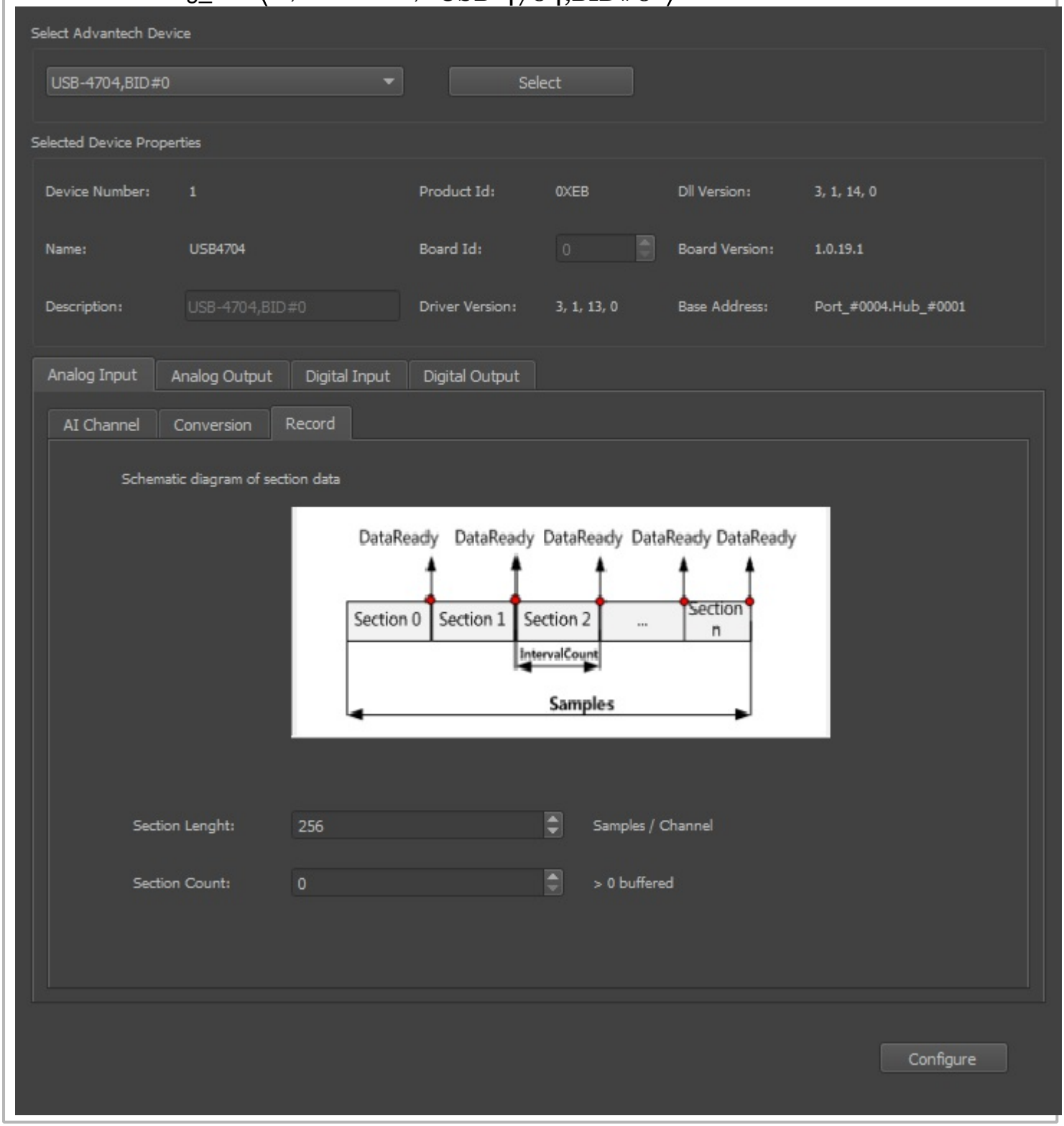
Advantech - AI in Buffered Mode in Real Time with Infinite Loop - Data Processing

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 the 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 is needed for further use.

```
at:= atconfig_form(0 , "FormAI2" , "USB-4704,BID#0")
```

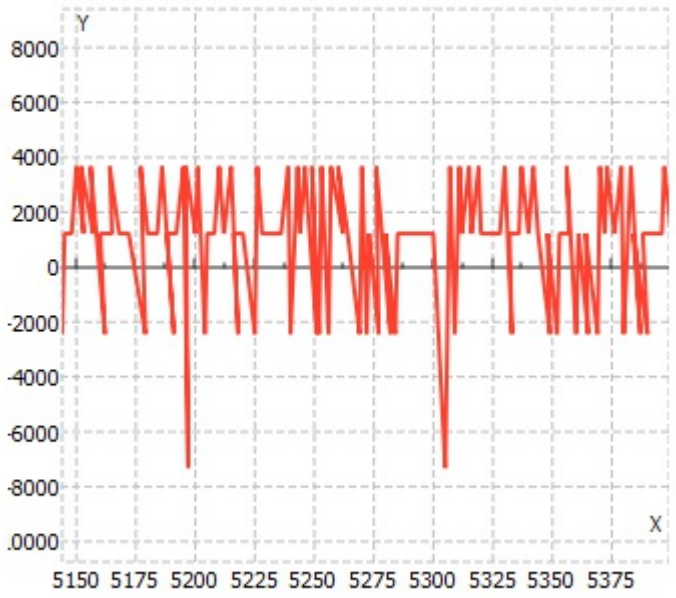


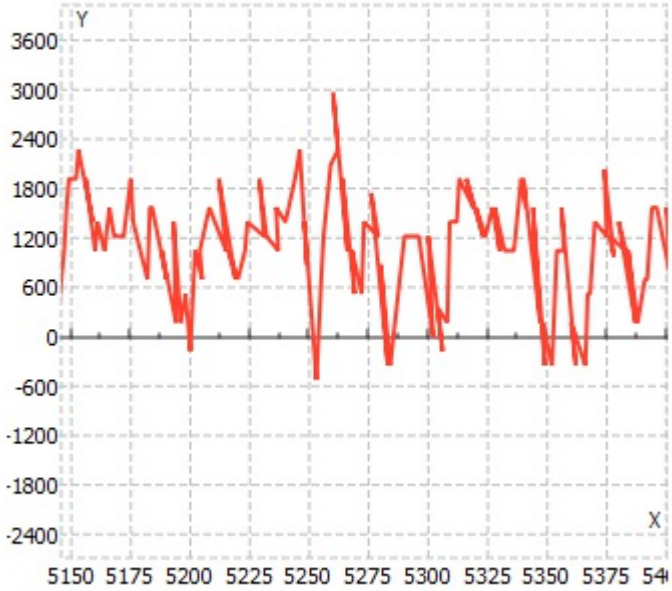
In this example, we use AI in buffered mode, and therefore we only use the aib handle (AI in buffered mode).

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

We use the graph to plot row data read from AI0, and filtered data after the Moving Average filter. Therefore, we define the graph, and global vector to store filtered data.

```
graphWidget := graph2d(0, 0)
graphWidget2 := graph2d(0, 0)
```






```
graph1 := matrix_create(2 , 2 , 0)
```

```
graph2 := matrix_create(2 , 2 , 0)
```

The Advantech device is used for 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 := 256
```

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

function is used in a infinite loop. The execution of the document should be stopped using the Stop option from the Programming tab.

```
//Step 3: GetData
print("Polling finite acquisition is in progress.\n")
scaledData := vector_create(sectionLength * channelCount, false, 0)
oneWave := size(scaledData) / channelCount
y_axis := vector_create(oneWave, false, 0)
y_axis1 := vector_create(oneWave, false, 0)
x_axis1 := vector_create(oneWave, false, 0)
counter := 0

while(true)
{
  scaledData = atdevice_ai_read_multi(AIhandle, startChannel,
channelCount)
  for(n := 0; n < oneWave; n += 1)
        y_axis[n] = scaledData[n * channelCount]
  x_axis := vector_create(oneWave, false, x + counter * sectionLength)
  x_axis1 = x_axis
  counter += 1
  graph1 = join_mat_cols(x_axis, y_axis *1000000)
  set_widget_value(graphWidget,graph1)
  fn()
}

fn()
{
  //data processing
  y_axis1 = movavg(y_axis, 7)
   graph2 = join_mat_cols(x_axis1, y_axis1 *1000000)
   set_widget_value(graphWidget2,graph2)
}
```

The final segment, 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)
```