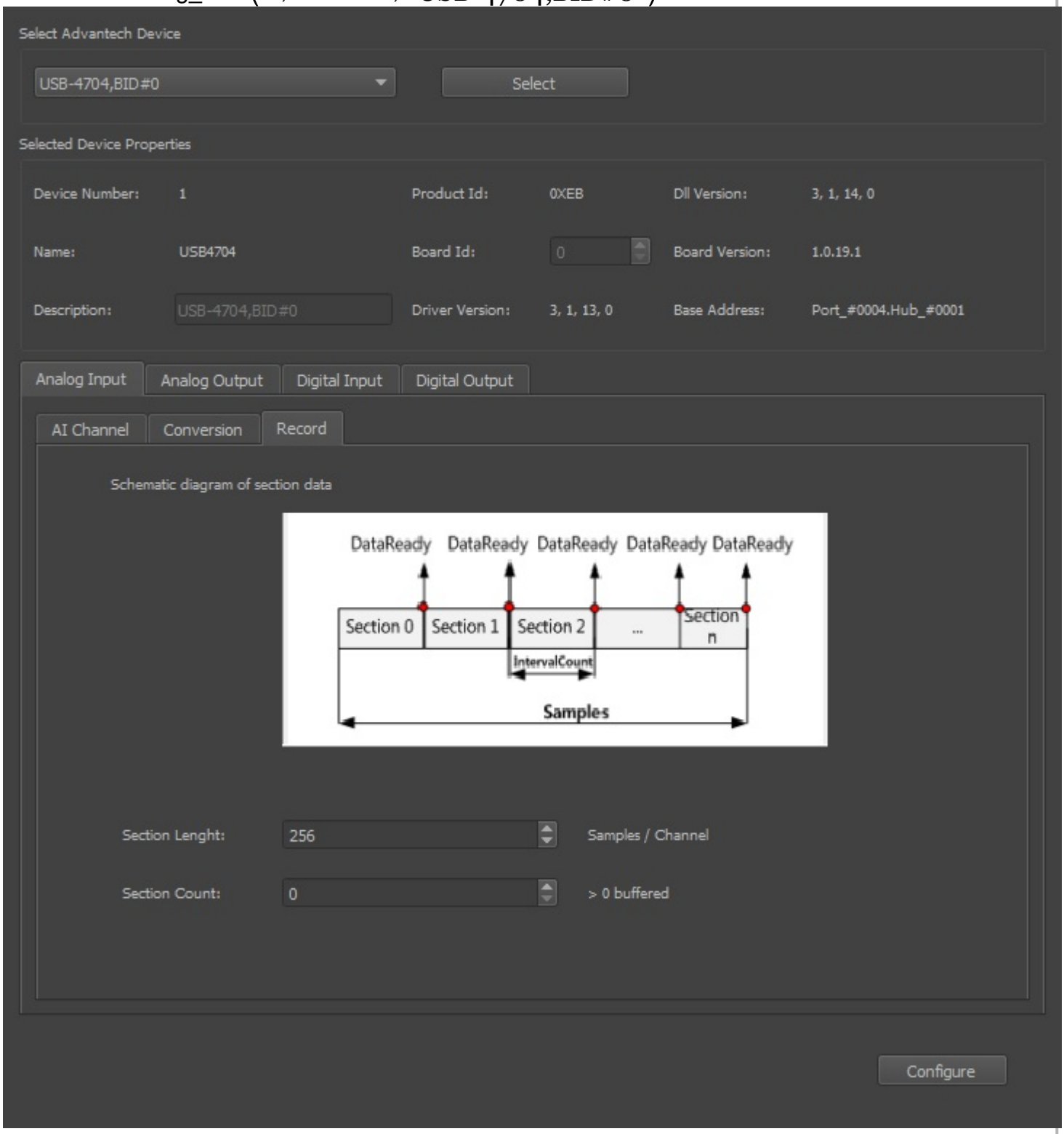
Advantech - AI in buffered Mode

In this example, we illustrate the use of AI for Advantech devices in buffered mode. The best way to set up buffered mode is to use the Advantech GUI. However, one should be careful because the parameters set in the GUI for buffered mode are stored in the MatDeck document and not in the device. That is why we have to export the device handle from the form for further use.

```
at := atconfig_form(0 , "FormAI" , "USB-4704,BID#0")
```



Here, in the canvas below, we can see all available handles that the form produced. In this example, we use AI in buffered mode, and therefore we will continue using the aib handle (AI in buffered mode).

```
atconfig_form_configure(at)

ai := atconfig_form_device_handle(at , "ai")

ao := atconfig_form_device_handle(at , "ao")

di := atconfig_form_device_handle(at , "di")      aob := atconfig_form_device_handle(at , "aob")

do := atconfig_form_device_handle(at , "do")      dib := atconfig_form_device_handle(at , "dib")

aib := atconfig_form_device_handle(at , "aib")    dob := atconfig_form_device_handle(at , "dob")
```

The Advantech device is used for AI in buffered mode, the operation can be divided into four steps. The first step is to define a device that will be used. As explained above, this is done by exporting the appropriate handle from the form.

```
// Step 1: Open AI device in buffered mode.
wfAiCtrl := aib
```

During the next phase, Step 2, all the necessary parameters are set: index of the starting AI channel, total number of AI channels (channelCount), length of the buffer, and buffered mode indicator.

```
// Step 2: Set necessary parameters.
startChannel := 0
channelCount := atdevice_ai_logical_channels(wfAiCtrl)
sectionLength := 1024
sectionCount := 1 //The sectionCount is nonzero value, which means 'One Buffered' mode.
```

Step 3 is data acquisition phase, for that reason the function atdevice_ai_read_multi() is used.

```
//Step 3: GetData
print("Polling finite acquisition is in progress.\n")
scaledData := atdevice_ai_read_multi(wfAiCtrl, startChannel, channelCount)
```

The next code chunk is used to print the first sample of each channel. The correct output is proof that the example was successful.

```
if(type(scaledData) == "vector")
{
      print("The first sample each channel are:")
      for(i := 0; i < channelCount; i += 1)
      {
            print("channel " + to_string(i) + ": " + to_string(scaledData[i]))
      }
}
```

Lastly, Step 4 is to close the device, which is necessary in order to release any allocated resources.

```

```

```
20
21 // Step 4 : Close device and release any allocated resource.
22 atdevice_close(wfAiCtrl)
```

Additionally, we can show the graph of the data read at the channel 0.

```
23 oneWave := size(scaledData) / channelCount
24 y_axis := vector_create(oneWave, false, 0)
25 for(n := 0; n < oneWave; n += 1)
26       y_axis[n] = scaledData[n * channelCount]
27 x_axis := vector_create(oneWave, false, x)
28 graph := join_mat_cols(x_axis, y_axis)
29 graphw := graph2d(0, graph)
```