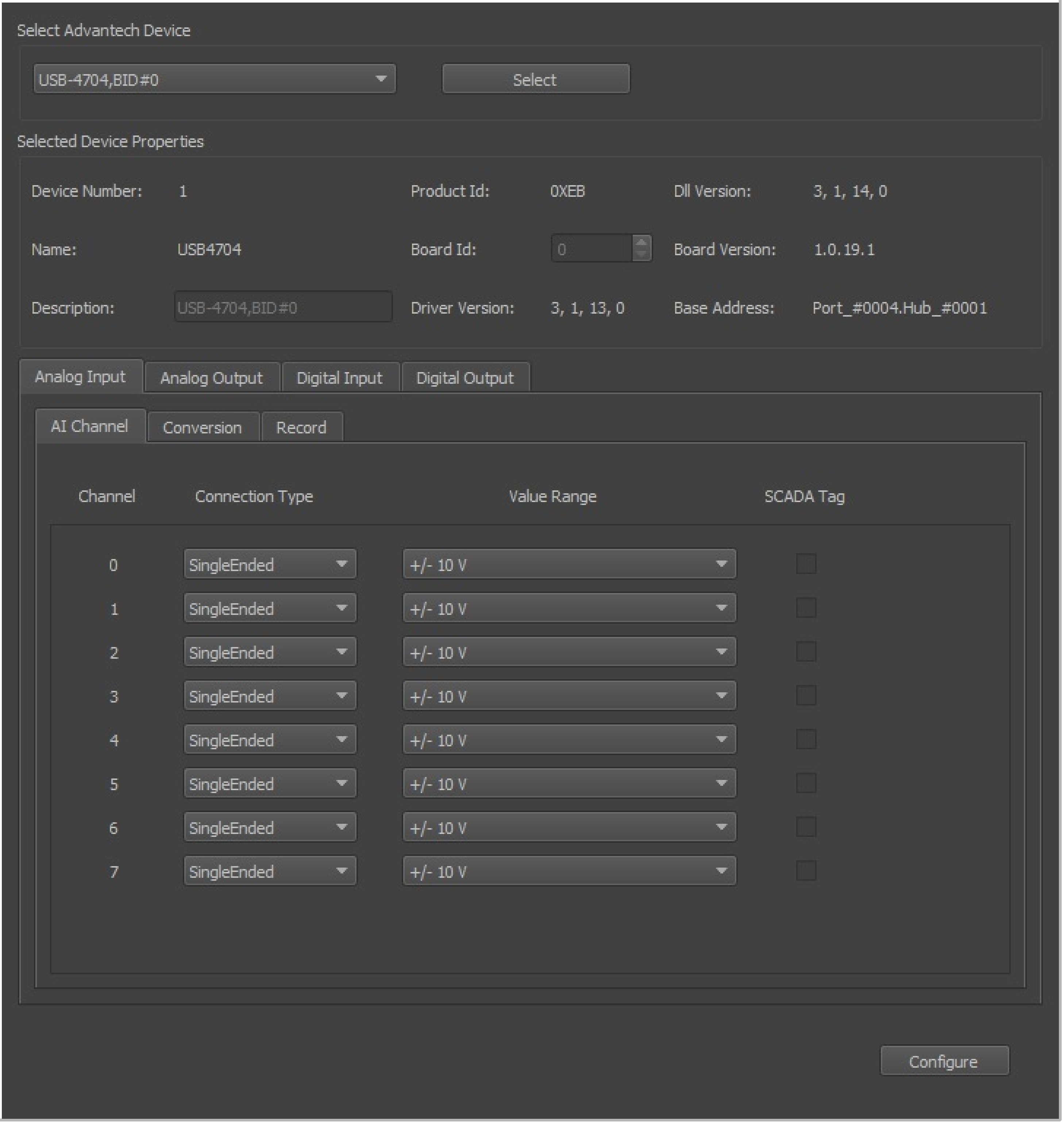
Advantech Devices and MatDeck Features

MatDeck allows its users to combine MatDeck's C++ style script, Drawings, Diagrams, Mathematical functions, Documentations, Tables, and other features all in the same MatDeck file/ document. All MatDeck features and functions are all available to be used with Advantech devices. In this example, we illustrate how Advantech devices are used with MatDeck. The device is configured using MatDeck's simple GUI form which is designed to make it quick and code-less. Once configured, the MatDeck mathematical functions used are for generating signals . The results are recorded in a .xls file in the form of a report.

```
form := atconfig_form(0 , "Form1" , "USB-4704,BID#0")
```



Select Advantech Device

USB-4704,BID#0 | Select

Selected Device Properties

Device Number:	1	Product Id:	0XEB	Dll Version:	3, 1, 14, 0
Name:	USB4704	Board Id:	0	Board Version:	1.0.19.1
Description:	USB-4704,BID#0	Driver Version:	3, 1, 13, 0	Base Address:	Port_#0004.Hub_#0001

Channel	Connection Type	Value Range	SCADA Tag
0	SingleEnded	+/- 10 V	☐
1	SingleEnded	+/- 10 V	☐
2	SingleEnded	+/- 10 V	☐
3	SingleEnded	+/- 10 V	☐
4	SingleEnded	+/- 10 V	☐
5	SingleEnded	+/- 10 V	☐
6	SingleEnded	+/- 10 V	☐
7	SingleEnded	+/- 10 V	☐

Configuring Advantech Devices using GUIs

Advantech devices are be easily set using the GUI, icpcom_multifunction7000_form. All the numerous lines of script code, and more importantly, their arguments are replaced by the GUI's settings. The same form can be used to set up all of the device pins. The document is "a live document" and simultaneously executes commands. This is one of MatDeck's unique advantages.

```
atconfig_form_configure(form)
```

After the Advantech device is set, the analog and digital inputs are read using standard read functions. Analog and digital outputs are also accessed using write functions.

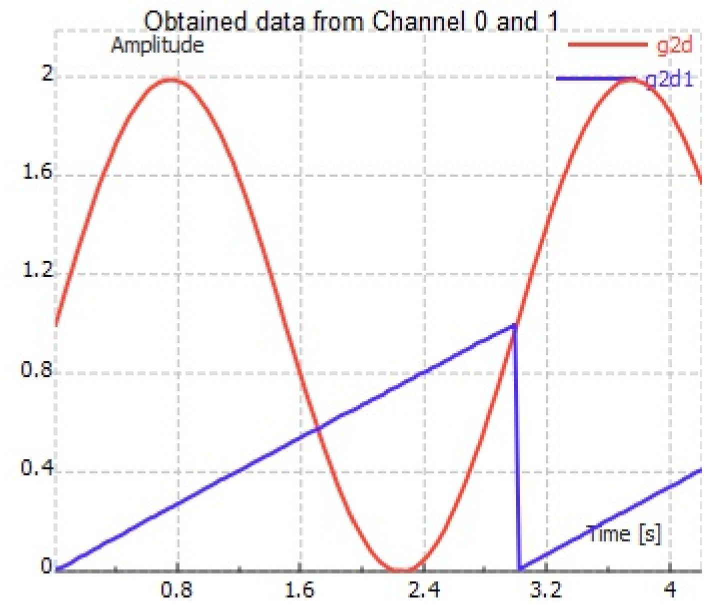
Combining Mathematics with Advantech Devices

We can generate sin functions with given frequencies based on the current time in the MatDeck script. First, the Advantech device needs to be opened.

```
dev1 := atdevice_ai_open("USB-4704,BID#0", false)
dev2 := atdevice_ao_open("USB-4704,BID#0", false)
```

The handle, dev1, is used to access the devices further on in the text. Next, the signals are generated and sent to the analog output channels 0 and 1. Finally, the signal is read at the analog inputs 0 and 1. The readings are then packed into vectors and displayed in a 2d graph .

```
T0 := timenow() // starting time
fre := 1 /3 // frequency of sinusoid in Hz
period := 3 // period of triangular in s (saw tooth function)
numit := 150 //nuber of iterations
val := vector_create(numit, false, 0)
st := vector_create(numit, false, 0)
tim := vector_create(numit, false, 0)
for(i := 0; i < numit; i += 1)
{
  ttemp :=(timenow() -T0) //current time
  temp := sin(2 * fre * cpi() * ttemp) + 1 //current sine value
  stemp := mod(ttemp, period) / period //current saw tooth value
  atdevice_ao_write(dev2, 0, temp) //write AO value at ch 0
  atdevice_ao_write(dev2, 1, stemp)//write AO value at ch 1
  val[i] = atdevice_ai_read(dev1, 0) //read current AI value at ch 0
  tim[i] = ttemp
  st[i] = atdevice_ai_read(dev1, 1) //read current AI value at ch 1
}
g2d := join_mat_cols(tim, val) //prepare graph
g2d1 := join_mat_cols(tim, st) //prepare graph
```



After all the operations are done, the device handle is be released by closing it.

```
atdevice_close(dev1)
atdevice_close(dev2)
```

Results

There will be N measurements taken, which is set above. The first 10 measurements will be automatically displayed in the table. After all N measurements, the data is exported to a .xlsx file. There, we read the voltage at the analog input channels 0 and 1.

```
TableH := ["Time", "Voltage AIN 0", "Voltage AIN 1"]
Data := join_mat_cols(subset(tim, 0, 0, 9, 0),subset(val, 0, 0, 9, 0))
Data = join_mat_cols(Data, subset(st, 0, 0, 9, 0))
Table := table_create(Data, TableH)
```

Table =

Time	Voltage AIN 0	Voltage AIN 1
0.001	0.996	0.007
0.026	1.049	0.011
0.054	1.107	0.017
0.084	1.171	0.026
0.112	1.224	0.035
0.141	1.287	0.049

⋮

Exporting Data to Excel Files

Here, the data obtained by the three measurements will be exported to a Excel file at an appropriate positions. The variables are exported manually.

```
excel_write("measurements.xlsx","Sheet1","A1","Time")
excel_write("measurements.xlsx","Sheet1","B1",tim)
excel_write("measurements.xlsx","Sheet1","A2","Voltage AIN 0")
excel_write("measurements.xlsx","Sheet1","B2",val)
excel_write("measurements.xlsx","Sheet1","A3","Voltage at AIN 1")
excel_write("measurements.xlsx","Sheet1","B3",st)
```

Exporting to Text Files

Measurement data can be exported to a .txt file as well. For example, the .txt file will be made if the maximal voltage at AIN0 is above 0.5V. The sentence "The measurement is successful" will be written in the test.txt file.

```
dat:= datef("d/m/y")
timc:= timef(":")
if(mat_max(val) > 0.5)
{
  file_name:= datef("d-m-y")+" "+ timef("-") + ".txt"
  F:= file_create(file_name,"text",true)
  file_write(F,"The measurement is successful\n")
  file_write(F, mat_max(val))
  file_write(F,"\n"+ dat +" "+ timc)
  file_close(F)
}
```