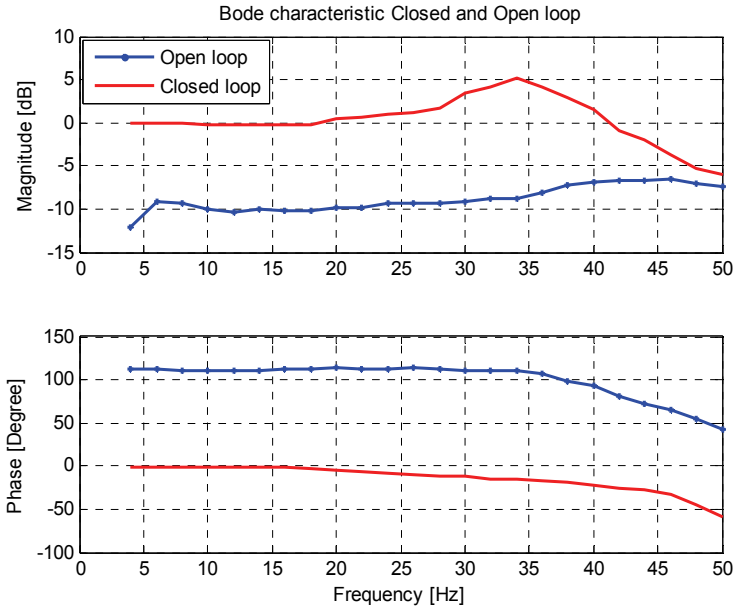
Fig. 11. Run simulation

5.2 Experimental results

By using the hardware and software described in section 4, it is possible to do the open loop and closed loop identification using the Chirp-TFA algorithm (Ionescu C., *et al.*, 2010). Some results are given by means of Bode plots in Figure 12. It can be observed that the bandwidth (frequency at -3dB) of the system is about 45Hz.
In order to be able to follow a reference signal in a closed loop it is necessary that the magnitude of the closed loop remains around 0dB and the phase around 0^{o} in the frequency-range of interest. From the Bode plot in Figure 12-right for the closed loop, we can observe that the results are in agreement with the expected bandwidth, and that the controller



(a)



(b)

Fig. 12. Open and Closed loop characteristics. a) Performance in time. b) Performance in frequency

performs satisfactorily. This result is also visible when a comparison in time domain between open loop and closed loop is done. The controller avoids distortions and nonlinear effects at the output of the lung function device; the desired signal will be successfully delivered at the patient's mouth as depicted in Figure 12-left.

6. Conclusions

In this work an interactive and effective tool to design control loops in real-time has been presented. A real system was used as an example to discuss the importance of real-time, and clarify some fundamental aspects about the meaning of real time in control. An introduction to real-time control from an educational and practical point of view has been given. Some well-known misconceptions coming from the control system community were discussed. The relevance of the real-time implementation has been exposed by implementing the closed loop control of a medical device for lung function testing.

Although Real-Time Windows Target is a good tool to run control algorithms over a higher priority than just using a typical m-file algorithm, this tool has two drawbacks: as this is still a tool running over Windows, complex algorithms could cause that it cannot ensure a fast sampling time, because it depends on the PC characteristics and its performance. Secondly, although points within a buffer are contiguous, the time required to transfer data back to the Simulink software forces an intermission for data collection until the entire buffer has been transferred and may result in lost sample points between data buffers.

7. References

Dixon W., Dawson D., Costic B., de Queiroz M., "A MATLAB-based Control Systems Laboratory Experience for Undergraduate Students: toward Standardization and Shared Resources", *IEEE Transactions on Education*, Vol. 45, No. 3, 2002

Gambier A., "Real-time Control Systems: a Tutorial", Automation Laboratory, B6 23-29, EG. Bauteil C, University of Mannheim, 68131 Mannheim, Germany, 2005

Hernandez A. , De Keyser R., Ionescu C., "Application of a novel PID Autotuner to a lung function testing device, in *Proc. of the Int. Conf. on Biomedical Electronics and Devices (BIODEVICES 2011)*, Rome, Italy, 55-61, 2011

Ionescu C., Robayo F., De Keyser R., Naumovic M., "The Fequency Response Analyse revisited", in *Proc. of the IEEE 18th Mediterranean Conference on Control and Automation*, Marrakesh, Marocco, 1441-1446, 2010

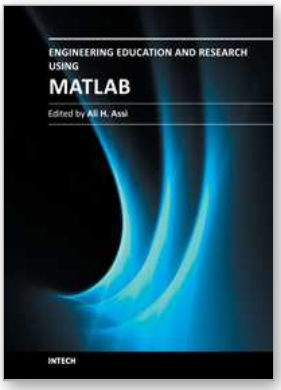
Northrop R., "Non-invasive measurements and devices for diagnosis", CRC Press, 2002

Oostveen E., Macleod D., Lorino H., Farré R., Hantos Z., Desager K., Marchal F., "The forced oscillation technique in clinical practice: methodology, recommendations and future developments", *European Respiratory Journal,* 22: 1026-1041, 2003

Pellegrino R., Viegi G., Brusasco V., Crapo R., Burgos F., Casaburi R., Coates A., van der Grinten C.P.M., Gustafsson P., Hankinson J., Jensen R., Johnson D.C., McKay R., Miller M.R., Navajas D., Pedersen O.F., Wanger J., "Interpretative Strategies for Lung Function Tests". *European Respiratory Journal,* 26: 948-968, 2005

"Real-Time Windows Target User's Guide", The MathWorks Inc. http://www.mathworks.com/products/rtwt/, 1999

Schoukens J., Pintelon R., "System Identification: a Frequency-domain Approach", (IEEE Press, 2001)



Engineering Education and Research Using MATLAB

Edited by Dr. Ali Assi

ISBN 978-953-307-656-0
Hard cover, 480 pages
Publisher InTech
Published online 10, October, 2011
Published in print edition October, 2011

MATLAB is a software package used primarily in the field of engineering for signal processing, numerical data analysis, modeling, programming, simulation, and computer graphic visualization. In the last few years, it has become widely accepted as an efficient tool, and, therefore, its use has significantly increased in scientific communities and academic institutions. This book consists of 20 chapters presenting research works using MATLAB tools. Chapters include techniques for programming and developing Graphical User Interfaces (GUIs), dynamic systems, electric machines, signal and image processing, power electronics, mixed signal circuits, genetic programming, digital watermarking, control systems, time-series regression modeling, and artificial neural networks.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Andres Hernandez, Adrian Chavarro and Robin De Keyser (2011). A Matlab® Approach for Implementing Control Algorithms in Real-Time: RTWT, Engineering Education and Research Using MATLAB, Dr. Ali Assi (Ed.), ISBN: 978-953-307-656-0, InTech, Available from: http://www.intechopen.com/books/engineering-education-and-research-using-matlab/a-matlab-approach-for-implementing-control-algorithms-in-real-time-rtwt

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821