Høgskolen i Telemark

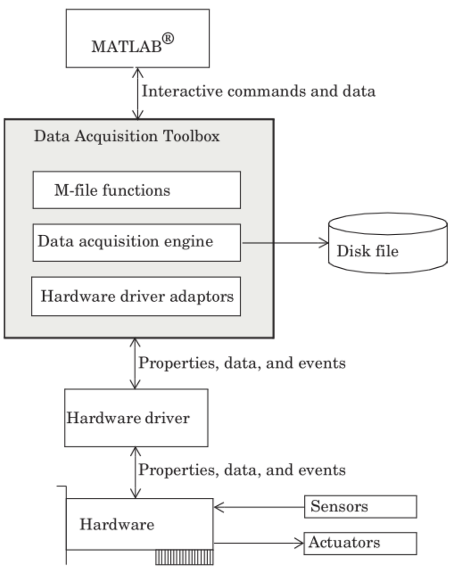
Telemark University College

Department of Electrical Engineering, Information Technology and Cybernetics

Tutorial

DAQ in MATLAB

HANS-PETTER HALVORSEN, 2011.06.07



Faculty of Technology, Postboks 203, Kjølnes ring 56, N-3901 Porsgrunn, Norway. Tel: +47 35 57 50 00 Fax: +47 35 57 54 01

Table of Contents

1 Introduction

In this Tutorial we will learn how to create DAQ (Data Acquisition) applications in MATLAB and Simulink. We will use a USB-6008 DAQ device from National Instruments as an example. In order to use DAQ devices from National Instruments in MATLAB/Simulink we need to install the NI-DAQmx driver provided by National Instruments. In addition we need the Data Acquisition Toolbox for MATLAB/Simulink.

1.1 MATLAB

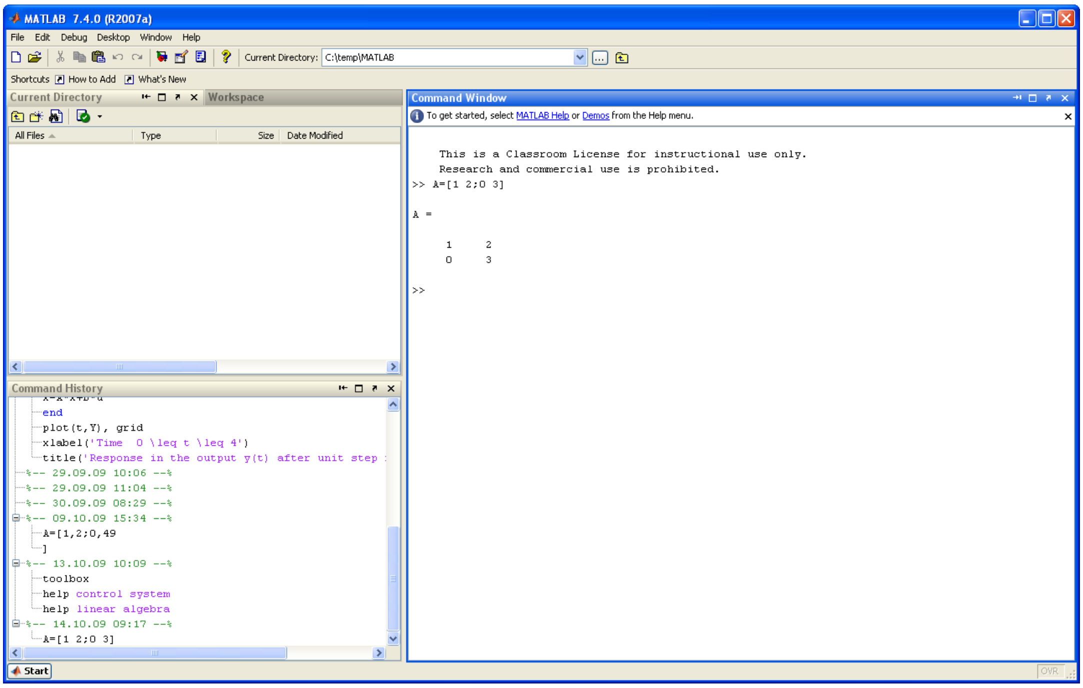
MATLAB is a tool for technical computing, computation and visualization in an integrated environment, e.g.,

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is developed by The MathWorks. MATLAB is a short-term for MATrix LABoratory. MATLAB is in use world-wide by researchers and universities.

For more information, see **www.mathworks.com**

Below we see the MATLAB Environment:



MATLAB has the following windows:

- Command Window
- Command History
- Workspace
- Current Directory

The **Command** window is the main window. Use the Command Window to enter variables and to run functions and M-files scripts (more about m-files later).

Watch the following "Getting Started with MATLAB" video:

http://www.mathworks.com/demos/matlab/getting-started-with-matlab-video-tutorial.html

1.2 Simulink

Simulink, developed by The MathWorks, is a commercial tool for modeling, simulating and analyzing dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in control theory and digital signal processing for simulation and design.

Basic information about Simulink can be found here:


Watch the video Getting Started with Simulink by The MathWorks.

Read the "**Introduction to Simulink**" document. You will need to know these topics when doing the rest of the tasks in this Lab Work.

The Tutorial is available from: **http://home.hit.no/~hansha/**.

I also recommend the following Video:

"**Simulink Quickie!**". Try to keep up and create the Simulink Block diagram while watching the video.

1.2.1 Data Acquisition Toolbox

Data Acquisition Toolbox software provides a complete set of tools for analog input, analog output, and digital I/O from a variety of PC-compatible data acquisition hardware. The toolbox lets you configure your external hardware devices, read data into MATLAB and Simulink environments for immediate analysis, and send out data.

Data Acquisition Toolbox also supports Simulink with blocks that enable you to incorporate live data or hardware configuration directly into Simulink models. You can then verify and validate your model against live, measured data as part of the system development process.

We will use the Data Acquisition Toolbox in order to write and read data to and from a USB-6008 DAQ device from National Instruments.

1.3 USB-6008 DAQ Device

NI USB-6008 is a simple and low-cost multifunction I/O device from National Instruments.

The device has the following specifications:

- 8 analog inputs (12-bit, 10 kS/s)

- 2 analog outputs (12-bit, 150 S/s)
- 12 digital I/O
- USB connection, No extra power-supply neeeded
- Compatible with LabVIEW, LabWindows/CVI, and Measurement Studio for Visual Studio .NET
- NI-DAQmx driver software

The NI USB-6008 is well suited for education purposes due to its small size and easy USB connection.

1.4 NI DAQmx driver

National Instruments provides a native .NET API for NI-DAQmx. This is available as a part of the NI-DAQmx driver and does not require Measurement Studio.

2 Data Acquisition

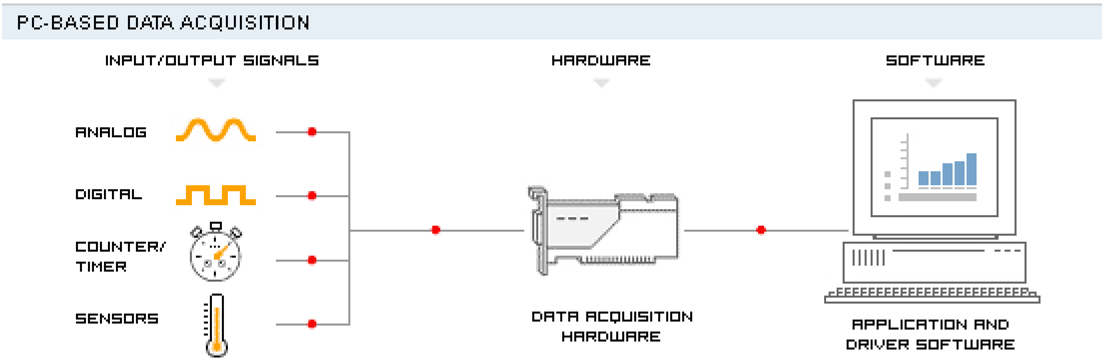
2.1 Introduction

The purpose of data acquisition is to measure an electrical or physical phenomenon such as voltage, current, temperature, pressure, or sound. PC-based data acquisition uses a combination of modular hardware, application software, and a computer to take measurements. While each data acquisition system is defined by its application requirements, every system shares a common goal of acquiring, analyzing, and presenting information. Data acquisition systems incorporate signals, sensors, actuators, signal conditioning, data acquisition devices, and application software.

So summing up, Data Acquisition is the process of:

- Acquiring signals from real-world phenomena
- Digitizing the signals
- Analyzing, presenting and saving the data

The DAQ system has the following parts involved, see Figure:



The parts are:

- Physical input/output signals
- DAQ device/hardware
- Driver software
- Your software application (Application software)

2.1.1 Physical input/output signals

A physical input/output signal is typically a voltage or current signal.

2.1.2 DAQ device/hardware

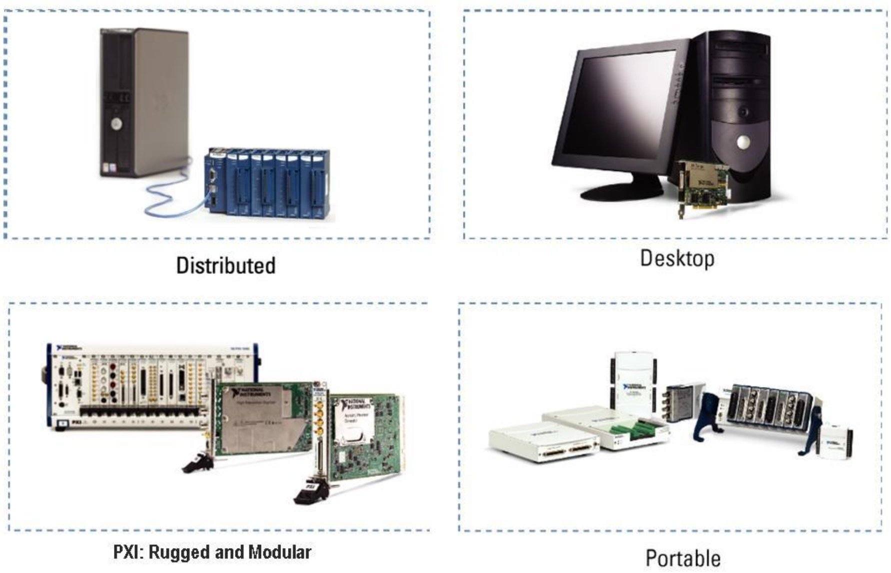
DAQ hardware acts as the interface between the computer and the outside world. It primarily functions as a device that digitizes incoming analog signals so that the computer can interpret them

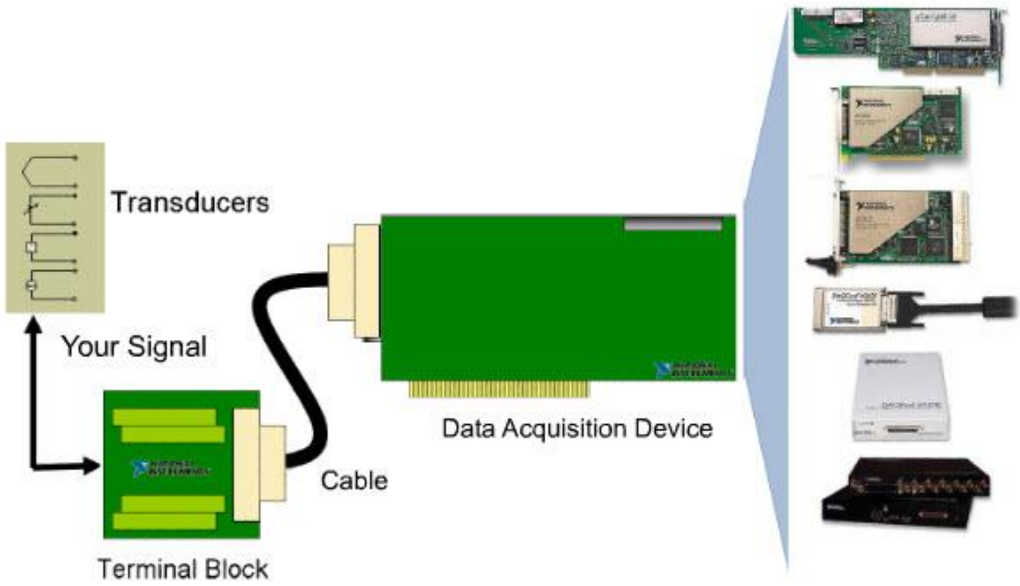
A DAQ device (Data Acquisition Hardware) usually has these functions:

- Analog input
- Analog output
- Digital I/O
- Counter/timers

We have different DAQ devices, such as:

- “**Desktop** DAQ devices” where you need to plug a PCI DAQ board into your computer. The software is running on a computer.
- “**Portable** DAQ devices” for connection to the USB port, Wi-Fi connections, etc. The software is running on a computer
- “**Distributed** DAQ devices” where the software is developed on your computer and then later downloaded to the distributed DAQ device.





2.1.3 Driver software

Driver software is the layer of software for easily communicating with the hardware. It forms the middle layer between the application software and the hardware. Driver software also prevents a programmer from having to do register-level programming or complicated commands in order to access the hardware functions.

Driver software from National Instruments: NI-DAQmx

2.1.4 Your software application

Application software adds analysis and presentation capabilities to the driver software. Your software application normally does such tasks as:

- Real-time monitoring
- Data analysis
- Data logging
- Control algorithms
- Human machine interface (HMI)

In order to create your DAQ application you need a programming development tool, such as Visual Studio/C#, LabVIEW, etc..

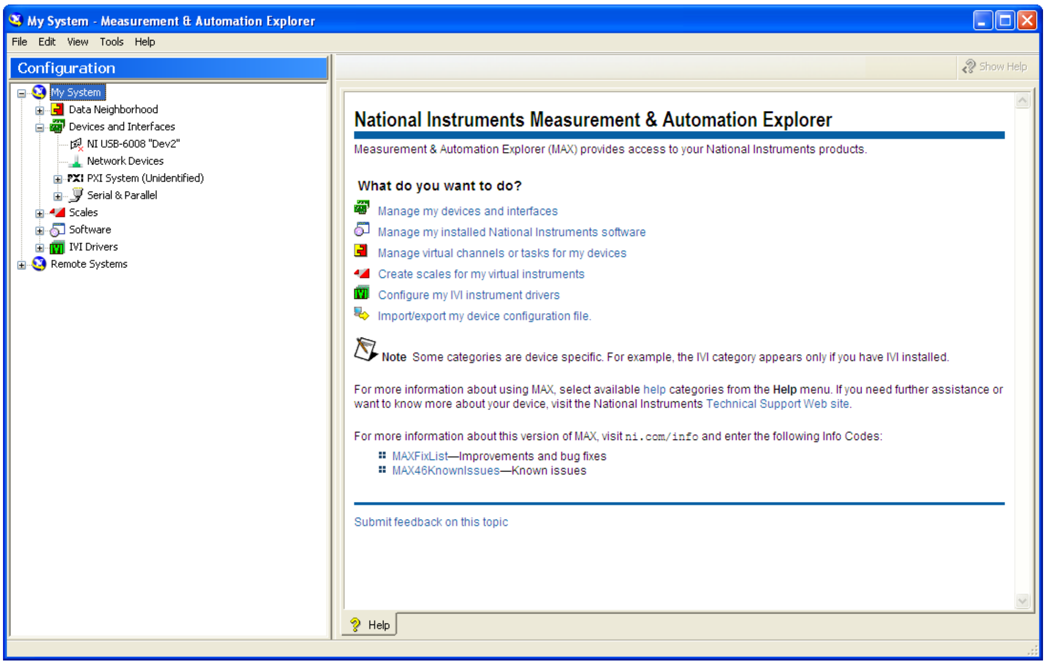
2.2 MAX – Measurement and Automation Explorer

Measurement & Automation Explorer (MAX) provides access to your National Instruments devices and systems.

With MAX, you can:

- Configure your National Instruments hardware and software
- Create and edit channels, tasks, interfaces, scales, and virtual instruments
- Execute system diagnostics
- View devices and instruments connected to your system
- Update your National Instruments software

In addition to the standard tools, MAX can expose item-specific tools you can use to configure, diagnose, or test your system, depending on which NI products you install. As you navigate through MAX, the contents of the application menu and toolbar change to reflect these new tools.



2.3 DAQ in MATLAB

We can create DAQ applications with or without Measurement Studio. In both situations you need the NI-DAQmx driver library.

2.3.1 NI-DAQmx

National Instruments provides a native .NET API for NI-DAQmx. This is available as a part of the NI-DAQmx driver and does not require Measurement Studio.

In general, data acquisition programming with DAQmx involves the following steps:

- Create a Task and Virtual Channels
- Start the Task
- Perform a Read operation from the DAQ
- Perform a Write operation to the DAQ
- Stop and Clear the Task.

Data acquisition in text based-programming environment is very similar to the LabVIEW NI-DAQmx programming as the functions calls is the same as the NI-DAQmx VI's.

3 Data Acquisition Toolbox

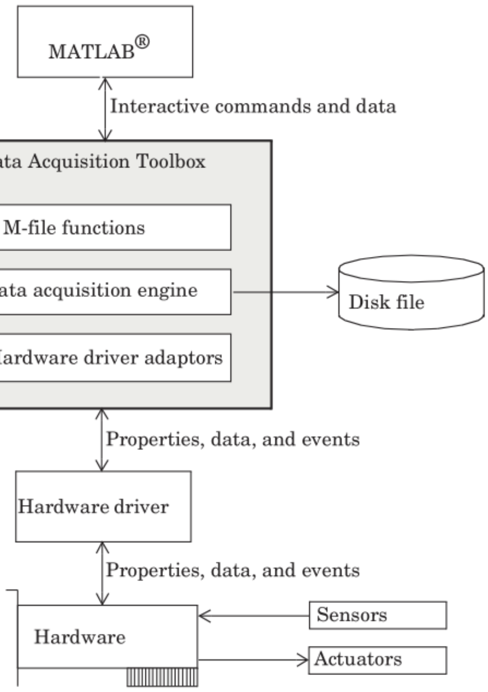
Data Acquisition Toolbox software provides a complete set of tools for analog input, analog output, and digital I/O from a variety of PC-compatible data acquisition hardware. The toolbox lets you configure your external hardware devices, read data into MATLAB and Simulink environments for immediate analysis, and send out data.

Data Acquisition Toolbox also supports Simulink with blocks that enable you to incorporate live data or hardware configuration directly into Simulink models. You can then verify and validate your model against live, measured data as part of the system development process.

We will use the Data Acquisition Toolbox in order to write and read data to and from a USB-6008 DAQ device from National Instruments.

Note! In addition you need to install the **NI DAQmx** driver from National Instruments.

Below we see the data flow from the sensors to the MATLAB:



3.1 Getting Help

To determine if Data Acquisition Toolbox software is installed on your system, type

```
ver
```

This will list all your Toolkits that you have installed and the version numbers.

In order to get an overview of the Data Acquisition Toolbox you can type the following in the MATLAB Command window:

```
help daq
```

Then you will get an overview of all the functions available in the Data Acquisition Toolbox.

This Toolbox has DAQ functionality both for MATLAB and Simulink.

You can view the code for any function by typing:

```
type function_name
```

You can view the help for any function by typing:

```
help function_name
```

4 My First DAQ App

4.1 Introduction

We will use the **Data Acquisition Toolbox** in MATLAB to create a simple Data Acquisition application.

A Simple DAQ application should follow these steps:

1. Initialization
2. Read/Write
3. Clean Up

We will explain the different steps below:

1 - Initialization:

Creating a Device Object:

In Initialization you need to specify what kind of device you are using. We can use the **analoginput()** and **analogoutput()** functions in the Data Acquisition Toolbox.

Example:

```
ai = analoginput('nidaq', 'Dev1');
```

and:

```
ao = analogoutput('nidaq', 'Dev1');
```

The Data Acquisition Toolbox supports DAQ devices from different vendors. In order to use a device from National Instruments, we need to set "nidaq" as the adapter name. "DevX" is the default name created by the system, se MAX (Measurement and Automation Explorer) for details about your device.

Adding Channels:

Next we need to specify which channel(s) we want to use. We can use the **addchannel()** function.

Example:

```
ai0 = addchannel(ai, 0);
```

2 - Read/Write:

If we want to write a single value to the DAQ device, we can use the **putsample()** function.

Example:

```
ao_value = 3.5;
putsample(ao, ao_value)
```

If we want to read a single value from the DAQ device, we can use the **getsample()** function.

Example:

```
ai_value = getsample(ai)
```

3 - Clean Up:

When we are finished with the Data Acquisition we need to close or delete the connection. We can use the **delete()** function.

Example:

```
delete(ai)
```

4.2 Source Code

In this simple example we will create a m-file that write one single value to the DAQ device and then read one single value from the DAQ device.

We start by connecting the Analog In and Analog Out wires together on the DAQ device (a so called Loopback connection).

If we write, e.g., 3.5V to the DAQ device on a AO channel, we will then read the same value on the AI channel.

Source Code for a Simple DAQ Example in MATLAB:

```
% Write and Read to a NI USB-6008 DAQ device
clear
clc
% Initialization-----------------------
% Analog Input:
ai = analoginput('nidaq', 'Dev1');
% Analog Output:
ao = analogoutput('nidaq', 'Dev1');

% Adding Channels-----------------------
% Analog Input - Channel 0
ai0 = addchannel(ai, 0);
% Analog Output - Channel 0
ao0 = addchannel(ao, 0);

% Write Data--------------------------
ao_value = 3.5;
```

```
putsample(ao, ao_value)

% Read Data----------------------------
ai_value = getsample(ai)

% Cleaning Up--------------------------
delete(ai)
delete(ao)
```

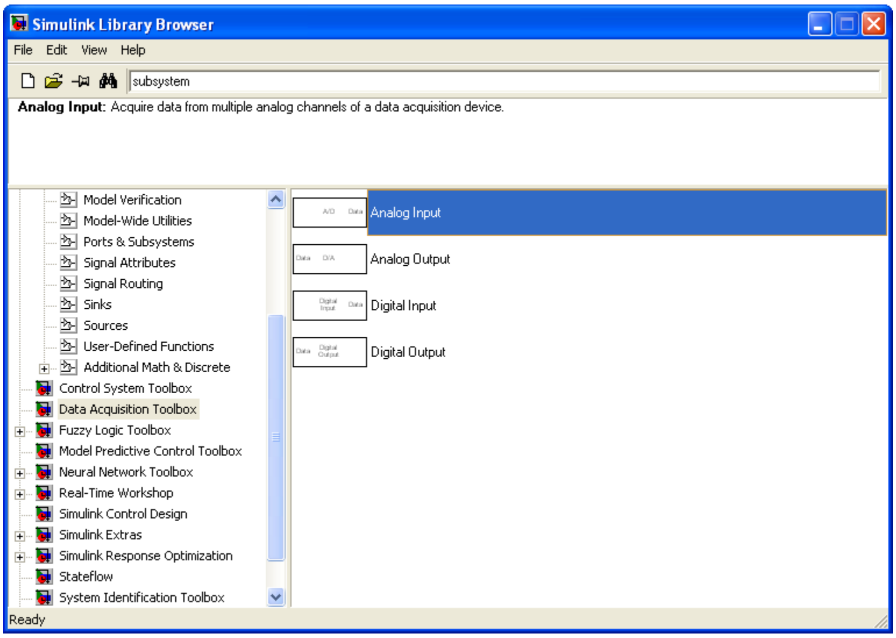
5 DAQ in Simulink

Simulink has built-in blocks for Data Acquisition, but depending on the version of MATLAB/Simulink you are using they might not work properly with the USB-6008 DAQ device. In that case you can call MATLAB functions from Simulink.

We will create a simple Simulink application where you write and read values from the USB-6008 DAQ device.

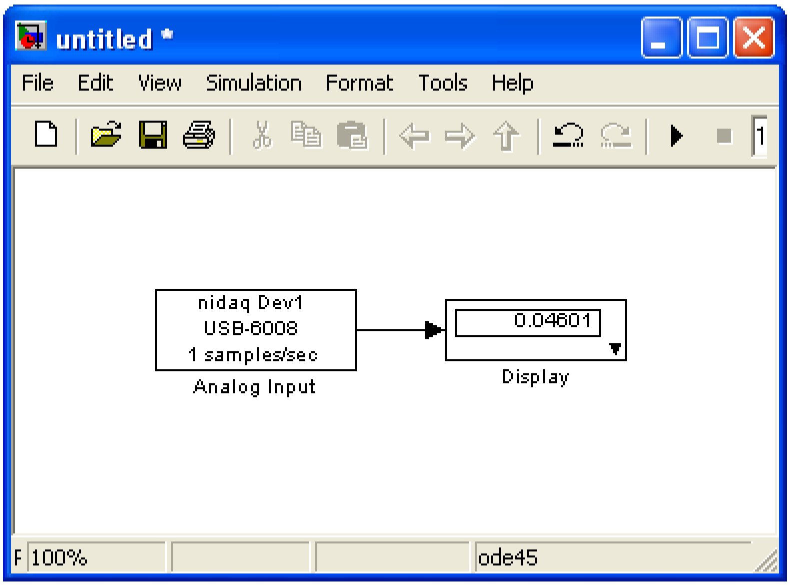
In this chapter MATLAB/Simulink R2007a is used. The Data Acquisition Toolbox has been updated since this release.

Below we see the built-in blocks in Simulink/Data Acquisition Toolbox (R2007a):

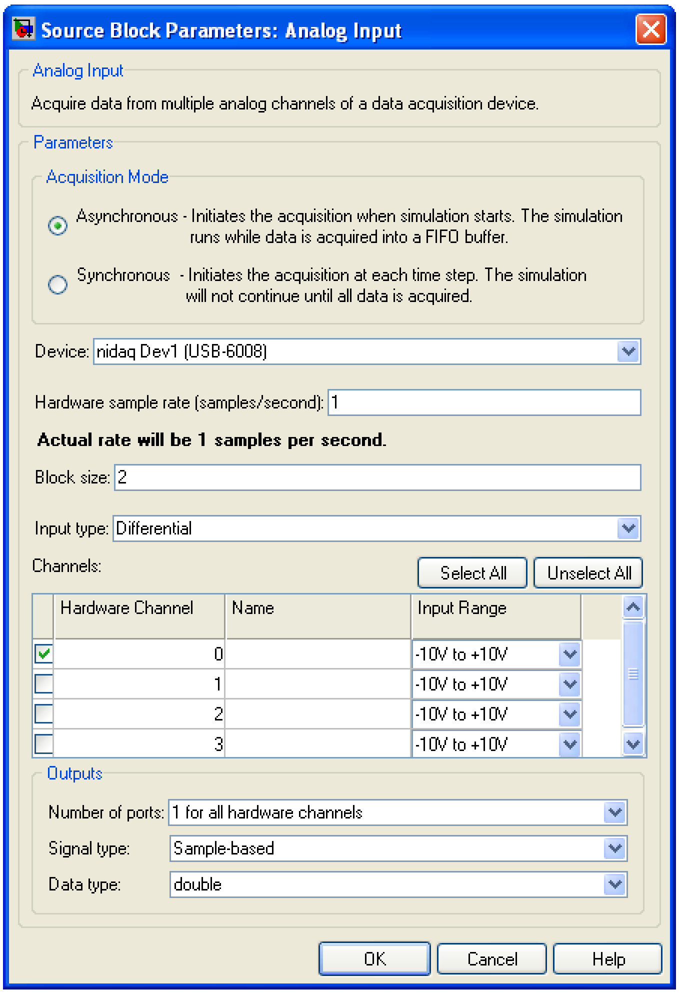


5.1 Analog In

We create a simple application in Simulink in order to demonstrate how to read from the DAQ device:



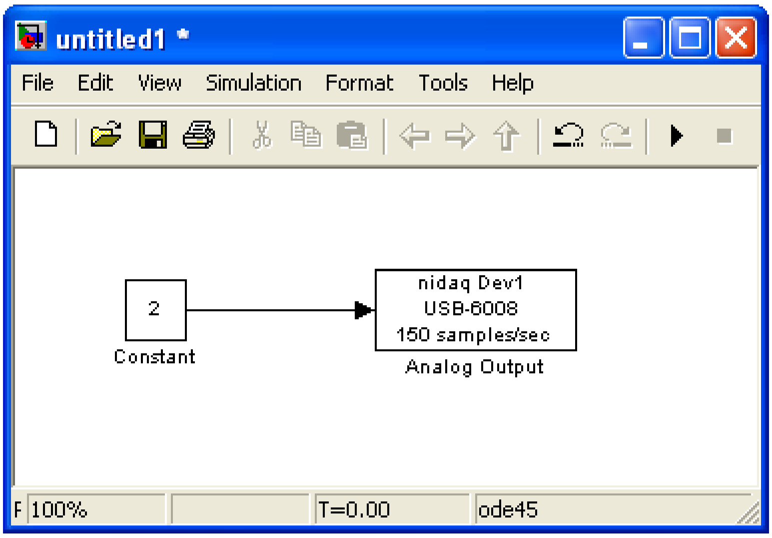
Properties for Analog Input Block:



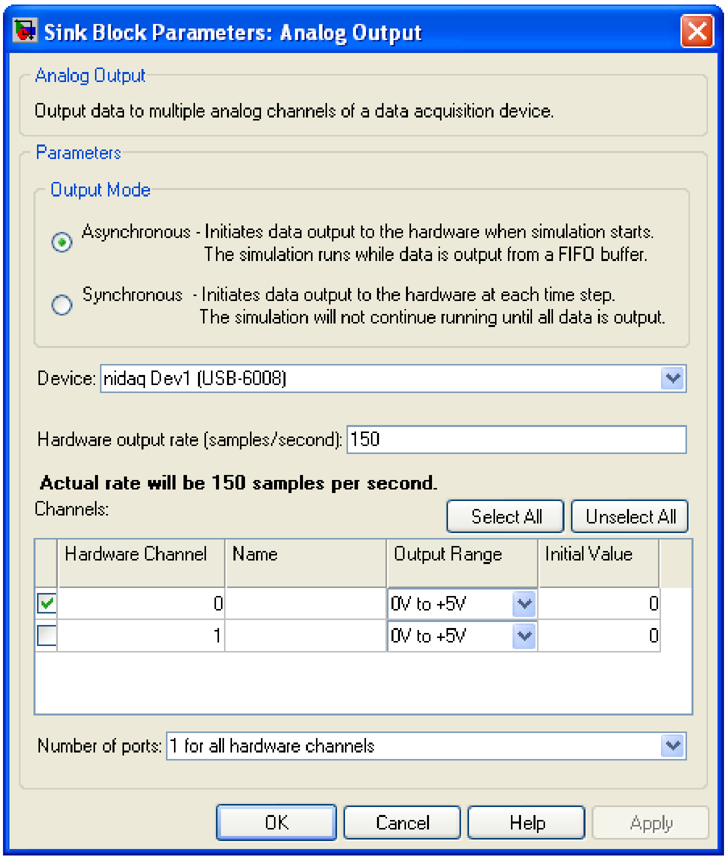
In the Properties window we can define channels, sample rate, etc.

5.2 Analog Out

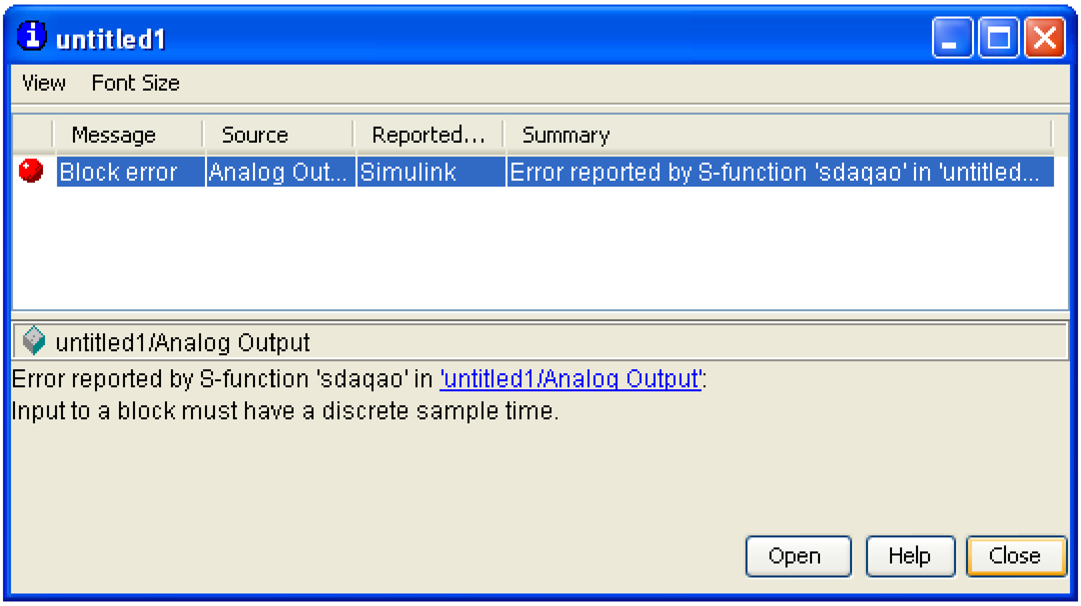
We create a simple application in Simulink in order to demonstrate how to write to the DAQ device:



Properties for Analog Output Block:

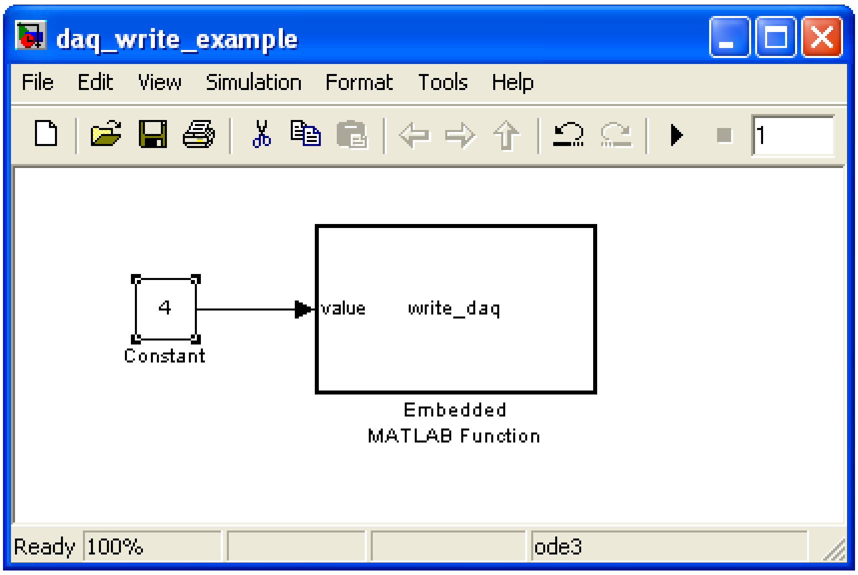


In this case we get the following error:

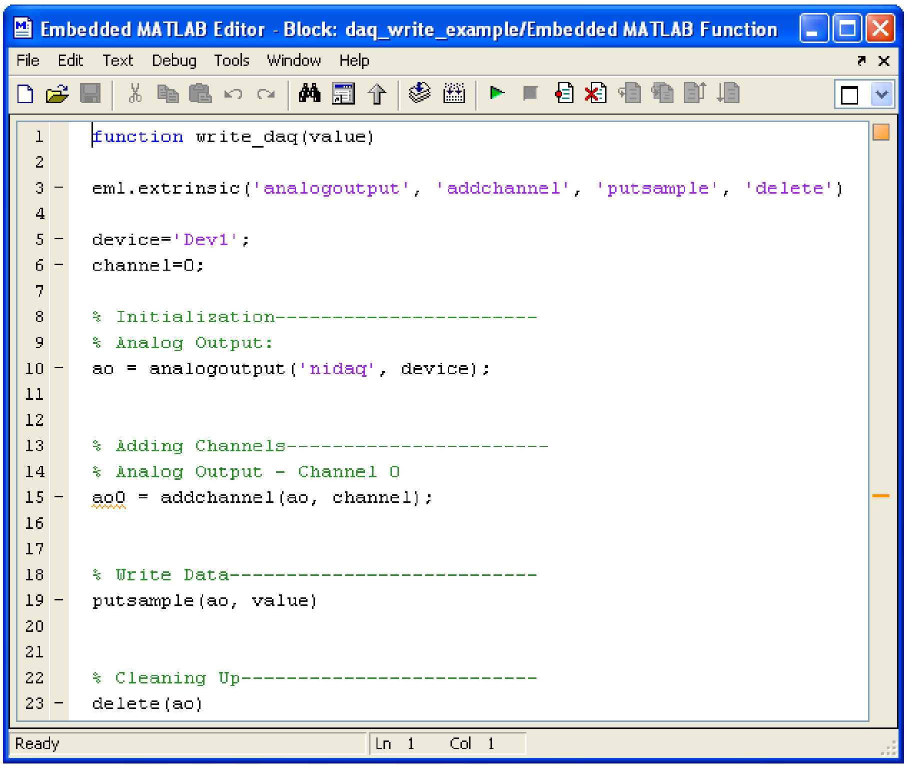


→ This is due to that the Simulink blocks only support hardware that has internal clocking, which the NI USB-6008 does not on analog output. You'll need to call into a MATLAB function to do PUTSAMPLE.

To solve this problem, we can use the "**Embedded MATLAB function**" block in Simulink where we implement MATLAB code for the write operation:



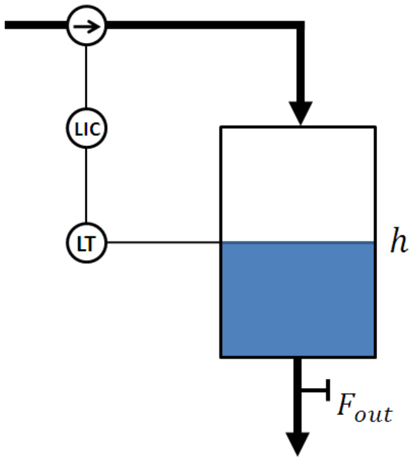
The Embedded MATLAB code is as follows:


Embedded MATLAB Editor - Block: daq_write_example/Embedded MATLAB Function
File Edit Text Debug Tools Window Help
function write_daq(value)
eml.extrinsic('analogoutput', 'addchannel', 'putsample', 'delete')
device='Dev1';
channel=0;
% Initialization-----------------------
% Analog Output:
ao = analogoutput('nidaq', device);
% Adding Channels-----------------------
% Analog Output - Channel 0
ao0 = addchannel(ao, channel);
% Write Data--------------------------
putsample(ao, value)
% Cleaning Up-------------------------
delete(ao)
Ready
Ln 1 Col 1

6 Control Application

6.1 Introduction

In this example we will use Measurement Studio to create a simple control application. We will control the level in a water tank using manual control. The process is as follows:



We want to control the level in the water tank using a pump on the inflow. We will read the level using our USB-6008 DAQ device (Analog In) and write the control signal (Analog Out) to the DAQ device.

The Analog Out (control signal) will be a signal between $0 - 5V$ and the Analog In (Level) will be a $0 - 5V$ signal that we need to scale to $0 - 20cm$.

The next improvements to our application would be to implement a Low-pass Filter in order to remove the noise from the signal when reading the level. Another improvement would be to replace the manual control with a PI controller that do the job for us. Finally it would be nice to have a mathematical model of our water tank so we can simulate and test the behavior of the real system without connect to it.

So we need to create discrete versions of the low-pass filter, the PI controller and the process model. We can, e.g., use the Euler Forward discretization method:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$$

or the Euler Backward discretization method:

$$\dot{x} \approx \frac{x_k - x_{k-1}}{T_s}$$

T_s is the Sampling Time.

6.2 Low-pass Filter

The transfer function for a first-order low-pass filter may be written:

$$H(s) = \frac{y_f(s)}{y(s)} = \frac{1}{T_f s + 1}$$

Where T_f is the time-constant of the filter, $y(s)$ is the filter input and $y_f(s)$ is the filter output.

Discrete version:

It can be shown that a discrete version can be stated as:

$$\boldsymbol{y_{f,k} = (1 - a)y_{f,k-1} + a y_k}$$

Where

$$\boldsymbol{a = \frac{T_s}{T_f + T_s}}$$

Where T_s is the Sampling Time.

6.3 PI Controller

A PI controller may be written:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i}\int_0^t e d\tau$$

Where u is the controller output and e is the control error:

$$e(t) = r(t) - y(t)$$

PI Controller as a Transfer function:

Laplace:

$$u(s) = K_p e(s) + \frac{K_p}{T_i s} e(s)$$

This gives the following transfer function:

$$H_{PI}(s) = \frac{u(s)}{e(s)} = K_P + \frac{K_p}{T_i s} = \frac{K_P(T_i s + 1)}{T_i s}$$

i.e,

$$\boldsymbol{H_{PI}(s) = \frac{K_P(T_i s + 1)}{T_i s}}$$

PI Controller as a State-space model:

We set $z = \frac{1}{s} e \Rightarrow sz = e \Rightarrow \dot{z} = e$

This gives:

$$\dot{z} = e$$

$$u = K_p e + \frac{K_p}{T_i} z$$

Where

$$e = r - y$$

Discrete version:

Using Euler:

$$\dot{z} \approx \frac{z_{k+1} - z_k}{T_s}$$

Where T_s is the Sampling Time.

This gives:

$$\frac{z_{k+1} - z_k}{T_s} = e_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

Finally:

$$\boldsymbol{e_k = r_k - y_k}$$

$$\boldsymbol{u_k = K_p e_k + \frac{K_p}{T_i} z_k}$$

$$\boldsymbol{z_{k+1} = z_k + T_s e_k}$$

This algorithm can easily be implemented in C#.

6.4 Process Model

A very simple (linear) model of the water tank is as follows:

$$A_t \dot{h} = K_p u - F_{out}$$

or

$$\dot{h} = \frac{1}{A_t}[K_p u - F_{out}]$$

Where:

- h [cm] is the level in the water tank
- u [V] is the pump control signal to the pump
- A_t [cm2] is the cross-sectional area in the tank
- K_p [(cm3/s)/V] is the pump gain
- F_{out} [cm3/s] is the outflow through the valve (this outflow can be modeled more accurately taking into account the valve characteristic expressing the relation between pressure drop across the valve and the flow through the valve).

We can use the Euler Forward discretization method in order to create a discrete model:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$$

Then we get:

$$\frac{h_{k+1} - h_k}{T_s} = \frac{1}{A_t}[K_p u_k - F_{out}]$$

Finally:

$$\boldsymbol{h_{k+1} = h_k + \frac{T_s}{A_t}[K_p u_k - F_{out}]}$$

Høgskolen i Telemark

Telemark University College

Faculty of Technology

Kjølnes Ring 56

N-3914 Porsgrunn, Norway

www.hit.no

Hans-Petter Halvorsen, M.Sc.

Telemark University College

Department of Electrical Engineering, Information Technology and Cybernetics

Phone: +47 3557 5158

E-mail: hans.p.halvorsen@hit.no

Blog: http://home.hit.no/~hansha/

Room: B-237a
