Advantech - DI and DO in buffered Mode

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

$at := atconfig\_form\left(0\ ,\ "FormAI"\ ,\ "USB-4704,BID\#0"\right)$

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

Analog Input	Analog Output	Digital Input	Digital Output

AI Channel	Conversion	Record

Schematic diagram of section data

Section Lenght:	256	Samples / Channel
Section Count:	0	> 0 buffered

Configure

Here, in the canvas below, we can see all available handles that the form has produced. In the experiment, we use DI and DO in buffered mode, and therefore we continue using the dib handle (DI in buffered mode), and the dob handle (DO 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 DI and DO in buffered mode. The DO port should be connected to the DI port to test the result. The operation can be divided into five 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.

```
1    // Step 1: Open DI device and DO device in buffered mode.
2    DIhandle := dib
3    DOhandle := dob
```

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.

```
4    // Step 2: Set necessary parameters.
5    startPort := 0
6    portCount := atdevice_do_ports(DOhandle)
7    //sectionCount := 1 //The sectionCount is nonzero value, which means 'One
     Buffered' mode.
```

Step 3 is the data output phase, for that purpose, we have to define the output vector which will be written to the output port 0. The length of data is equal to sectionLength.

```
8    //Step 3: Write Data
9    sectionLength := atdevice_do_record(DOhandle)
10   port0data := vector_create(sectionLength, false, 1)
11   for(i := 0; i < sectionLength; i += 1)
12     port0data[i] = mod(i, 256)
13   atdevice_do_write(DOhandle, 0, port0data)
```

In the next phase, Step4, we read input port 0 to check if the data is correctly sent.

```
14   disectionLength := atdevice_di_record(DIhandle)
15   di0data := vector_create(disectionLength, false, 0)
16   di0data = atdevice_di_read(DIhandle, 0)
```

Finally, Step 5 is to close the device, which is necessary in order to release any allocated resources. Additional code can also be easily written to plot the input and output data..

```
17   // Step 5 : Close device and release any allocated resource.
18   atdevice_close(DOhandle)
19   atdevice_close(DIhandle)
```