

# Advantech - Analog Output

In this example, we illustrate how to produce analog outputs at a given AO channel. There are several different ways to produce the output. The first step is to configure the Advantech device using an Advantech GUI.

```
1 AO := atconfig_form(0, "A01", "")
```

The screenshot shows the Advantech GUI for configuring an Analog Output channel. The interface is divided into several sections:

- Select Advantech Device:** A dropdown menu shows "USB-4704,BID#0" and a "Select" button.
- Selected Device Properties:** A table of device information:

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
- Configuration Tabs:** "Analog Input", "Analog Output", "Digital Input", and "Digital Output". The "Analog Output" tab is selected.
- AI Channel Configuration:** Sub-tabs for "AI Channel", "Conversion", and "Record". The "AI Channel" sub-tab is active, showing a table of 8 channels:

Channel	Connection Type	Value Range	SCADA Tag
0	SingleEnded	+/- 10 V	<input type="checkbox"/>
1	SingleEnded	+/- 10 V	<input type="checkbox"/>
2	SingleEnded	+/- 10 V	<input type="checkbox"/>
3	SingleEnded	+/- 10 V	<input type="checkbox"/>
4	SingleEnded	+/- 10 V	<input type="checkbox"/>
5	SingleEnded	+/- 10 V	<input type="checkbox"/>
6	SingleEnded	+/- 10 V	<input type="checkbox"/>
7	SingleEnded	+/- 10 V	<input type="checkbox"/>
- Configure Button:** A "Configure" button is located at the bottom right of the interface.

The easiest way to use the device further on, is to export the device handle from the form, which is done in the canvas below. In the given scenario, we are using the Advantech USB-4704 device. In order to test and display the Analog Output, we have connected AO0 to AI0. We need two handles for AO and AI.

```
atconfig_form_configure(AO)

AOhandle := atconfig_form_device_handle(AO, "ao")

AIhandle := atconfig_form_device_handle(AO, "ai")
```

## Different options for Analog Output generation

### Predefined Output Signal

The waveform data is generated in such a manner that there is a single column vector which contains data to be written to AOs in parallel. Here, the main interest is the signal shape and not timing. We start by defining the length of the signal, x axis, and waveform.

Define the x axis and sine waveform

```
2 ONE_WAVE_POINT_COUNT := 512
3 xaxis := ynodes(x, 0, 511, 512)
4 waveform := sin(2 * cpi() * xaxis / 100)
```

Creating data for the graph

```
5 A01 := join_mat_cols(xaxis, waveform)
6 //Vector to read AI
7 AIread := vector_create(size(waveform), false, 0)
```

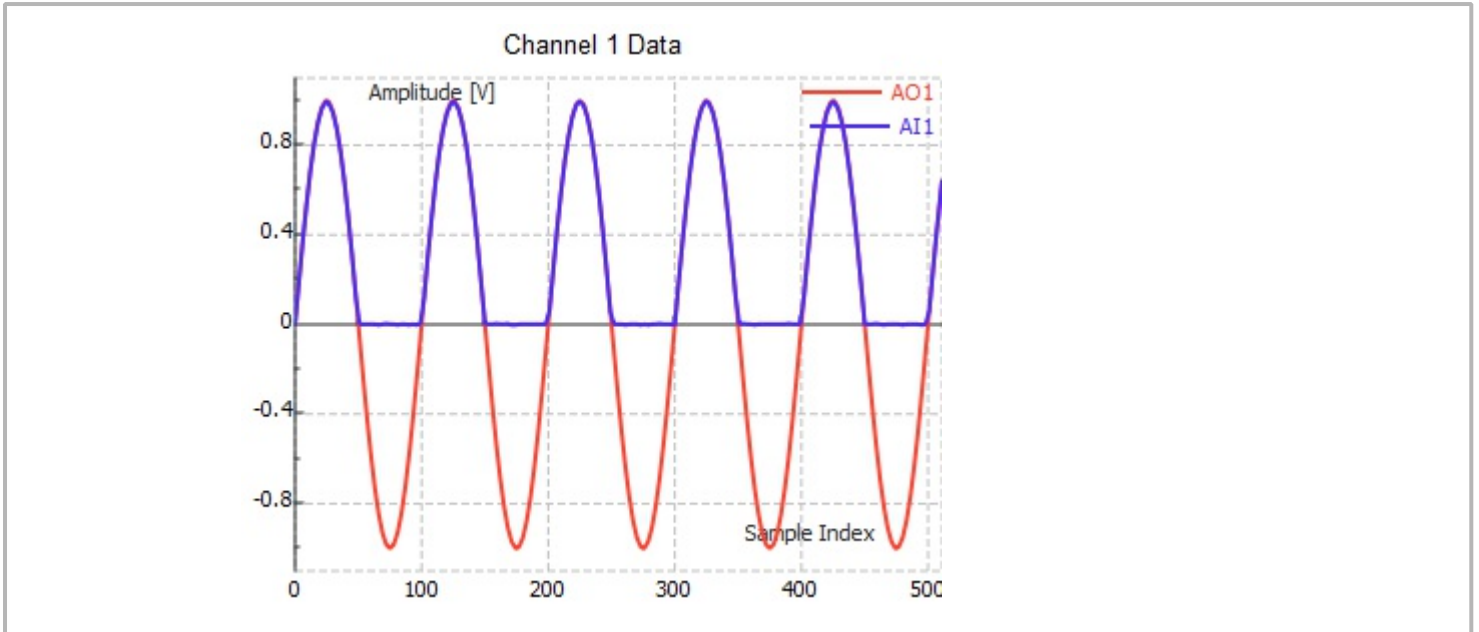
Next, we select the AO channel to write:

```
8 chanStart := 0
```

Loop for writing to A00 and reading from AI0 which is connected to A00.

```
9 for(i := 0; i < ONE_WAVE_POINT_COUNT; i += 1)
10 {
11   atdevice_ao_write(AOhandle, chanStart, waveform[i])
12   AIread[i] = atdevice_ai_read(AIhandle, chanStart)
13 }
14 AI1 := join_mat_cols(xaxis, AIread)
```

In graph below, the x axis is sample index, and it is not related to time.



## Output Signal in Real Time

In real time, the signal which we generate will be related to current time. The frequency of the signal is defined manually. Again, we will generate the same number of samples. The time of execution is measured and it depends on the device which is used. Here, we used the Advantech USB-4704 device.

```

15 freq := 0.5 //Frequency in Hz
16 t0 := timenow()
17 for(i := 0; i < ONE_WAVE_POINT_COUNT; i += 1)
18 {
19   xaxis[i] = timenow() - t0
20   waveform[i] = 5 * sin(2 * cpi() * xaxis[i] * freq)
21   atdevice_ao_write(AOhandle, chanStart, waveform[i])
22   AIread[i] = atdevice_ai_read(AIhandle, chanStart)
23 }
24 write_to_ao0 := join_mat_cols(xaxis, waveform)
25 ain0_read_ao0 := join_mat_cols(xaxis, AIread)

```

The data output at AO0 is displayed in the graph below.

Channel 1 Data

