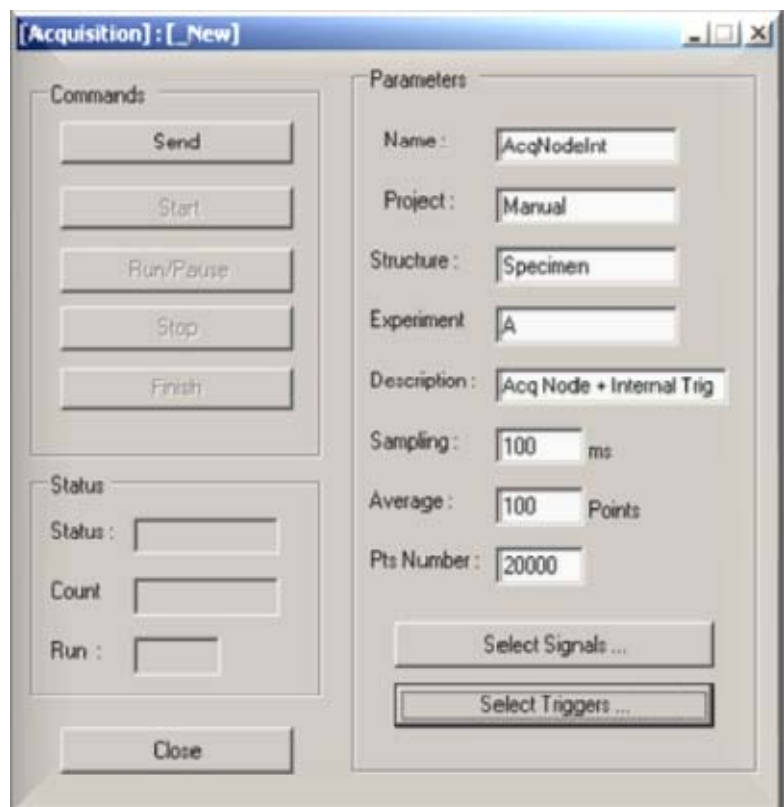




ELASPSD Data Acquisition and Signal Generator User Manual

- Beatriz Zapico Blanco, F. Javier Molina



EUR 23436 EN - 2008

The Institute for the Protection and Security of the Citizen provides research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute maintains and develops its expertise and networks in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information

Address: ELSA Laboratory, IPSC, Joint Research Centre, via Enrico Fermi 2749, 21027
Ispra, Italy
E-mail: beatriz.zapico@jrc.it
Tel.: 0332785712
Fax: 0332785379

<http://ipsc.jrc.ec.europa.eu/>
<http://www.jrc.ec.europa.eu/>

Legal Notice

Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of this publication.

***Europe Direct is a service to help you find answers
to your questions about the European Union***

**Freephone number (*):
00 800 6 7 8 9 10 11**

(*) Certain mobile telephone operators do not allow access to 00 800 numbers or these calls may be billed.

A great deal of additional information on the European Union is available on the Internet. It can be accessed through the Europa server <http://europa.eu/>

JRC 45985

EUR 23436 EN
ISBN 978-92-79-09511-5
ISSN 1018-5593
DOI 10.2788/84013

Luxembourg: Office for Official Publications of the European Communities

© European Communities, 2008

Reproduction is authorised provided the source is acknowledged

Printed in Italy

Introduction

The European Laboratory for Structural Assessment (ELSA) belongs to the Joint Research Centre (JRC) of the European Commission. It is one of the units of the Institute for the Protection and Security of the Citizen (IPSC) at the Ispra Site of the JRC. The main facility of ELSA is a large Reaction Wall-Strong Floor system equipped with powerful servo-actuators used to simulate the response of civil full-size structures submitted to dynamic loads using the ELSAPSD testing system.

During a test, the user may want some data to be displayed, analyzed or/and stored in the computer. This can be made through an acquisition object. This manual explains how to use an acquisition starting from the most simple case, which is using an acquisition node with an internal trigger. Each chapter adds new information to the previous one, so it is interesting to read the manual in a sequential order.

This manual refers to the acquisition software of ELSAPSD according to the versions PSDCYC03.DLL and M13.004 of the master controller software and `acqui.exe`???????

A. Acquisition through an acquisition node with an internal trigger

As a difference with respect to a master controller, an acquisition node is a computer exclusively dedicated to acquire signals. The user may choose whether to make the acquisition using an internal trigger, or through an external one. The procedure in the former case will be explained in this chapter with an example: the acquisition of an analog displacement.

A.1 WIRING

The displacement the user wants to acquire is a physical phenomenon. A displacement transducer is used to convert it into the corresponding measurable analog signal. As the user may want to acquire more than one variable, a conditioning box is used to put together up to 16 signals, coming from 16 different transducers. Then, the 16-wires cable coming from the conditioning box must be connected to the acquisition node. The node admits up to 4 of those cables, making a total of 64 signals per acquisition node. Once the signals have reached the node, they are converted from analog to digital by using an A/D converter. At this point the variables, called `DEV_IN.CHANNEL.NN`, where `NN` is the channel number, are ready to be acquired.

As an internal trigger is being used, no other wiring is required, apart from the LAN connection. As it will be seen afterwards, the node must be on the same network as the work station, where the Remote Control program will be run.

A.2 REMOTE CONTROL

Once the wiring has been done, the acquisition can be launched from the Remote Control program that has to be installed on a PC connected to the same network as the acquisition node. After clicking on the **New Connection** button, the user must provide the I.P. number of the acquisition node that is being used and click **Connect**.

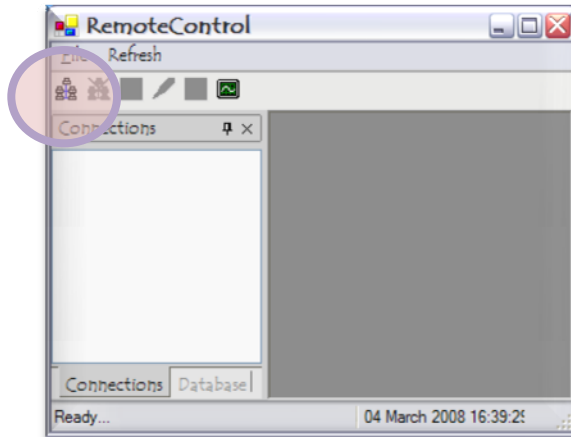


Figure 1

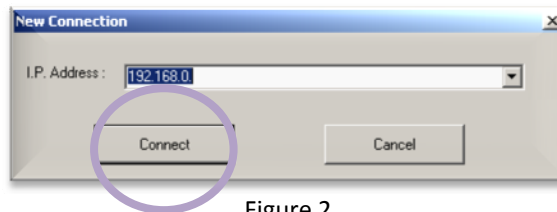


Figure 2

Then a tree menu will appear. The user must open it and click with the right button of the mouse on **Acquisitions**:

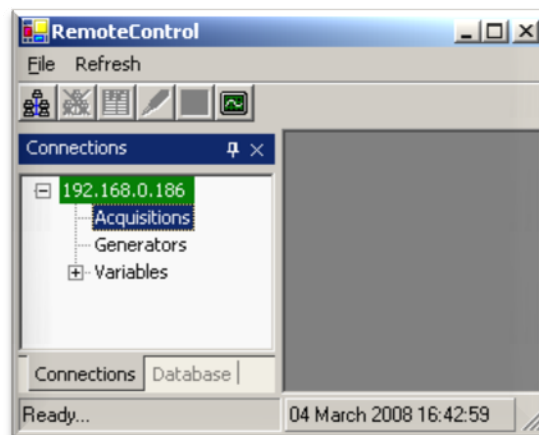


Figure 3

then select **New Acquisition**:

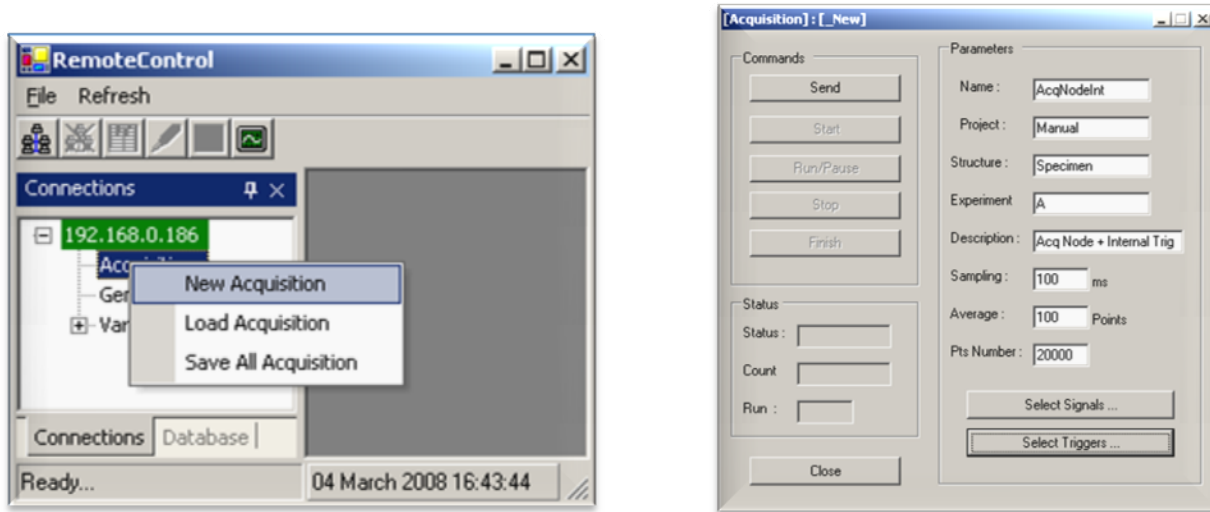


Figure 4

A dialog box will appear that must be filled in as follows (see Figure 4 above):

Name: this is the name of the *acquisition object* that will be created. It is convenient to include a reference to the acquisition node that is being used within this name, specially when several acquisitions are done at the same time on different nodes. Several acquisitions may exist in the same node if they have different names.

Project and Structure: the user must fill these areas with the proper definition of the project under development and the tested structure.

Experiment: the name of the experiment is very important because the *output data files* (see A.4 OUTPUT FILES) will be saved with the same name of the experiment. Several acquisitions may be run during the same experiment. It is convenient to choose a name that includes the experiment number and the kind of acquisition in order to easily identify them.

Description: this a free area where the user can describe the acquisition.

Sampling: the value of every channel is read and can be acquired by the node at every *internal clock* pulse. The user can choose a sampling time, multiple of the clock period, which is the interval used to check the value of the trigger and decide whether to store the data or not. The internal clock has a period of 1ms that is also the minimum sampling time value (see Figure 5).

Average: as it has been aforementioned, the value of the variables is read once every internal clock pulse. It is also possible to calculate a mean of the last n readings at every clock pulse by selecting an *Average Points Number* different from one. In this way, once every sampling time (and if the run trigger is active as it will be seen later) this mean value will be acquired. By selecting an average number equal to one, the user will avoid this calculation and the acquisition will store the instantaneous value of the variables (see Figure 5).

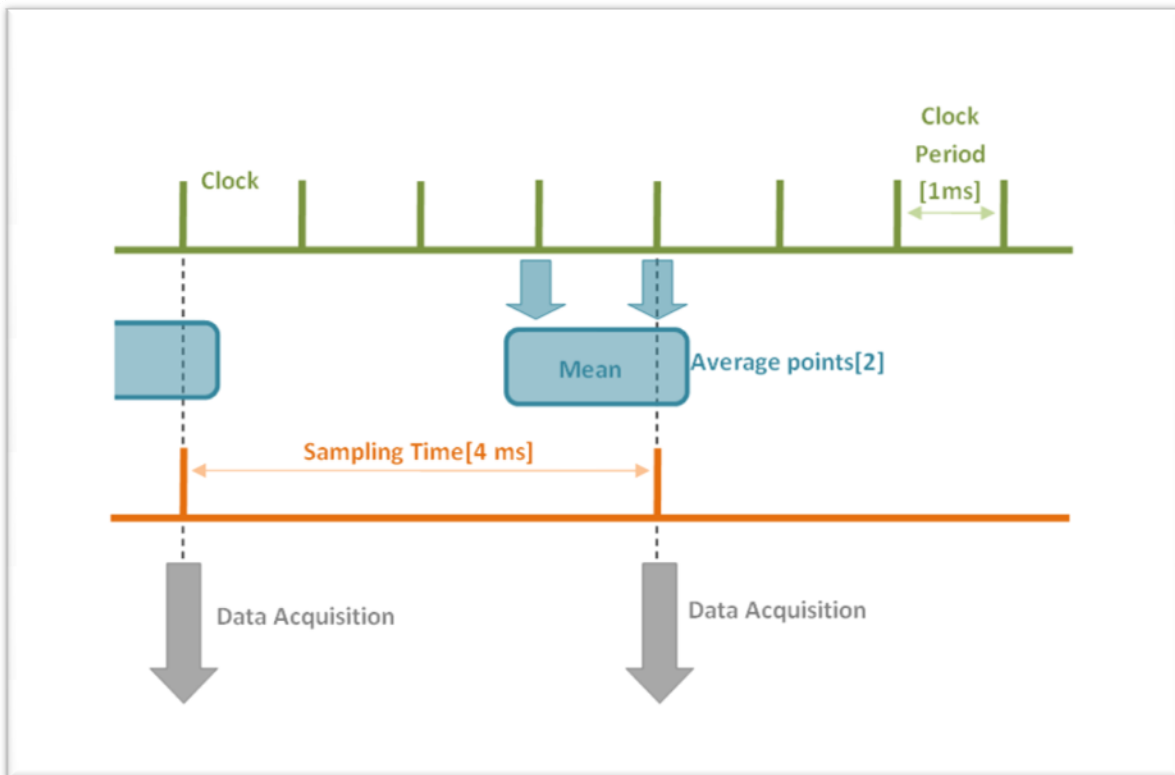


Figure 5 Calculation of the Average value during the acquisition

Pts Number: specifies the maximum number of samplings to be stored. When the number of acquired samples reaches this value, the process will stop.

At this point the user must select the signals he or she is interested in. For doing it, the user must click on the **Select Signal** button. The user will find a dialog box like the one on Figure 18. Every channel (`DEV_IN.CHANNEL_NN`) is a different signal coming from the conditioning boxes.

Now the user must define the parameters of the triggers used for the acquisition by clicking the **Select Triggers** button. There are four different triggers:

① **Start Trigger:** it switches the state of the acquisition from *ready* to *running*. That means that the system will stand still waiting for the run trigger to acquire the data.

② **Run Trigger:** after the acquisition *starts*, this trigger enables the signal acquisition. Once every n ms, where n is the sampling time selected by the user, and once the start trigger has been switched on, the acquisition object will check the run trigger. If it is on (see Figure 6 below) the value of the variables will be stored creating one point of the acquisition.

③ **Stop Trigger:** it switches the state of the acquisition from *running* to *ready* and closes the acquisition file in the hard disk. This finishes a *Run* of the acquisition. Thus, the acquisition will be waiting for the change of state of the start trigger to make a second run with a new file name (see A.4 OUTPUT FILES).

④ **Finish trigger:** when this trigger is activated the acquisition object is deleted by the system and no more runs can be made with it. It is not possible to open another acquisition with the same name of the finished one for the current run of the master application either.

There are three fields that must be filled for every trigger:

① **Channel:** By selecting the channel, the user is choosing the trigger signal that is going to be used for that specific trigger. The system contains up to ten predefined internal trigger signals. When using one of those predefined triggers (*Trigger1*, *Trigger2*, etc.), the remote application connects directly to these triggers the action of the corresponding button in the acquisition interface dialogue box. The user can activate every trigger (always following the order in which they have been explained) by clicking on the proper button. In this case the trigger signal is negative and uniform until the trigger button is pressed. At this moment it immediately changes to positive and remains there. The predefined trigger channel is commonly used for *Start*, *Stop* and *Finish* trigger, and, in this case, also for the *Run* trigger. Other kinds of *Run* trigger signals will be explained in the next sections.

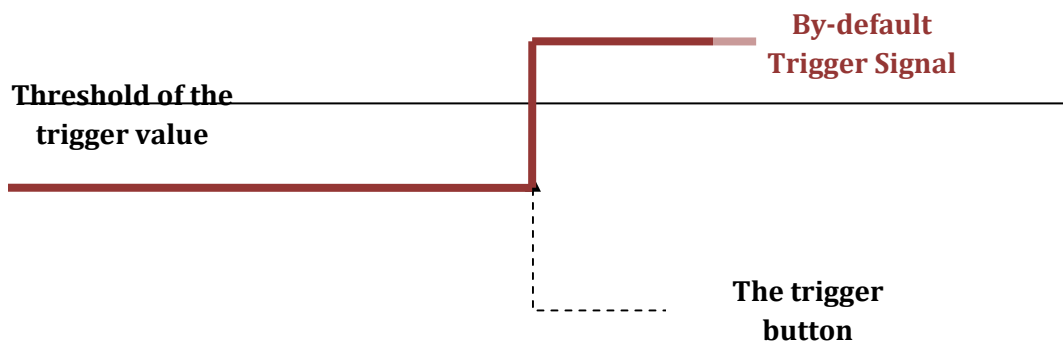


Figure 6 Predefined Run Trigger

② **Type:** this is the method used to evaluate the trigger signal. There are four different types of trigger: rising edge, rising level, falling edge and falling level.

⚠ **Note:** The acquisition node reads the trigger signal only once every clock pulse, (see Figure 7).

- **Rising edge:** a rising edge type trigger activates the acquisition of one sampling when the input signal passes the threshold while rising (changing from $\text{signal} < \text{threshold}$ to $\text{signal} > \text{threshold}$). This kind of trigger allows the acquisition of just one sample each time this condition is verified.
- **Rising Level:** the trigger keeps the acquisition active for all the instants when the input signal is over the threshold value. This enables the acquisition of several samplings while the trigger signal maintains over the threshold, commonly used for continuous acquisitions.
- **Falling edge:** a falling edge type trigger activates the acquisition of one sampling when the input signal passes the threshold while falling (changing from

signal > threshold to signal < threshold). This kind of trigger allows the acquisition of just one sample each time this condition is verified.

→ *Falling Level*: the trigger keeps the acquisition active for all the instants when the input signal is under the threshold value. This enables the acquisition of several samplings while the trigger signal maintains under the threshold.

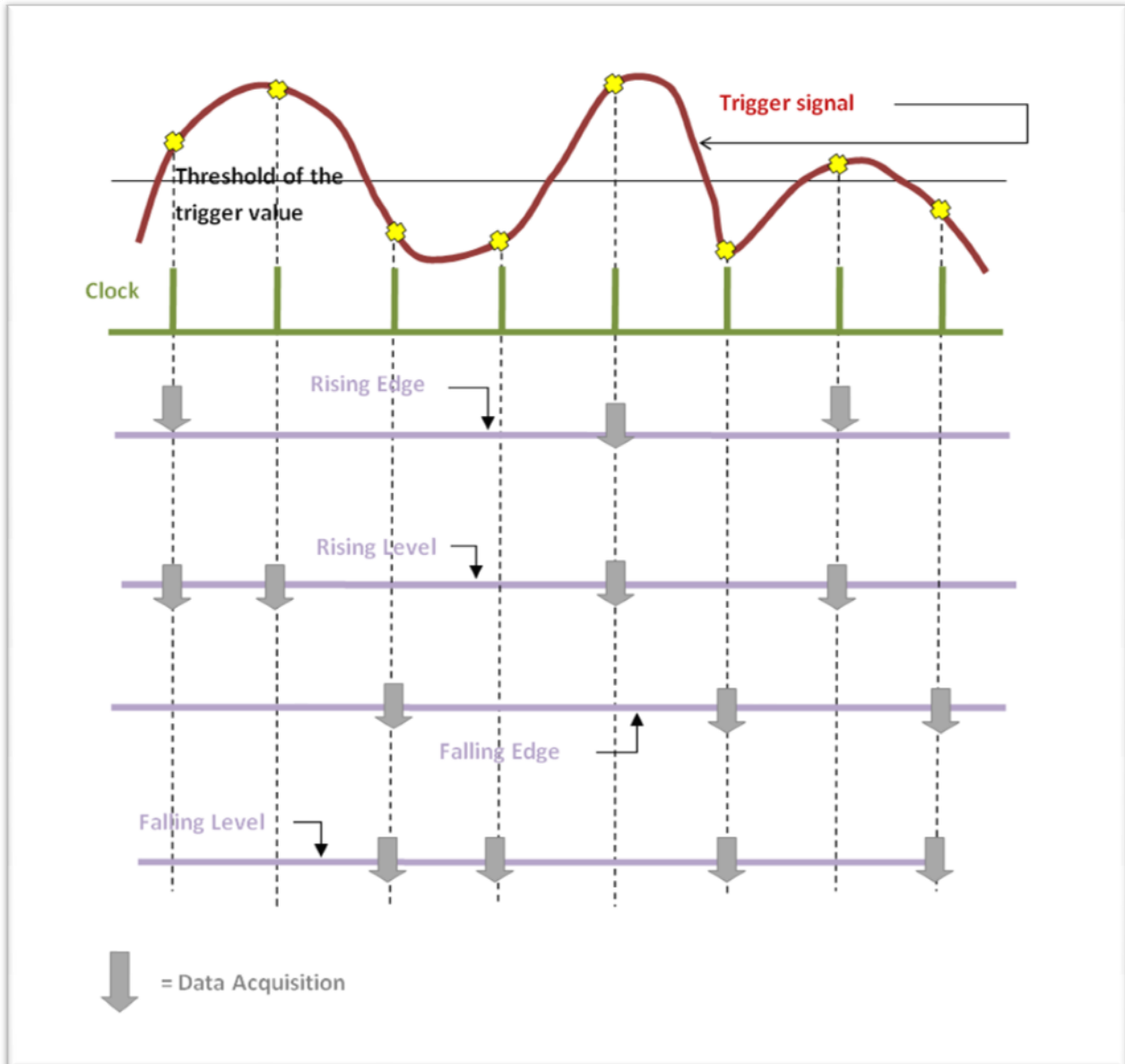


Figure 7 Acquisition activation depending on the trigger signal and the trigger type

The right type for start, stop and finish trigger is *Rising Edge*, because they must introduce a single action when the user presses the button, giving way to starting, stopping or finishing the acquisition process. In the case that is being explained in this section, the run trigger type must be *Rising Level* instead, because it has to introduce a continuous action during a time interval. During this time interval all the samplings will be saved.

③ *Value*: is the numerical value used to evaluate the trigger state, typically one.

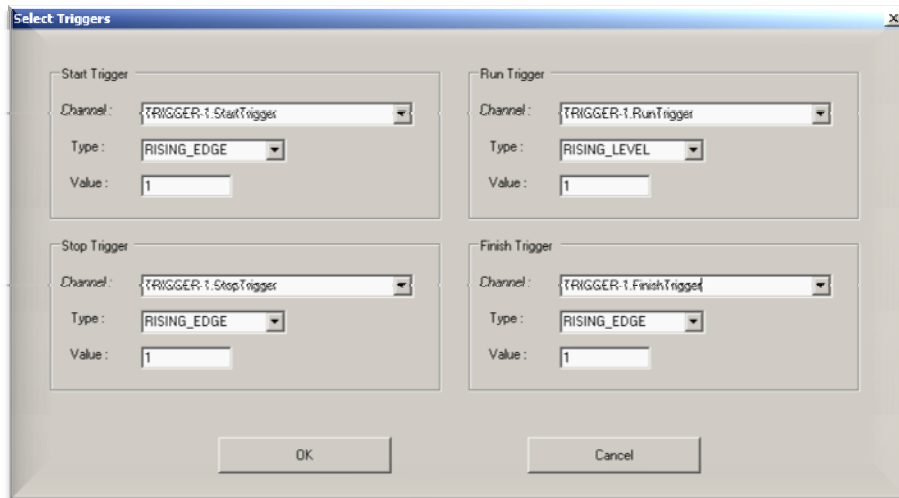


Figure 8

Once every field has been filled out, the user must click on the **Send** button, so that the acquisition object is created. No parameter of the acquisition can be changed from now on. The name of the sent acquisition setting will appear replacing the word **_New** that can be seen with a blue background on Figure 9. The acquisition setting can now be saved in a file. The user has to click with the right button of the mouse on the tree menu: [Connections](#) / [Acquisitions](#) / [New](#) / [Save](#) and then select the folder to save the configuration in as a text file.

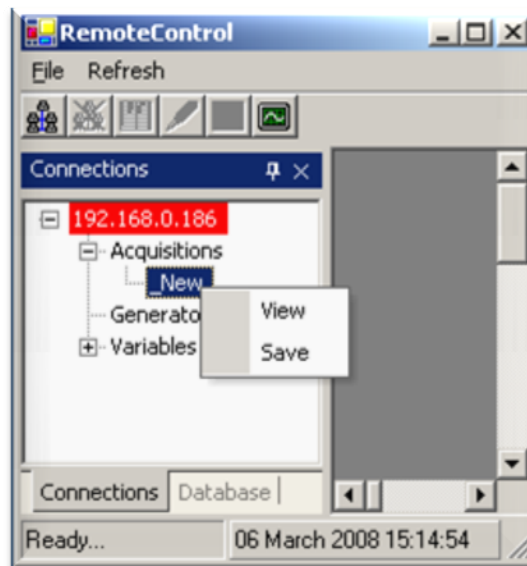


Figure 9

Once an acquisition setting has been saved, the user can reload it in future experiments. An example of acquisition configuration file could be the following:

```

#Start of Acquisition Setup
File : \\C\j01acq181.Acq

#Name of the acquisition Object
Name=ACQ181

#Message queue size
dwQueueSize=8000

#Memory size
dwMemorySize=200480

#Maximum block
dwBlockNumber=50000

#Acquisition Type
usAcquisitionType=1

#project name
szProjectName=Manual

#structure name
szStructureName=None

#experiment name
szExperimentName=j01std

#acquisition name
szAcquisitionName=ACQ181

#description
szDescription=example

#sampling time
dwSamplingTime=100

#Average
dwAverage=100

#buffer size (number of acq)
dwBufferSize=10000

#signalnumber
dwSignalNumber=64

szSignal2=DEV_IN.CHANNEL_2
szSignal3=DEV_IN.CHANNEL_3
.
.
szSignal64=DEV_IN.CHANNEL_64

```

```

# RISING_EDGE           = 1
# RISING_LEVEL         = 2
# FALLING_EDGE         = 3
# FALLING_LEVEL        = 4

#StartTrigger
szStartTrigger=TRIGGER-1.StartTrigger

#StartTriggerType
usStartTriggerType=1

#StartTriggerValue
dStartTriggerValue=1

#AcqTrigger
szAcqTrigger=TRIGGER-1.RunTrigger

#AcqTriggerType
usAcqTriggerType=2

#AcqTriggerValue
dAcqTriggerValue=1

#StopTrigger
szStopTrigger=TRIGGER-1.StopTrigger

#StopTriggerType
usStopTriggerType=1

#StopTriggerValue
dStopTriggerValue=1

#FinishTrigger
szFinishTrigger=TRIGGER-1.FinishTrigger

#StopTriggerType
usFinishTriggerType=1

#FinishTriggerValue
dFinishTriggerValue=1

#End of Acquisition Setup File
:
\\C\Manual\Experiments\j01\Config_Acq\j01acq181.Acq

```

Figure 10 Acquisition Configuration Text File

A.3 STATES OF THE ACQUISITION OBJECT

Once the acquisition has been sent, the acquisition object has been created and its status has changed from *Init* to *Ready*. By clicking on the *Start* button, the user can change the object state from *Ready* to *Running*, but it will generate points only after pressing the *Run* button, once every sampling time. When the previously selected *Number of Points* has been reached, or the *Stop* button is pressed, the state changes from *Running* to *Ready* and the acquisition file is closed. At this point the user can choose whether to run a new acquisition (the run number will be increased by one, as well as the output file extension) by clicking again the *Run* button, or to terminate the object by clicking the *Finish* button. When the *Finish* trigger is activated, the state of the acquisition passes to *Finished* and the acquisition object is deleted by the system.

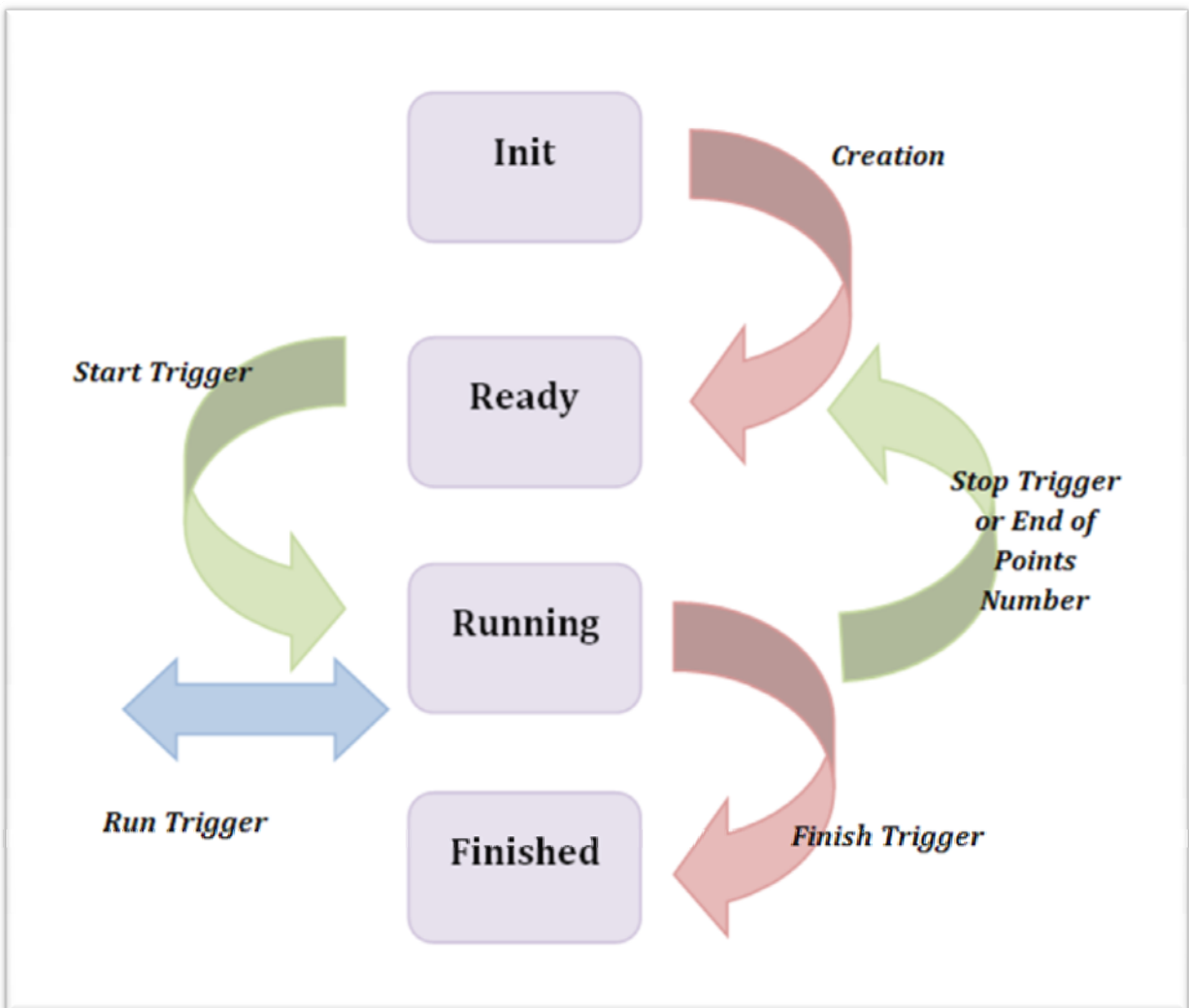


Figure 11 States of the Acquisition Object

A.4 OUTPUT FILES

The results of every acquisition run are two files. The first one is a header, which is a text file containing the description of the acquisition: parameters employed, channels, etc. The name of the header file is composed by the experiment name, specified when the acquisition object was created, and the extension *Hnn*, where *nn* is the run number (i.e. J01STD.D01).

```
[HEADER]
Project Name : Manual
Experiment Name : j01std
Run Number : 1
Description : example
Sampling Time : 100
Average : 100
Start Time : 04/03/2008
11:24:02
[CHANNELS]
1:TIME
2:DEV_IN.CHANNEL_1
3:DEV_IN.CHANNEL_2
.
.
.
65:DEV_IN.CHANNEL_64
```

The second file created contains the data in binary format real single (*float*). Its name will be the name of the experiment followed by *Dnn*, where *nn* is the run number (i.e. J01STD.D01).

To process a data file the user can employ the following lines of matlab:

```
Nsig=65;
fid=fopen('J01STD1.D01','r');
if fid==-1
    display('The file was not found')
end
adata=fread(fid,inf,'float');
fclose(fid);
npoints=length(adata)/Nsig;
Signal_values=reshape(adata,Nsig,npoints);
```

Note that *Nsig* must be equal to the number of acquired signals in the file, plus the time in first position, as reflected in the header file. The matrix *Signal_values* will then contain as many rows as time instants and one column for every signal.

B. Acquisition through an acquisition node with an external trigger (synchronous acquisition with a master DLL)

The user may want the run trigger to be different from the simple on/off curve provided by any of the ten predefined triggers that have been explained in the previous section, or may want the acquisition samples to be synchronized with another device. An external trigger is now needed. It can be provided by a signal generator, for example, or by a Master Controller with a DLL running on it. The procedure for implementing an acquisition in this case is explained in this section. Based in the former chapter, only the new requirements and possibilities will be described.

Wiring: In addition to the former wiring, the external trigger must be connected to the trigger-in port of the acquisition node via a BNC cable. The trigger signal, thus, will be treated by the acquisition node as any other input channel, been read at every pulse of the internal clock.

When using a master controller channels from 17 to 32 of the master are output channels. They must be connected to a digital output box through a flat cable. In this way they get to the box and can be distributed from it. The DLL trigger signal is at the 17th channel, so port 17th of the digital output console must be connected to the trigger-in port of the acquisition node, which is the device-in channel number 65. For a better synchronism the clock-in port of the acquisition node has to be connected to the clock-out port of the master too. Some changes must be done on the set up file also:

Sampling: when using a signal generator for the trigger signal the acquisition node keeps its own internal clock. When the trigger signal is coming from the master instead, the clock is external: the master internal one is used. This new clock have a period of 2ms, so now the channels will be read every 2ms, and the sampling time selected by the user must be equal to this period (2ms).

Trigger Channel: when selecting the run trigger signals, the user has to choose: `Channel=DEV_IN.CHANNEL_65` for the wiring that is been explained before. The trigger channel may change depending on the acquisition node hardware.

Trigger Type: now the acquisition is no longer triggered by pressing a button, but by the pass through the threshold of a dedicated signal. Thus, the trigger type the user have to select is Rising Edge.

Once the start button has already been pressed, the acquisition node will read the value of the variables every clock pulse. Every sampling time (a multiple of the clock period), if the external trigger signal has passed through the threshold while rising, those values are acquired.

⚠ *Note: In order to make synchronous acquisitions with several acquisition nodes working without a master, all the nodes should use the clock signal coming from one of them and the definition of the run trigger type should be RISING_LEVEL. A common signal coming from a signal generator should then be used for all the nodes.*

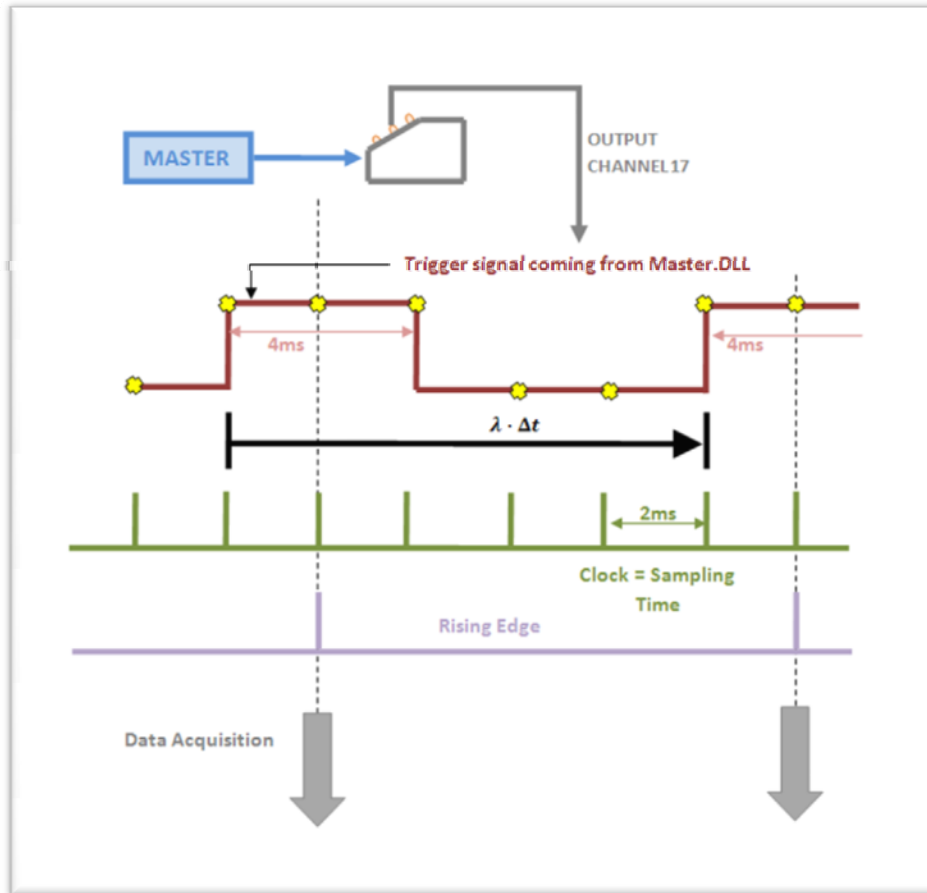


Figure 12 Acquisition through an acquisition node with an external trigger coming from a master DLL

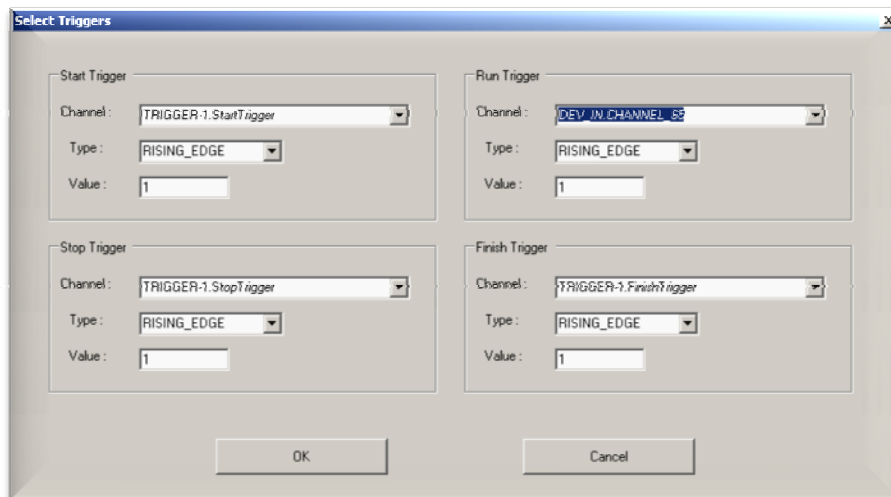



Figure 13

C. Acquisition through a Master Controller with an internal predefined trigger

When using a *Master Controller* for conducting an experiment, the acquisition can be made on the master itself, without using an acquisition node. The stored data are, thus, the values of the variables used by the application that is being run in the master during the test.

Wiring: The variables used by the application are physical phenomena transformed into digital or analog signals. The analog ones must be converted into digital using an A/D converter. Those signals arrive to the slave controller via the piston cables or BNC cables at the slave connection box. For example, channels AD9 to AD16 of every slave can be used to acquire a signal. Every slave is connected to the master through a 16-wires cable.

Besides the measured signals coming from the slaves, on the master can be found other signals of internal variables used in the internal calculation process.

 **Note:** There are also four channels arriving from the slave that typically are not used during the test: *acc*, *speed*, *force2* and *lvdt*, corresponding to the measurement of the acceleration, speed and a secondary force, and the measurement coming from a linear variation displacement transducer respectively. In these cases the user may employ them for the acquisition of different signals for debugging purpose (i.e. measurement of the temperature or of secondary displacements) by connecting the transducer to the correspondent input port of the slave.

The trigger can be, again, internal or external. When using an internal trigger no special wiring is needed.

Signals: During a test the master exchanges with the slaves the groups of variables `internal_algo_input` and `internal_algo_output` once every internal clock pulse. These variables are typically acquired using one of the ten predefined internal triggers available in the application. This kind of acquisition is called *continuous* and it saves every desired sampling from the moment the run button is pressed until the user presses the stop button. This acquisition will not be synchronous with the DLL records.

Name: It is convenient to include a reference to the master that is being used within the name of the acquisition, specially when several acquisitions are done at the same time with different masters or acquisition nodes, and also an indication of the type of acquisition, continuous in this case (see Figure 15).

Experiment: the name of the experiment is very important because the acquisition data file will be saved with the same name of the experiment. It is convenient to choose a name that includes the experiment number and the kind of acquisition that is going to be done (see Figure 15).

Sampling: The internal clock of the master has a period of 2ms that is also the minimum sampling time value (see Figure 15).

Trigger: As it has been said before, the trigger will be selected among one of the ten internal ones. The type must be *Rising Edge* for the *Start*, *Stop* and *Finish* triggers, and *Rising Level* for the *Run* trigger. In this way the data will be acquired from the moment the trigger signal pass through the threshold, that is when the user press the *Run* button, and untill the acquisition is stoped.

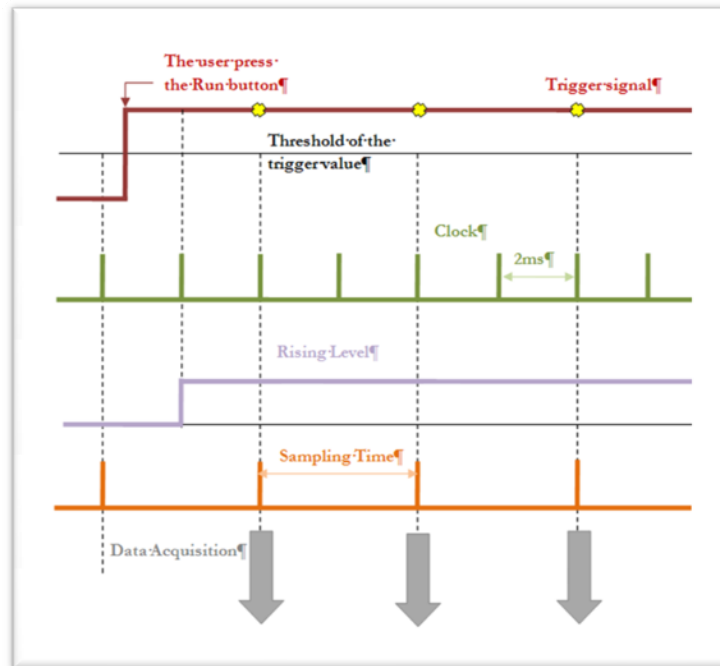


Figure 14 Acquisition through a Master Controller with an internal trigger

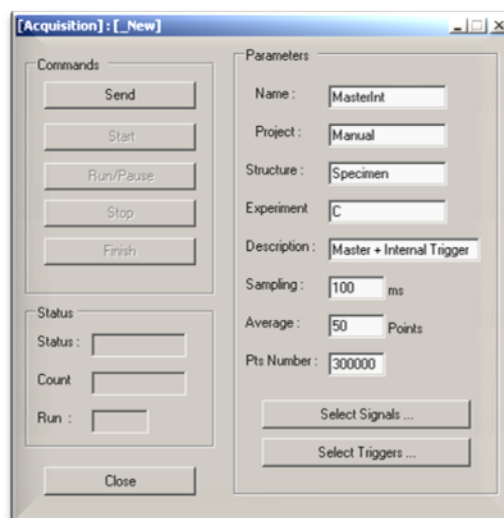


Figure 15

D. Acquisition through a Master Controller with an internal trigger synchronous with the DLL

When a *DLL* is run on the master, there are other variables that can be saved, and also an alternative way of triggering the acquisition:

Signals: There are different groups of variables created by the *DLL* run on the master that can be acquired:

① **ALGOR_T:** where the instantaneous value of the variables is stored. It is refreshed every 2ms. Typically used for debugging purpose.

② **ALGORAV:** where the accumulative value of the variables is stored and divided by the number of points at the end of every input record, so as to obtain a mean value of the variable. It is refreshed at the end of every input record. The average acquisition, which is a *synchronous* one, is usually the most interesting one (see Figure 16, Figure 17 and Figure 18).

③ **ALGORESTART:** where the variables that are useful for the restart of the test are stored. It is refreshed at the beginning of every input record but the values of the variables are not averaged. The restart acquisition, as the average one, is *synchronous*.

Average: When doing an instantaneous acquisition, as it has been explained before, the user can select the number of points used for the calculation of the average. When running an average or a restart acquisition instead, the acquired data are already mean values of the selected variables and thus, there is no need of former average calculation (Average points=1). For other reasons, also for the rest of the variables, the average should be avoided.

Trigger: When a *DLL* is run on the master there is the possibility of using a trigger signal generated by the *DLL*. This trigger signal is greater than the threshold for two sampling periods (4ms) at the end of every input record. The *Run Trigger Type* must be *Rising Edge* in order to do the acquisition of just one instant per input record. When selecting the *Run Trigger Channel* the user must choose `DEV_OUT.CHANNEL_17`, which is the channel containing the signal modified by the *DLL*. As it has been seen before, this channel is also used as an output channel when the acquisition is done outside the master.

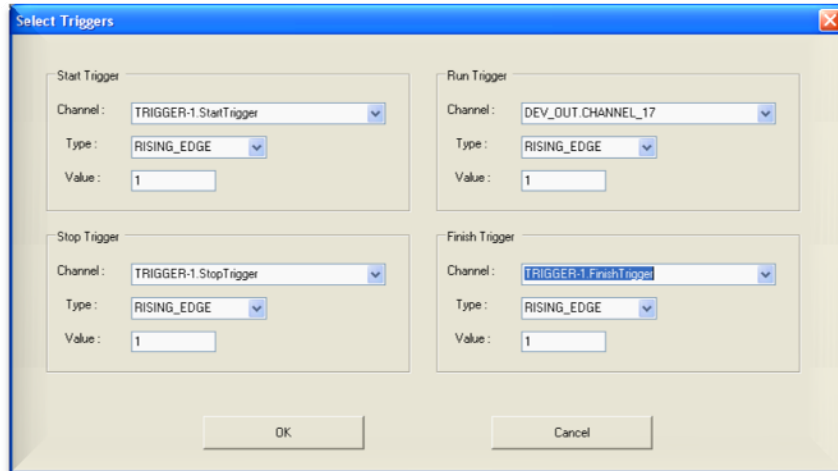


Figure 16

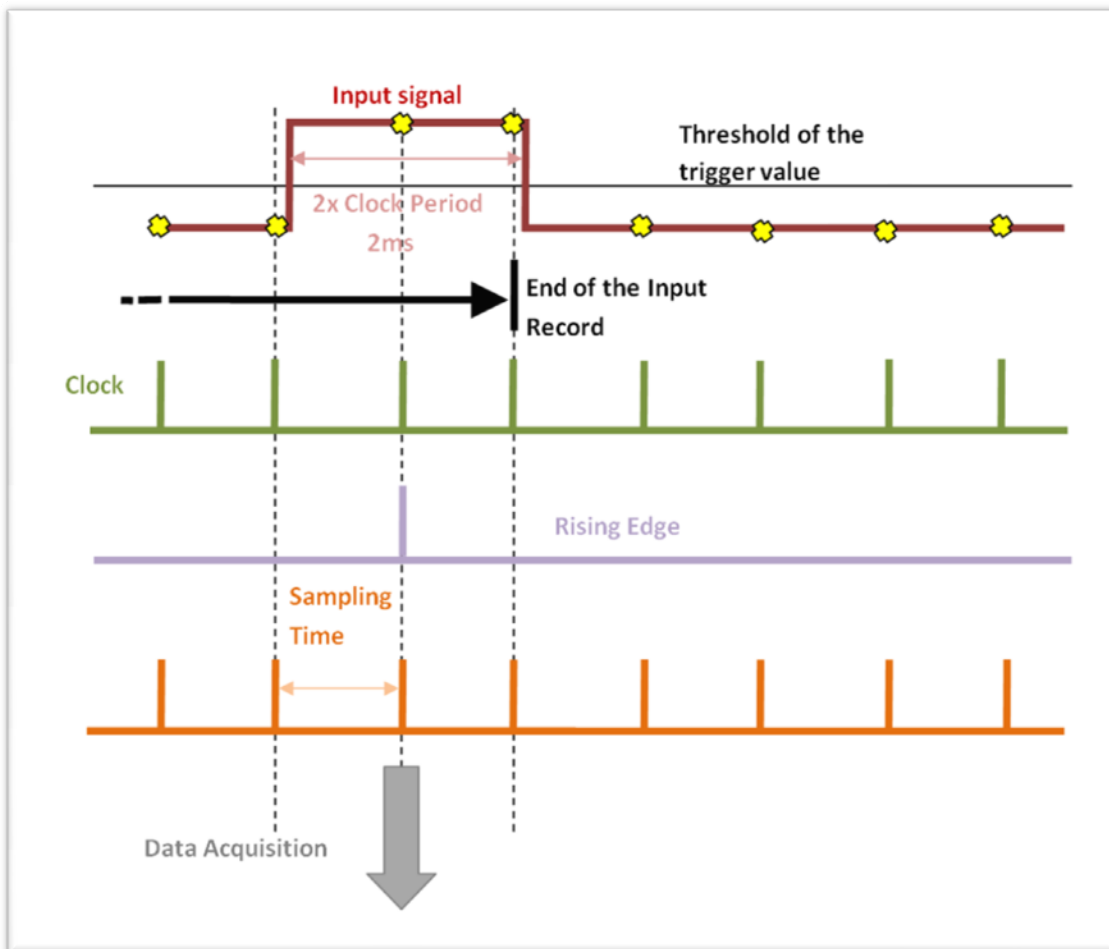


Figure 17 Acquisition through a Master Controller with an internal trigger synchronous with the DLL

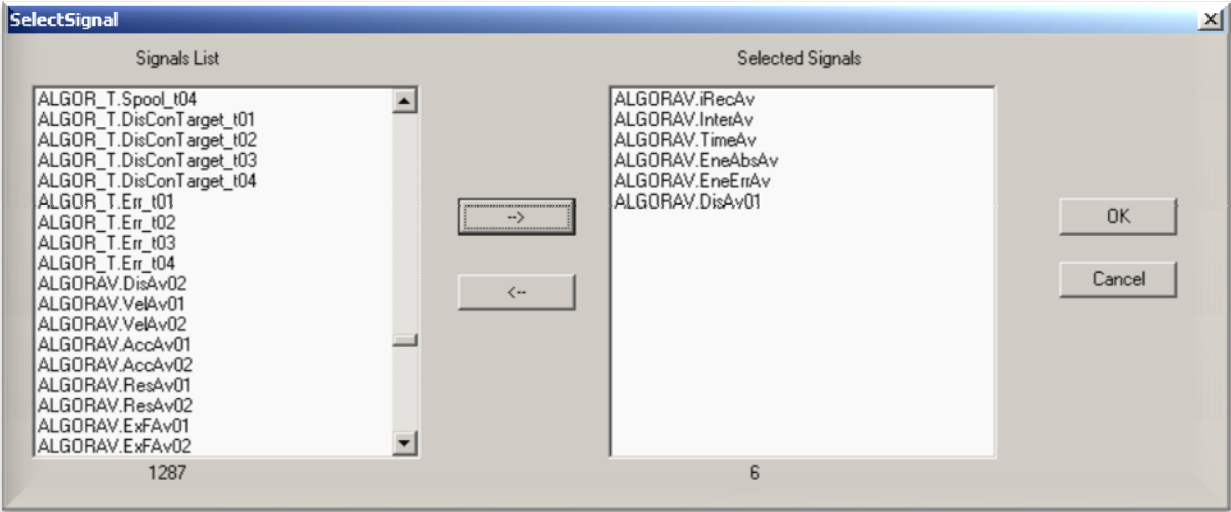


Figure 18

	Acquisition Node		Master Controller	
	Internal Trigger	External Trigger	Internal Trigger	Int. Trigger + DLL
Name	AcqNodeInt	AcqNodeExt	MasterInt	MasterDLLInt
Project	Manual	Manual	Manual	Manual
Structure	Specimen	Specimen	Specimen	Specimen
Experiment	A	B	C	D
Description	Acq Node + Internal Trigger	Acq Node + External Trigger	Master + Internal Trigger	Master + DLL + Internal Trigger
Sampling [ms]	100	2	100	2
Average [points]	100	100	50	1
Pts Number	20 000	10 000	300 000	300 000
Signals	DEV_IN.CHANNEL_N	DEV_IN.CHANNEL_N	INTERNALALGOIN PUTN INTERNALALGOO UTUTN	ALGORAV
Run Trigger Channel	TRIGGER_N.RunTrigger	DEV_IN.CHANNEL_65	TRIGGER_N.RunTrigger	DEV_OUT.CHANNEL_17
Run Trigger Type	Rising Level	Rising Edge	Rising Level	Rising Edge

Table 1 Summary

E. Generator

A cyclic test may be carried out using a DLL algorithm (see PSDCYC03 Manual/Master Configuration). In this case an input pattern file must be provided to the system. When running a very simple test, the user is not familiar with the DLL algorithm or a large amount of points are needed, the test can be carried out using a **Generator**.

The generation object, as the acquisition object, is launched from the remote control program. The signal can be directly generated by the application or read from a file. This file can be created using matlab, using double format values, and must be placed in the folder [C:\master](#). (See the example below).

```
arData = sin(1:1000);  
fid = fopen('Signal.Gen','w');  
fwrite(fid,arData,'double');  
fclose(fid);
```

Figure 19: Matlab generation of a signal vector

The application has six predefined signals: square wave, sinus wave, triangle wave, DC wave. Randon wave and manual ramp.

We are not describing here the acquisition status or the trigger signals as the user can find the information in the previous chapters (A. Acquisition through an acquisition node with an internal trigger).

The selection of the signals, instead, must be done in a different way. The generated signal becomes a variable that must be selected by the user. Typically this variable is the [INTERNALALGOUTPUTn.Reference](#), where *n* is the number of the corrispondent controller.

When runing an acquisition, its run trigger must be the same of the generator one (called *Gen Trigger*) so as to acquire exactly the generated points.

The steps to follow in order to create and save a generator object are the same of those explained for the acquisitions. Here are, anyway, some parameters that the user should know before filling in the the generator set up (see Figure 21):

- ① The user can choose any generator **Name** he or she likes, and the setup will be saved under this name.
- ② With the generator **Type** the user can choose the kind of signal to create, either a predefined or a external generated one (see Table 1 below).

③ The **Filename** is the name of the external signal, when used. It must be a **.dat** file stored in **C:\master**. This voice will be ignored when using any of the other generator types.

Generator type	
1	Read file signal
2	Generate square wave
3	Generate sinus wave
4	Generate triangle wave
5	Generate DC wave
6	Generate random wave
7	Generate Manual Ramp

Table 2 Generator type

④ The **Period** is the sampling used for generating the points, in milliseconds, e.g. 2ms.

⑤ The user may want the wave to describe a non-symmetric cycle. The **DutyCycle** defines the percentage of the cycle run on the positive side. A DutyCycle of 50% corresponds to a symmetric cycle (see Figure 20 below).

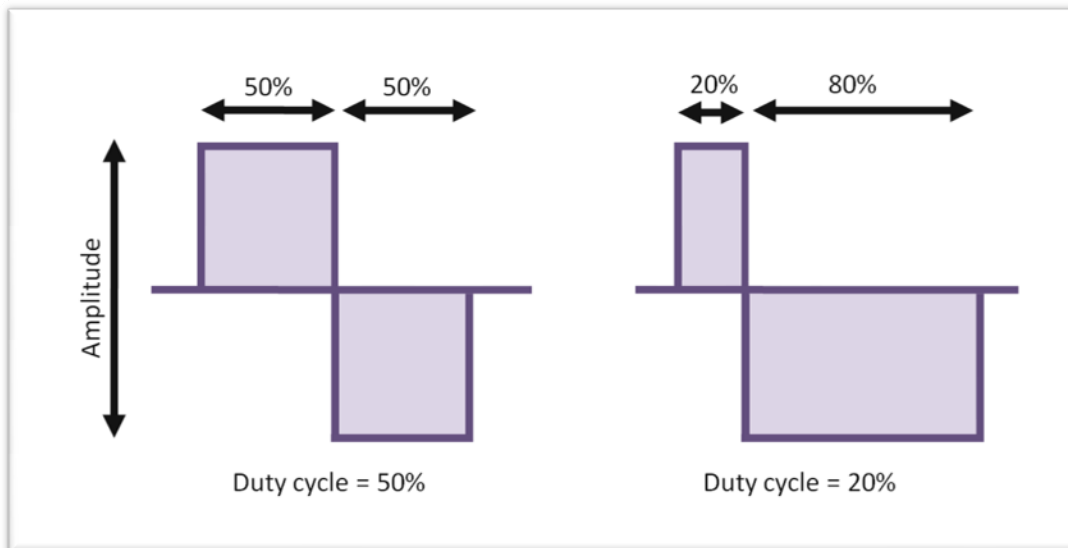


Figure 20 Wave

⑥ When using a predefined generator, the user can select the maximum **Amplitude** of the generated signal, expressed in the physical unit of the output signal (see Figure 20 below).

⑦ The user may want the generated wave to oscillate around a non-zero value. This can be done by selecting a non-zero **Offset** in the configuration.

- ⑧ The **Span** is a percentage factor applied to the generated signal (e.g. 100%).
- ⑨ The **Counter** is total number of waves the user wants to accomplish. A counter equal to zero means that the wave will be repeated until infinite.
- ⑩ **Slope Inc** is an obsolete voice and therefore not to be filled in

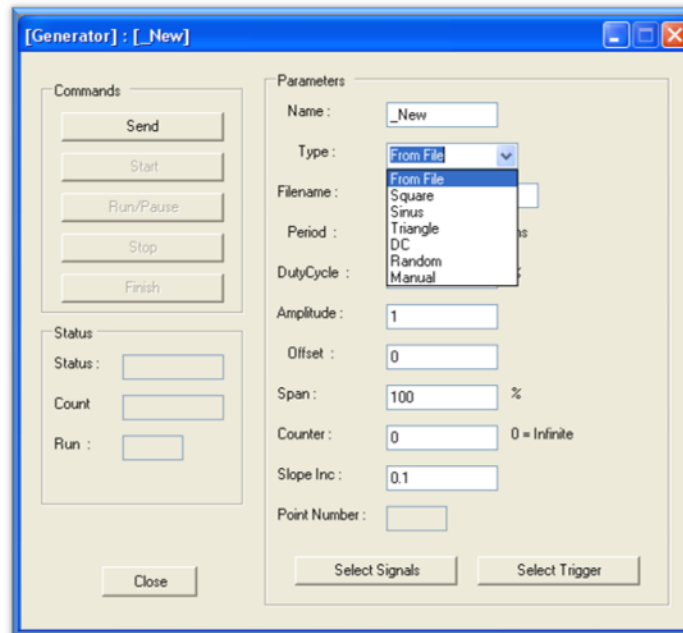


Figure 21: Generator set up window

INDEX AND REFERENCES

INDEX

Introduction.....	1
A. Acquisition through an acquisition node with an internal trigger	1
A.1 WIRING	1
A.2 REMOTE CONTROL.....	2
A.3 STATES OF THE ACQUISITION OBJECT.....	9
A.4 OUTPUT FILES	10
B. Acquisition through an acquisition node with an external trigger (synchronous acquisition with a master DLL).....	11
C. Acquisition through a Master Controller with an internal predefined trigger.....	13
D. Acquisition through a Master Controller with an internal trigger synchronous with the DLL	15
E. Generator.....	18

REFERENCES

- Elsa PSD Testing System, Philippe Buchet, ELSA Laboratory, Joint Research Centre, Ispra, Italy. (2004)
- PSDCYC03.DLL User Manual, Beatriz Zapico Blanco, F. Javier Molina, ELSA Laboratory, Joint Research Centre, Ispra, Italy. (2008)

European Commission

EUR 23436 EN– Joint Research Centre – Institute for the Protection and Security of the Citizen

Title: Acquisition User Manual

Author(s): Beatriz Zapico Blanco, F. Javier Molina

Luxembourg: Office for Official Publications of the European Communities

2008– 22 pp. – 21 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1018-5593

ISBN 978-92-79-09511-5

DOI 10.2788/84013

Abstract

During a test, the user may want some data to be displayed, analyzed or/and stored in the computer. This can be made through an acquisition. This manual explains how to do an acquisition starting from the most simple case: using an acquisition node with an internal trigger. Each chapter adds new information to the previous one: using an external trigger, a master controller, etc.

How to obtain EU publications

Our priced publications are available from EU Bookshop (<http://bookshop.europa.eu>), where you can place an order with the sales agent of your choice.

The Publications Office has a worldwide network of sales agents. You can obtain their contact details by sending a fax to (352) 29 29-42758.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

