Advantech Devices and MatDeck

Unlike other available software, MatDeck allows you to integrate and combine MatDeck's C++ style script, Python, Drawings, Diagrams, Mathematical functions, Documentations, Tables, Custom Graphical User Interfaces (MatDeck visual GUI designer) and other features in the same MatDeck file/document. MatDeck and its numerous different features are all now available to Advantech series devices.

Uses of GUIs for Advantech Devices

MatDeck makes it possible to set up multiple Advantech device, the devices can be set via code (Python or MatDeck C++ style script) or via a dedicated GUI for Advantech devices. The settings GUI is integrated together with the user code, which in turn means the GUI is directly combined with the user's code. This unique MatDeck feature will reduce the amount of code needed to be written by a factor of 2 to 5. Additionally, you can avoid coding mistakes as well as simplifying the process of writing code.

At the moment, in MatDeck, there is only one GUI form for configuring all Advantech USB 4704 series devices.

Configuration of Advantech USB4704 Devices

In this example, we will illustrate two alternative ways of configuring the channels for Advantech devices. Advantech functions are present in the functions tab under 'Devices - Advantech', and configuration can be done directly in MatDeck's C++ style Script. Additionally, you can configure devices in Python. However, the most effective and intuitive way for configuring the devices is by using MatDeck's GUI configuration.
This document is "a live document" and simultaneously executes commands, which is one of MatDeck's unique advantages.

The first step is to access the device, USB4704, by opening communications and getting the device handle:

```
dev := atdevice_ai_open(0, false)
```

Here, the first argument is an integer or string which identifies the device. The device's handle is stored as dev after its been opened. For further commands, the device handle is used. Please note that there are several functions for accessing the device for different sets of channels. In the example above, atdevice_ai_open() is used to open the device for analog input - ai configuration and reading.

In order to set the range of an ai channel, the atdevice_ai_set_range() function is used.

```
atdevice_ai_set_range(dev, 0, "+/- 10 V")
```

Here, the channel is identified by an appropriate integer, and range is given by using a string. The input value can be read using the standard ai read function. The result is shown in the canvas:

```
aivalue := atdevice_ai_read(dev, 0)
```

Once done, we will finish closing the device and releasing the handle

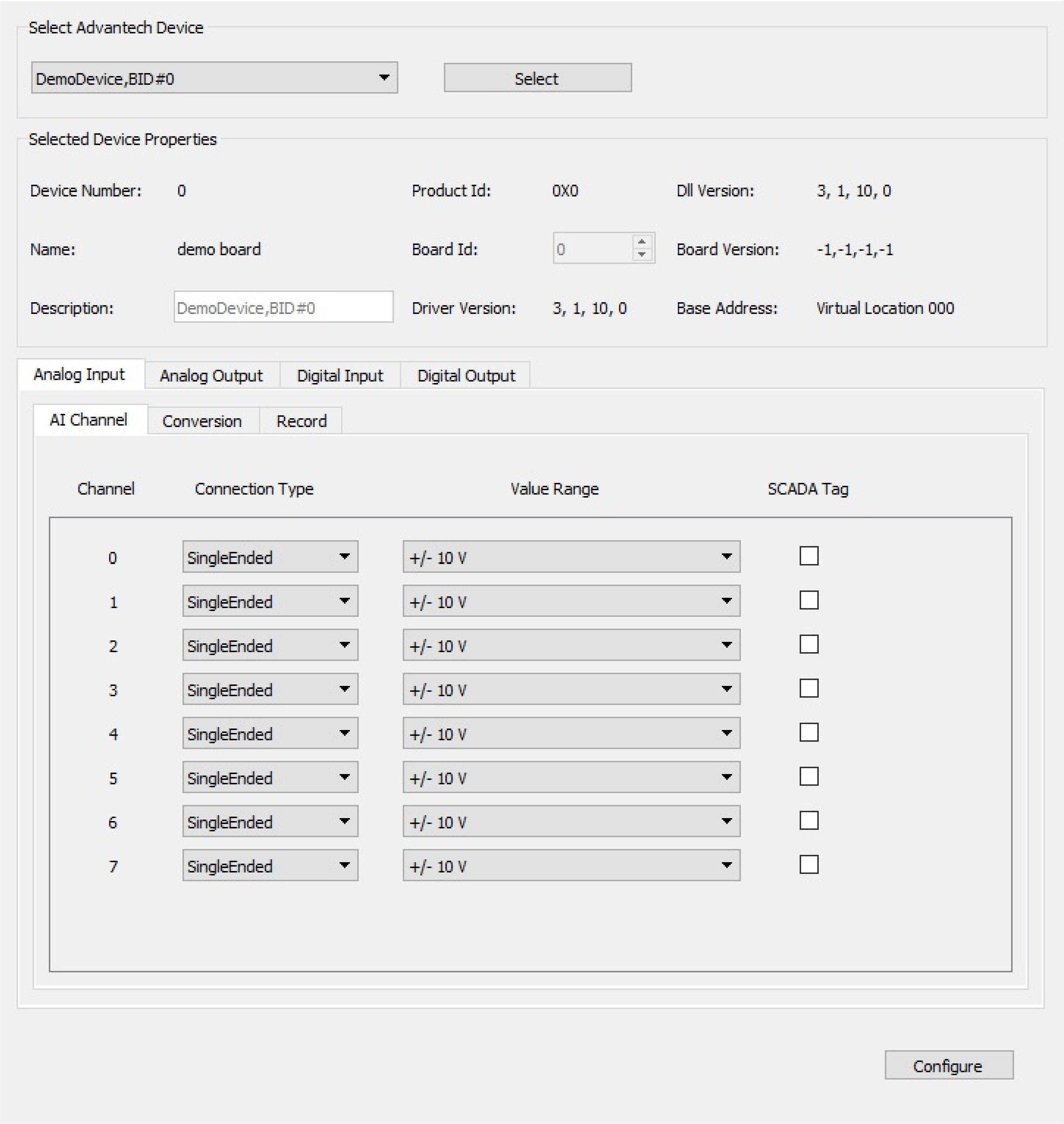
```
atdevice_close(dev)
```

```
aivalue = 4.248
```

GUIs for Configuration

Configuration via code requires knowing the device's configuration functions, strings, and programming to prepare all these scripts in a very intuitive manner. Advantech devices can be easily set by using the GUI form, atconfig_form. All lines of script code, and more importantly, their arguments are replaced by the GUI's settings. The same form can be used to setup all the device pins.

```
form := atconfig_form(0 , "Form 1" , "")
```



Select Advantech Device

DemoDevice,BID#0 ▾ Select

Selected Device Properties

Device Number:	0	Product Id:	0X0	Dll Version:	3, 1, 10, 0
Name:	demo board	Board Id:	0	Board Version:	-1,-1,-1,-1
Description:	DemoDevice,BID#0	Driver Version:	3, 1, 10, 0	Base Address:	Virtual Location 000

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 ▾	□

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

```
dev1 := atdevice_ai_open(0, false)
aivalue1 := atdevice_ai_read(dev1, 0)
atdevice_close(dev1)
```

```
aivalue1 = 4.81
```