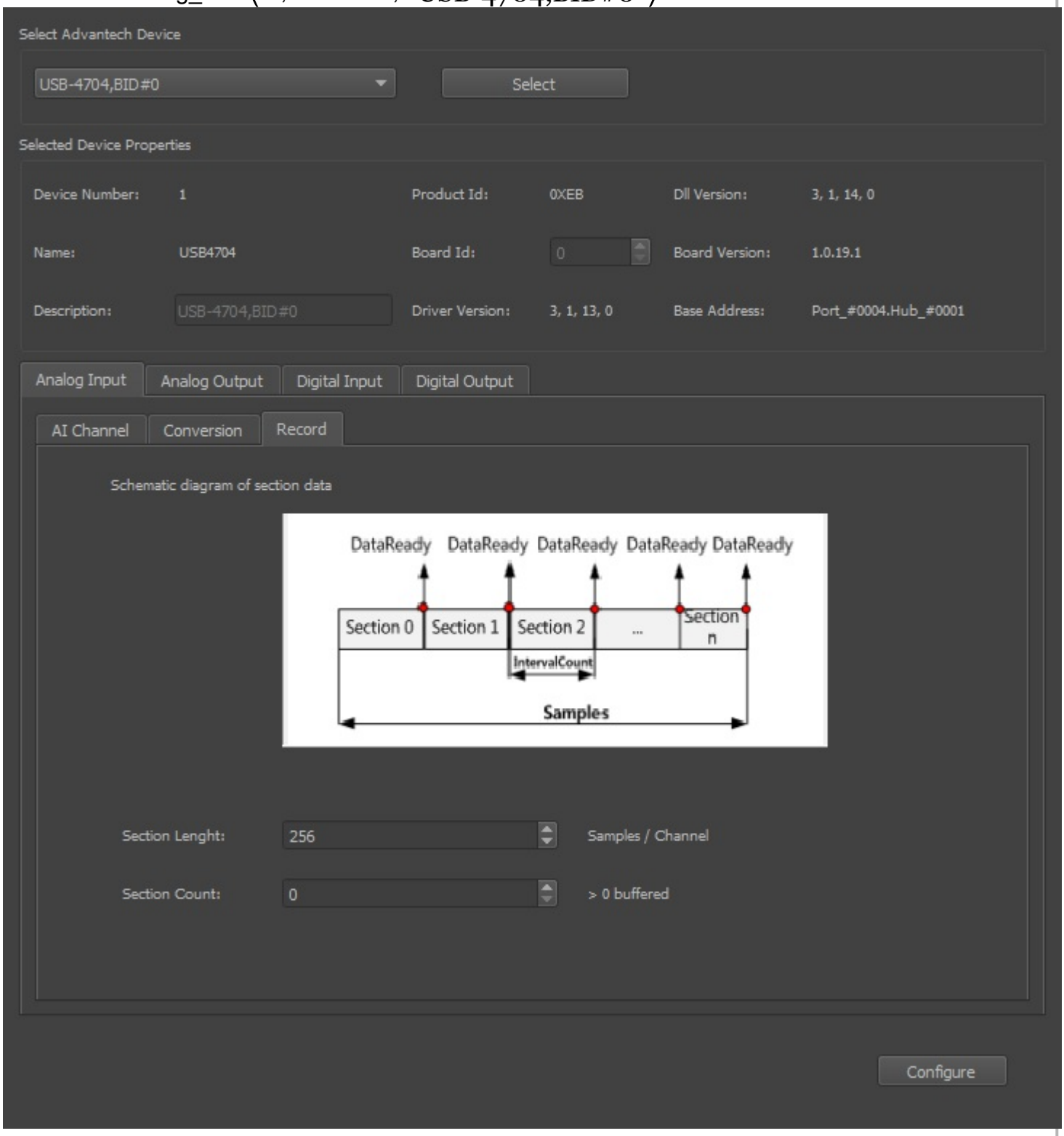
Advantech - AI in buffered Mode

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

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



In this example, we use AI in buffered mode, and therefore we only use the aib handle (AI in buffered mode).

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

The Advantech device is used for the AI in buffered mode. The operation can be divided into four steps. The first step is to define the 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.
AIhandle := aib
```

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

```
// Step 2: Set necessary parameters.
startChannel := 0
channelCount := atdevice_ai_logical_channels(AIhandle)
sectionLength := 1024
```

Step 3 is the data acquisition phase, which is why the function, atdevice_ai_read_multi() is used.

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

The next code segment 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]))
      }
}
```

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

```

// Step 4 : Close device and release any allocated resource.
atdevice_close(AIhandle)
```

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

```
oneWave := size(scaledData) / channelCount
y_axis := vector_create(oneWave, false, 0)
for(n := 0; n < oneWave; n += 1)
      y_axis[n] = scaledData[n * channelCount]
x_axis := vector_create(oneWave, false, x)


```

```
29
30 graph := join_mat_cols(x_axis, y_axis)
31 graphw := graph2d(0, graph)
```