How to Control Tabor AWGs with MATLAB

Using the IVI driver

In the previous tutorial, we have shown how to communicate with Tabor AWG using SCPI commands. Another way of using the Test & Measurement Tool, is by creating a device object using the Tabor IVI driver. This way, one can communicate with the Tabor AWG, using pre-defined functions. In this tutorial, we will give a quick start guide on how you can communicate with the Tabor AWG using the IVI driver.

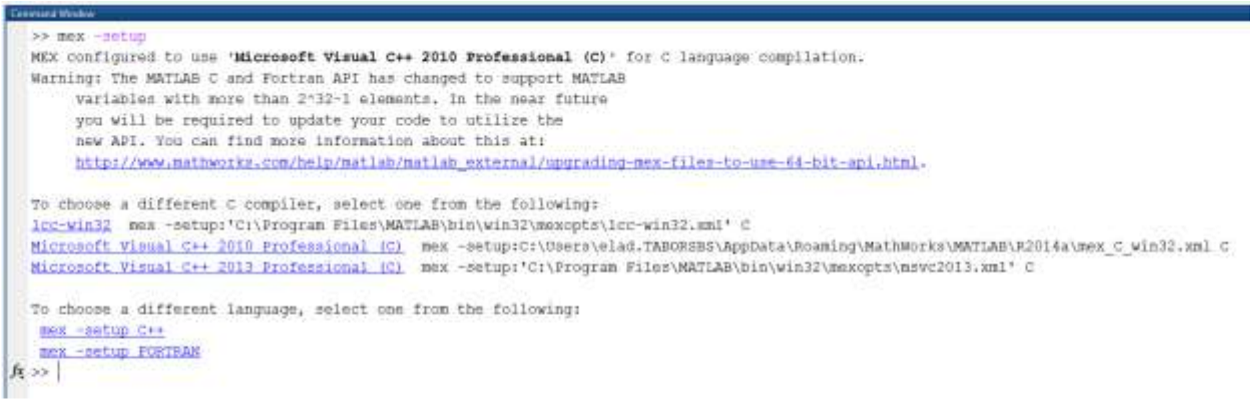
In order to control instruments using MATLAB, the instrument control toolbox is required. Please note that the Instrument Control Toolbox is an additional application that needs to be added. For more information you can visit the Mathworks website.

For this tutorial, we will use MATLAB version R2014a 32bit and a USB interface. We will demonstrate it using Tabor WX2184C and the WX218xx IVI driver. There are differences with operating WW AWGs using the WW257xx IVI driver but the basics are the same. To ensure you successfully established all the necessary settings for remote control over the Tabor instrument using LAN/USB/GPIB, please go over the connectivity tutorials on the Tabor's website.

➔ **To connect and control the Tabor's Instrument using the Test & Measurement Tool**

1. First you may need to choose a MATLAB compiler. There is a need to check that MATLAB uses Microsoft visual C++ compiler or other compiler that can read C (as the IVI driver is written in C language).

   Please enter the following command:



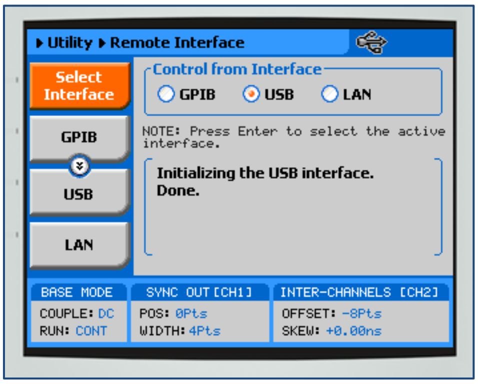
```
>> mex -setup
MEX configured to use 'Microsoft Visual C++ 2010 Professional (C)' for C language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements. In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html.

To choose a different C compiler, select one from the following:
lcc-win32  mex -setup:'C:\Program Files\MATLAB\bin\win32\mexopts\lcc-win32.xml' C
Microsoft Visual C++ 2010 Professional (C)  mex -setup:C:\Users\elad.TABORSBS\AppData\Roaming\MathWorks\MATLAB\R2014a\mex_C_win32.xml C
Microsoft Visual C++ 2013 Professional (C)  mex -setup:'C:\Program Files\MATLAB\bin\win32\mexopts\msvc2013.xml' C

To choose a different language, select one from the following:
 mex -setup C++
 mex -setup FORTRAN
fx >> |
```

   It will start a routine that should allow you to view/choose between all avialable compilers. In case you don't have any C/C++ compiler as an option, please use the following links to finish this routine:

a. https://www.mathworks.com/matlabcentral/answers/101105-how-do-i-install-microsoft-windows-sdk-7-1

b. http://www.mathworks.com/support/compilers/R2016b/

2. Set the USB/LAN/GPIB as the remote interface, using the Tabor's front panel buttons. To do so, go to: "Utility"->"Remote Interface"->"Select Interface"->"Control from Interface". Press Enter to select the active Interface you need. Wait for the answer "Done". We chose to demonstrate using USB.



Control from Interface
○ GPIB ● USB ○ LAN

NOTE: Press Enter to select the active interface.

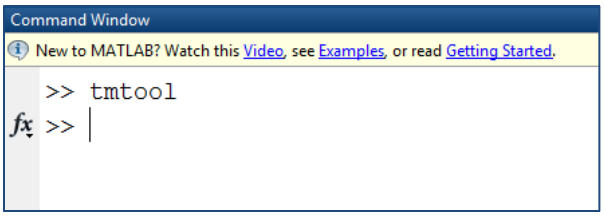
Initializing the USB interface.
Done.

COUPLE: DC
RUN: CONT

POS: 0Pts
WIDTH: 4Pts

OFFSET: -8Pts
SKEW: +0.00ns

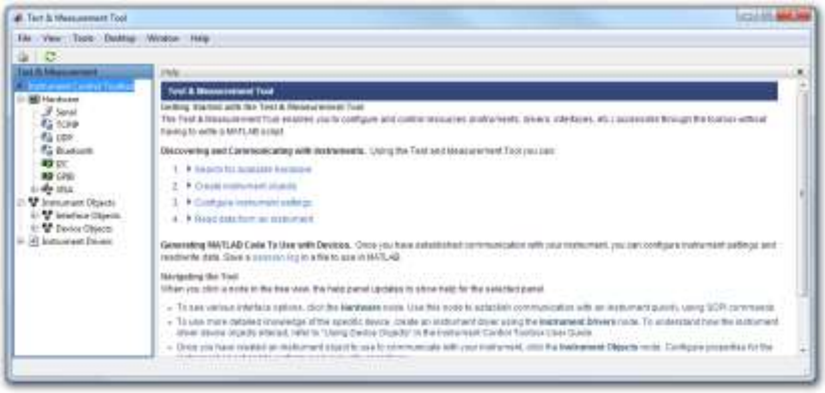
3. Once the MATLAB is up and running, type "tmtool" on the command window to open the Test & Measurement Tool.



New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
>> tmtool
>> 
```

A new Test & Measurement Tool dialog box opens up:



Test & Measurement Tool

Getting Started with the Test & Measurement Tool

The Test & Measurement Tool enables you to configure and control resources (instruments, drivers, interfaces, etc.) accessible through the toolbox without having to write a MATLAB script.

Discovering and Communicating with Instruments. Using the Test and Measurement Tool you can:

1. Search for available hardware
2. Create instrument objects
3. Configure instrument settings
4. Read data from an instrument

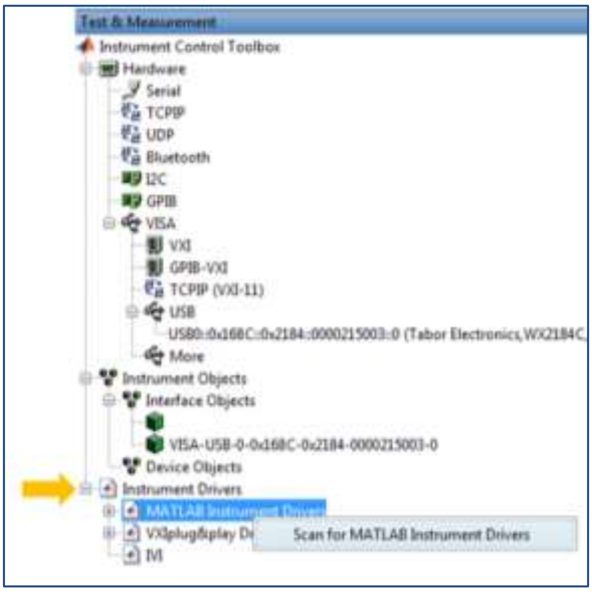
Generating MATLAB Code To Use with Devices. Once you have established communication with your instrument, you can configure instrument settings and read/write data. Save a session log in a file to use in MATLAB.

Navigating the Tool

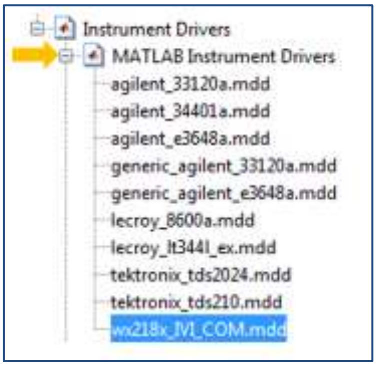
When you click a node in the tree view, the help panel updates to show help for the selected panel.

- To see various interface options, click the **Hardware** node. Use this node to establish communication with an instrument quickly, using SCPI commands.
- To use more detailed knowledge of the specific device, create an instrument driver using the **Instrument Drivers** node. To understand how the instrument driver device objects interact, refer to "Using Device Objects" in the Instrument Control Toolbox User Guide.
- Once you have created an instrument object to use to communicate with your instrument, click the **Instrument Objects** node. Configure properties for the

4. Click on “Instrument Drivers” and then right click on MATLAB Instrument drivers and select “Scan for MATLAB Intstrument Drivers” :


Test & Measurement
Instrument Control Toolbox
Hardware
Serial
TCPIP
UDP
Bluetooth
I2C
GPIB
VISA
VXI
GPIB-VXI
TCPIP (VXI-11)
USB
USB0::0x168C::0x2184::0000215003::0 (Tabor Electronics,WX2184C,
More
Instrument Objects
Interface Objects
VISA-USB-0-0x168C-0x2184-0000215003-0
Device Objects
Instrument Drivers
MATLAB Instrument Drivers
VXIplug&play Dr
IVI
Scan for MATLAB Instrument Drivers

5. Click on “MATLAB Instrument Drivers” to see the list of supported IVI drivers:

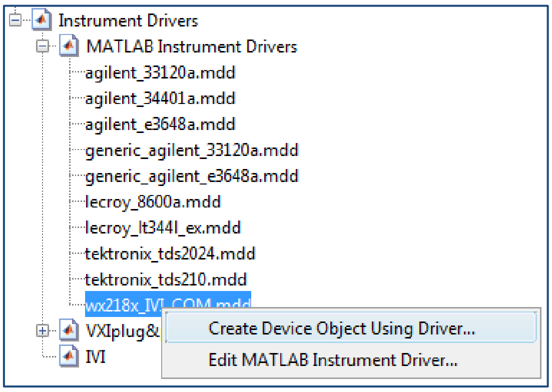


Instrument Drivers
MATLAB Instrument Drivers
agilent_33120a.mdd
agilent_34401a.mdd
agilent_e3648a.mdd
generic_agilent_33120a.mdd
generic_agilent_e3648a.mdd
lecroy_8600a.mdd
lecroy_lt344l_ex.mdd
tektronix_tds2024.mdd
tektronix_tds210.mdd
wx218x_IVI_COM.mdd

NOTE

If the Tabor ‘wx218x_IVI_COM.mdd’ file isn’t there, please refer to the instructions in “How to Connect to Tabor Using Instrument Control Toolbox”” tutorial, regarding how to copy & paste this ‘*.mdd’ file to MATLAB’s ‘Drivers’ folder.

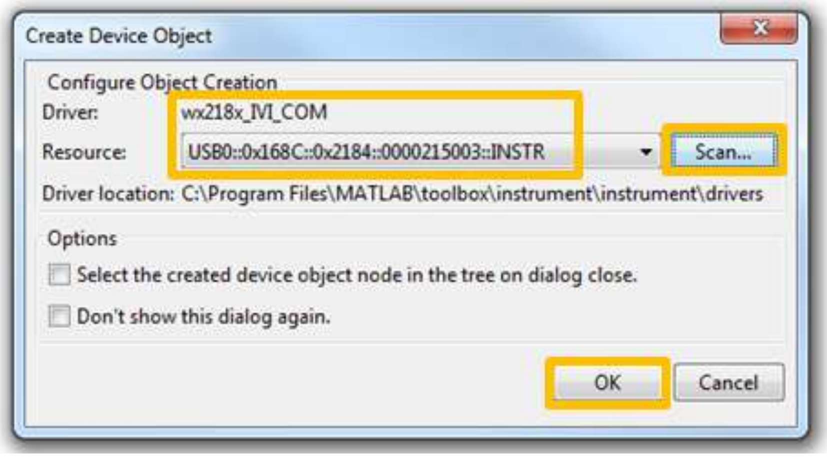
6. Right click on “wx218x_IVI_COM.mdd”. Choose “Create Device Object Using Driver…”



Instrument Drivers
MATLAB Instrument Drivers
agilent_33120a.mdd
agilent_34401a.mdd
agilent_e3648a.mdd
generic_agilent_33120a.mdd
generic_agilent_e3648a.mdd
lecroy_8600a.mdd
lecroy_lt344l_ex.mdd
tektronix_tds2024.mdd
tektronix_tds210.mdd
wx218x_IVI_COM.mdd
VXIplug&
IVI

Create Device Object Using Driver...
Edit MATLAB Instrument Driver...

7. A new window pops out. Click on ‘Scan’. Make sure the address and driver are the correct ones for this specific AWG. Click on ‘OK’ to create a device object:



Create Device Object

Configure Object Creation

Driver: wx218x_IVI_COM

Resource: USB0::0x168C::0x2184::0000215003::INSTR

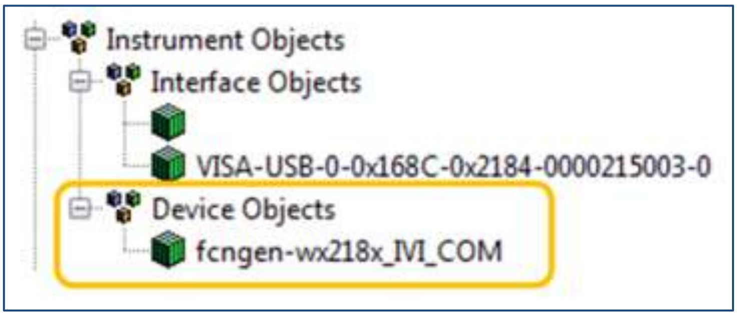
Driver location: C:\Program Files\MATLAB\toolbox\instrument\instrument\drivers

Options

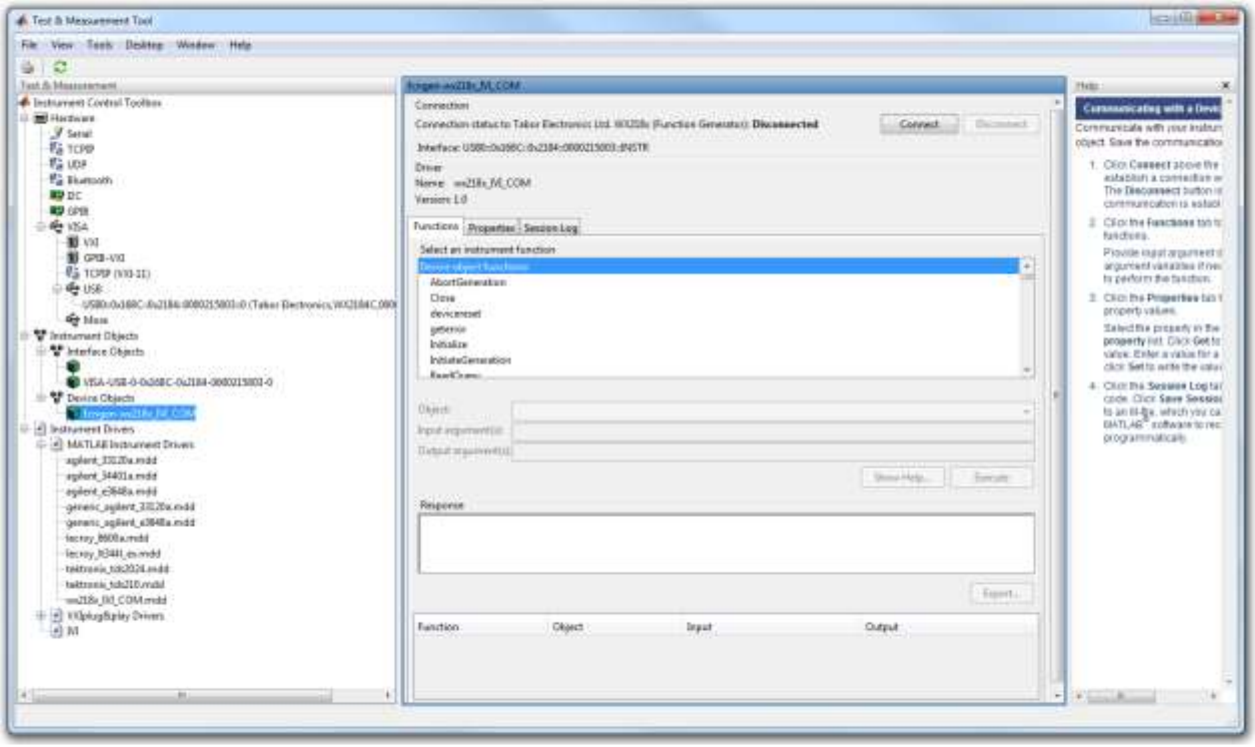
□ Select the created device object node in the tree on dialog close.

□ Don't show this dialog again.

A new device object should appear:



8. Click on the ‘fcngen-wx218x_IVI_COM’ and a new dialog box will pop up:



fcngen-wx218x_IVI_COM

Connection

Connection status to Tabor Electronics Ltd. WX218x (Function Generator): **Disconnected**

Interface: USB0::0x168C::0x2184::0000215003::INSTR

Driver

Name: wx218x_IVI_COM

Version: 1.0

Select an instrument function

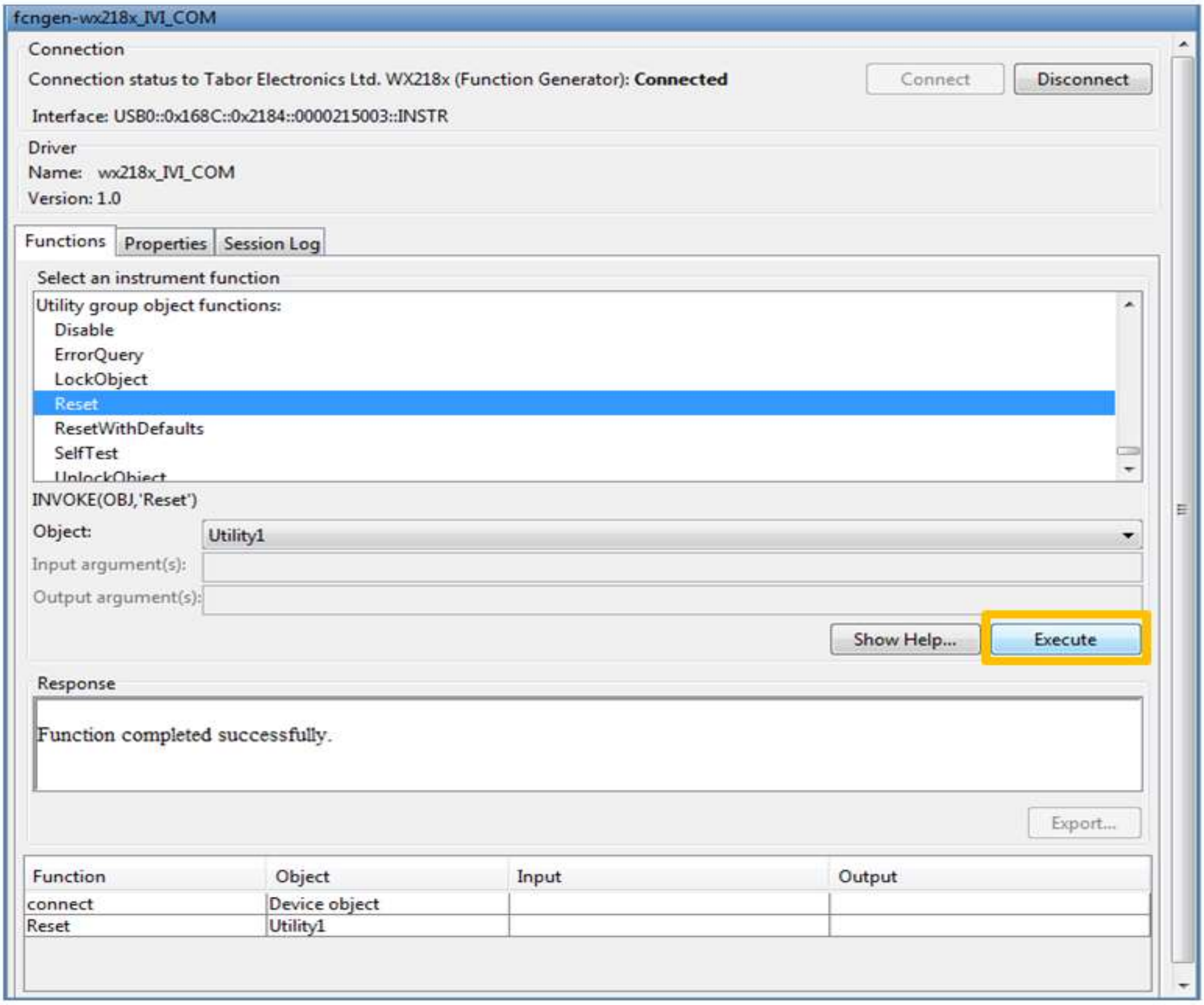
On the Device Object dialog box you will see ‘Functions’, ‘Properties’ and ‘Session Log’.

- ***Functions*** - Allows you to communicate with the device using its driver functions.
- ***Properties*** – Allows you to set parameters and modes regarding the device under control.
- ***Session Log*** – Allows you to save the last session as MATLAB script.

9. Here is a short basic example, to set a 50MHz 2Vp-p square waveform in standard mode, just to get a feel of how to communicate using the IVI driver.

   a. Press on the ‘Connect’ button.

b. After successfully connecting to Tabor, execute the 'Reset' function under 'Utility group object functions' as shown below. A new action line should appear.



Connection status to Tabor Electronics Ltd. WX218x (Function Generator): **Connected**

Interface: USB0::0x168C::0x2184::0000215003::INSTR

Name: wx218x_IVI_COM

Version: 1.0

INVOKE(OBJ,'Reset')

Object: Utility1

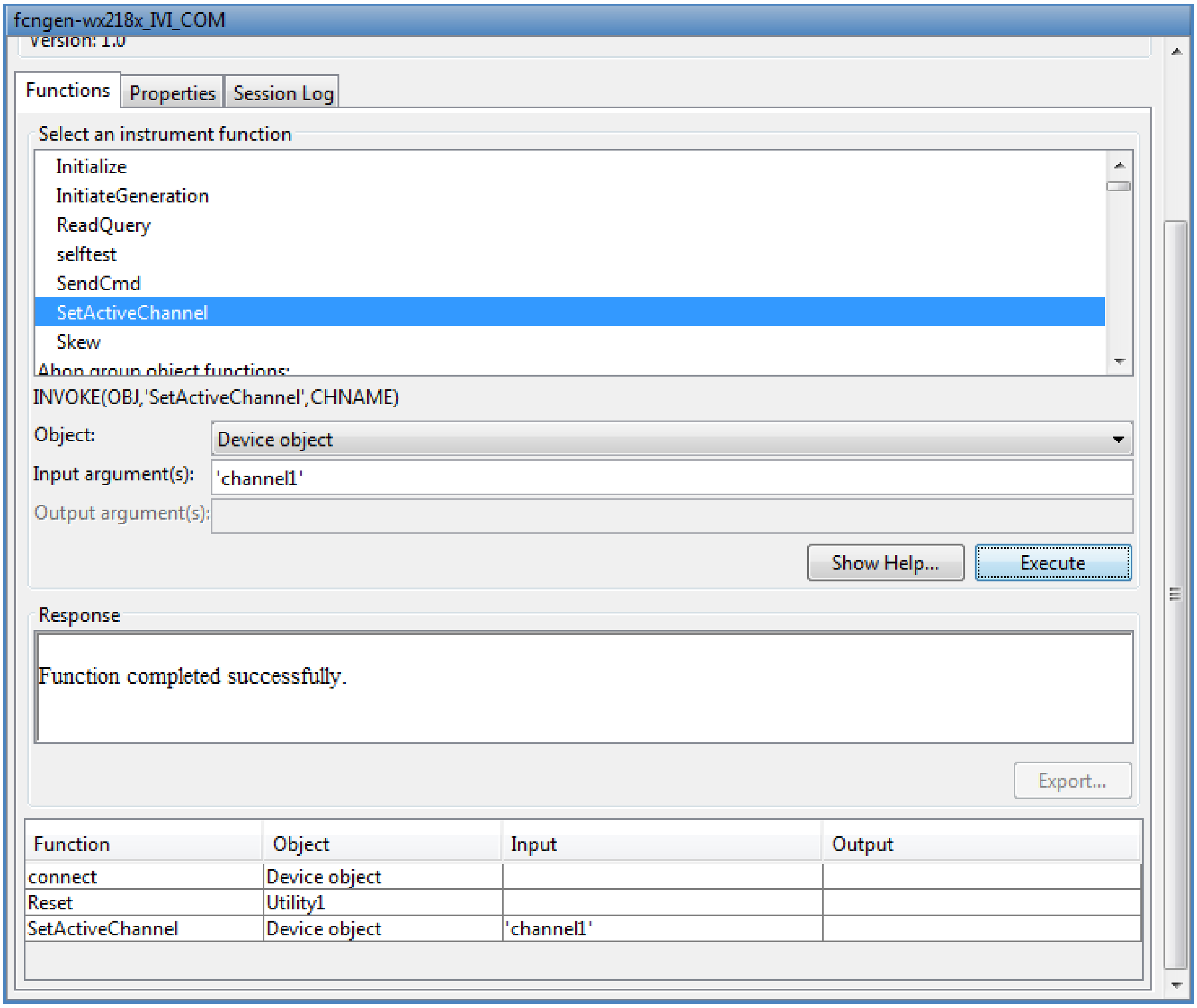
Input argument(s):

Output argument(s):

Function completed successfully.

Function	Object	Input	Output
connect	Device object		
Reset	Utility1		

c. Execute the ‘SetActiveChannel’ function under ‘Device object functions’. As input type: ‘channel1’ and click on ‘Execute’.



INVOKE(OBJ,'SetActiveChannel',CHNAME)

Object: Device object

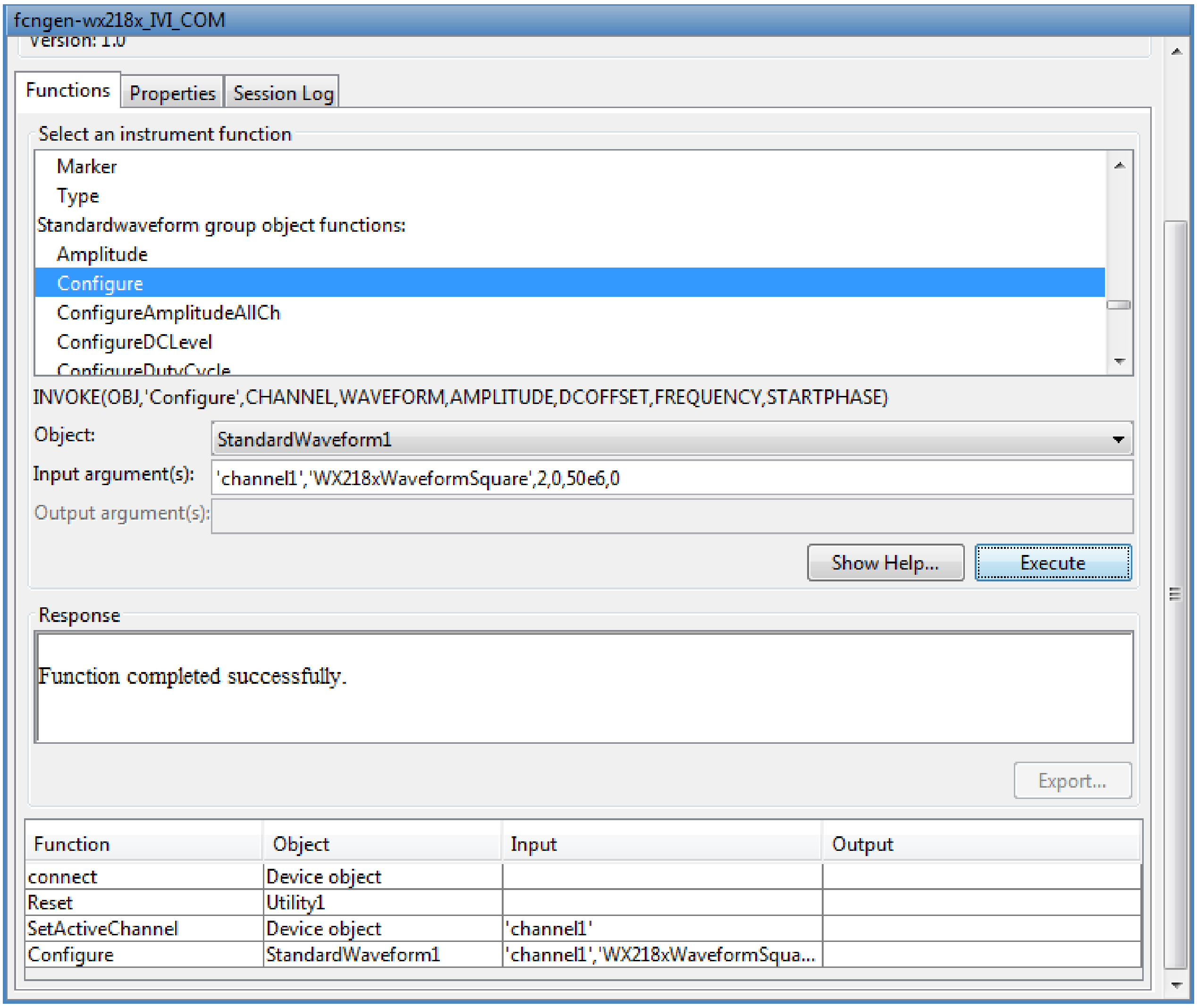
Input argument(s): 'channel1'

Output argument(s):

Function completed successfully.

Function	Object	Input	Output
connect	Device object		
Reset	Utility1		
SetActiveChannel	Device object	'channel1'	

d. Execute the 'configure' function under 'StandardWaveform group object functions'. As inputs type: 'channel1', 'WX218xWaveformSquare', 2, 0, 50e6, 0. Then click on 'Execute' to set the waveform parameters.



INVOKE(OBJ,'Configure',CHANNEL,WAVEFORM,AMPLITUDE,DCOFFSET,FREQUENCY,STARTPHASE)

Object: StandardWaveform1

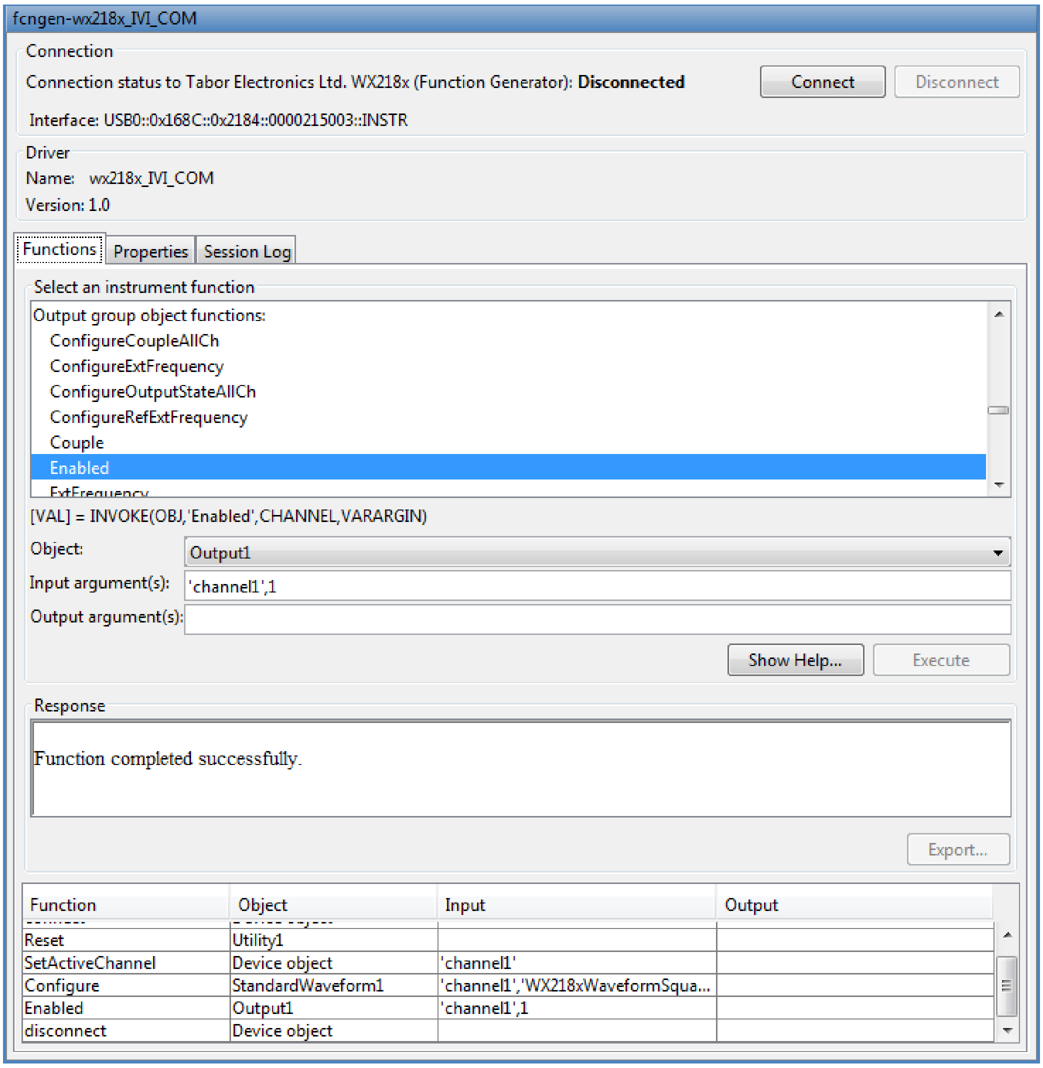
Input argument(s): 'channel1','WX218xWaveformSquare',2,0,50e6,0

Output argument(s):

Function completed successfully.

Function	Object	Input	Output
connect	Device object		
Reset	Utility1		
SetActiveChannel	Device object	'channel1'	
Configure	StandardWaveform1	'channel1','WX218xWaveformSqua...	

e. Finally execute the ‘Enable’ function under ‘Output group object functions’. As input, type: ‘channel1’, 1 to turn CH1 on (0 to turn CH1 off). After doing so, press on the ‘Disconnect’ button.



Connection

Connection status to Tabor Electronics Ltd. WX218x (Function Generator): **Disconnected**

Interface: USB0::0x168C::0x2184::0000215003::INSTR

Driver

Name: wx218x_IVI_COM

Version: 1.0

Select an instrument function

Output group object functions:
ConfigureCoupleAllCh
ConfigureExtFrequency
ConfigureOutputStateAllCh
ConfigureRefExtFrequency
Couple
Enabled
ExtFrequency

[VAL] = INVOKE(OBJ,'Enabled',CHANNEL,VARARGIN)

Object: Output1

Input argument(s): 'channel1',1

Output argument(s):

Response

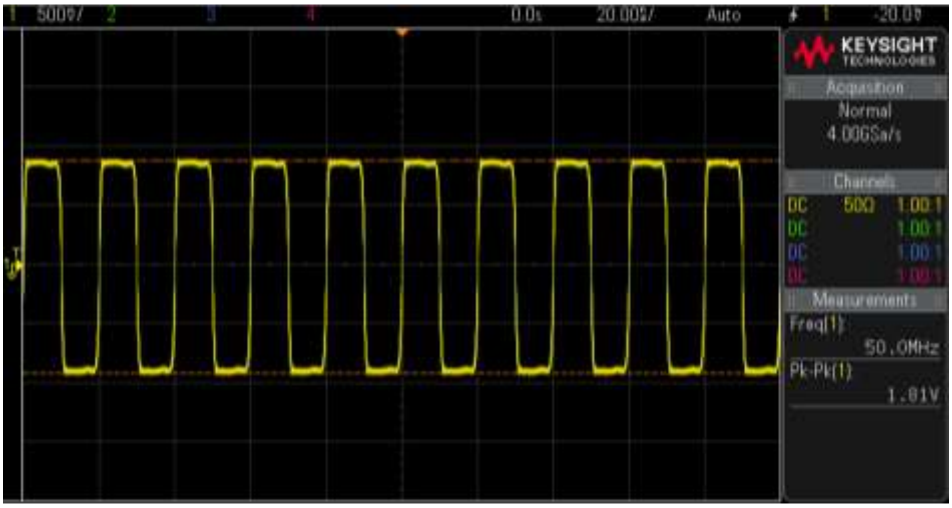
Function completed successfully.

Function	Object	Input	Output
Reset	Utility1		
SetActiveChannel	Device object	'channel1'	
Configure	StandardWaveform1	'channel1','WX218xWaveformSqua...	
Enabled	Output1	'channel1',1	
disconnect	Device object		

As can be seen above, each time you press the ‘Execute’ button, the function you executed is also saved as an action line that can be used later on as part of a MATLAB script for automation purposes.

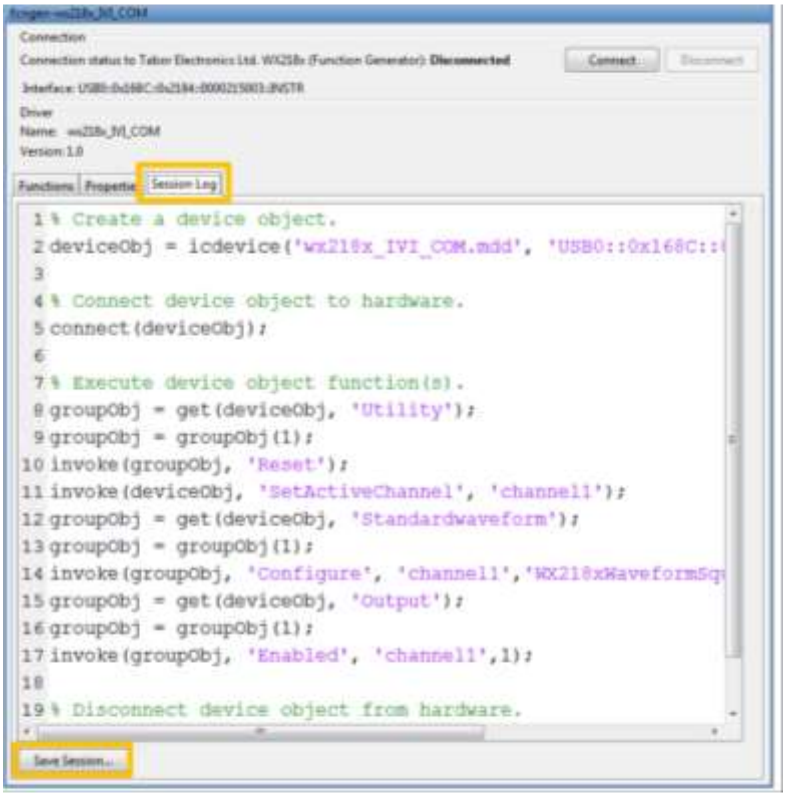
Each time a function is successfully executed, you can go to the ‘Session Log’ and see its syntax. You could also copy and paste it to a different MATLAB script . This is a great helping tool for those who are new to the IVI driver functions used with Tabor AWGs.

10. As can be seen on scope, a 50MHz 2Vp-p square waveform was created:



The outputted 50MHz 2Vp-p square waveform.

11. In order to save this process as a MATLAB code, go to ‘Session log’ where you will see that all the actions you have made were automatically translated into MATLAB code. Press the ‘Save Session’ button to save as MATLAB script.



Connection

Connection status to Tabor Electronics Ltd. WX218x (Function Generator): **Disconnected**

Interface: USB0::0x168C::0x2184::0000215003::INSTR

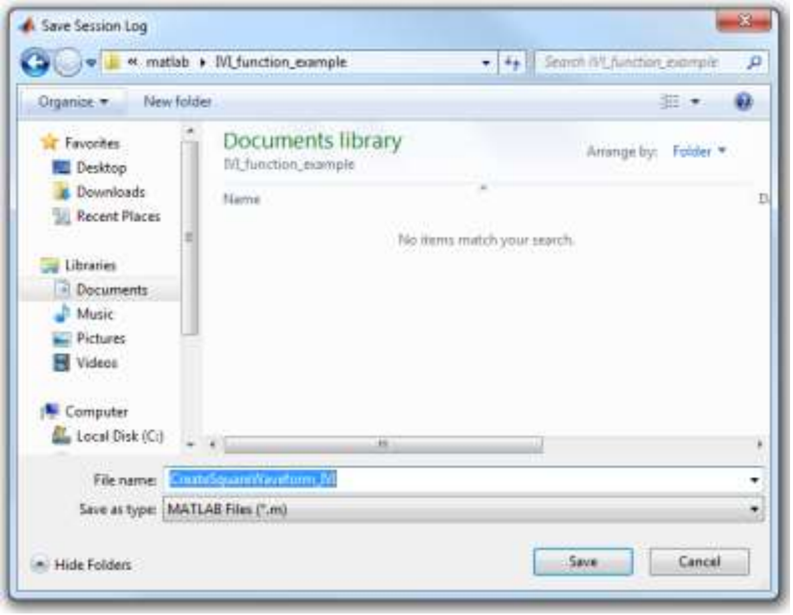
Driver

Name: wx218x_IVI_COM

Version: 1.0

```
% Create a device object.
deviceObj = icdevice('wx218x_IVI_COM.mdd', 'USB0::0x168C::(

% Connect device object to hardware.
connect(deviceObj);

% Execute device object function(s).
groupObj = get(deviceObj, 'Utility');
groupObj = groupObj(1);
invoke(groupObj, 'Reset');
invoke(deviceObj, 'SetActiveChannel', 'channel1');
groupObj = get(deviceObj, 'Standardwaveform');
groupObj = groupObj(1);
invoke(groupObj, 'Configure', 'channel1','WX218xWaveformSq
groupObj = get(deviceObj, 'Output');
groupObj = groupObj(1);
invoke(groupObj, 'Enabled', 'channel1',1);

% Disconnect device object from hardware.
```

And save your session as MATLAB script:



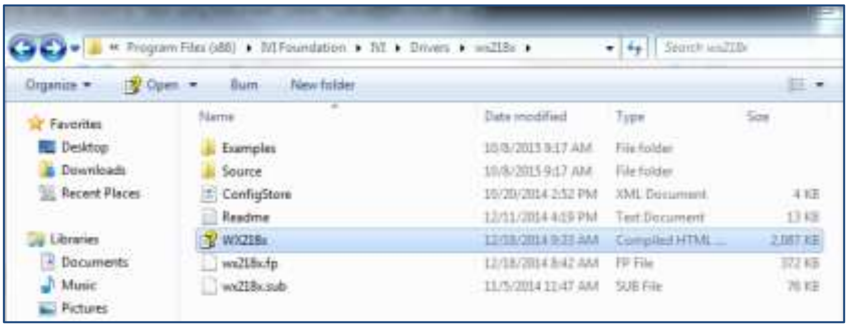
File name: CreateSquareWaveform_IVI

Save as type: MATLAB Files (*.m)

12. For more information regarding the IVI driver functions:

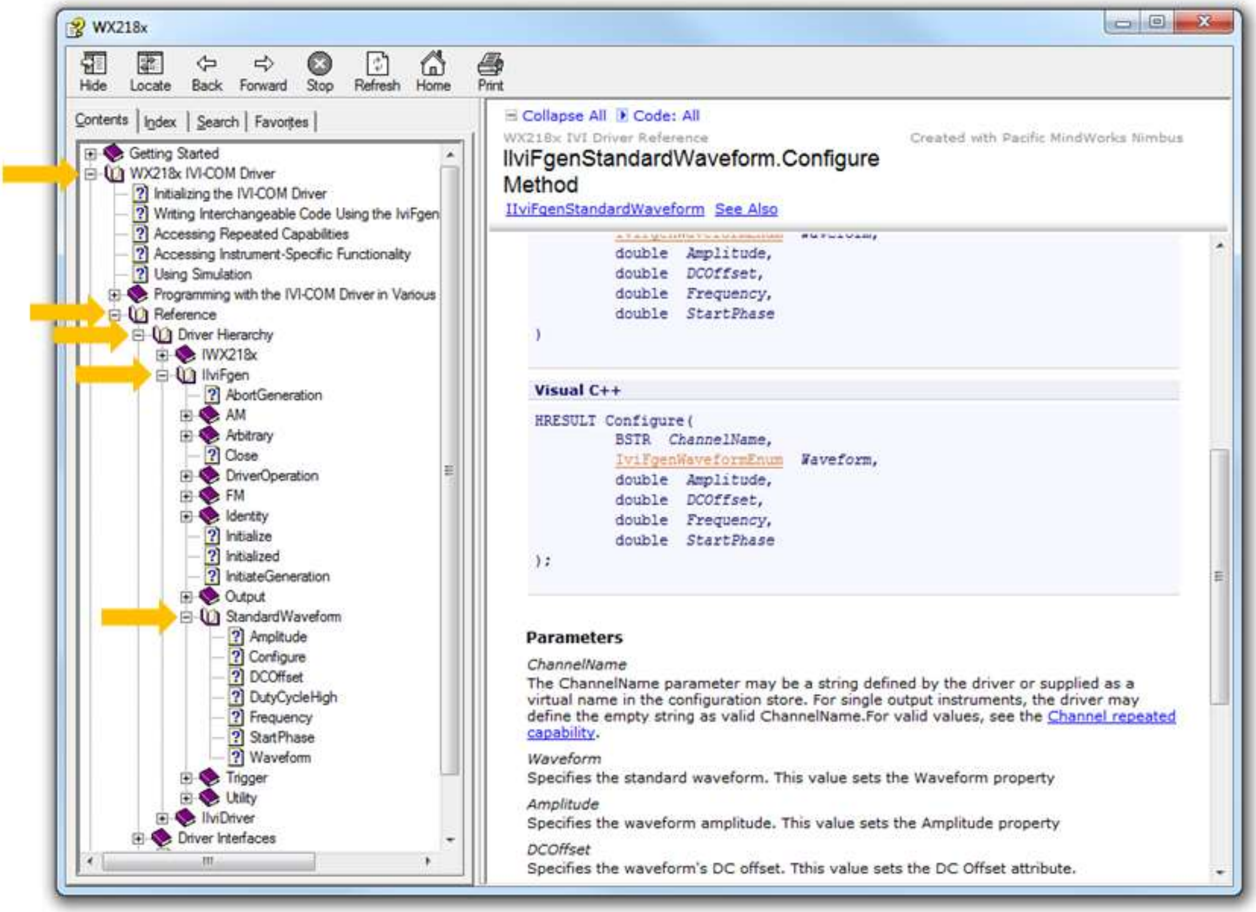
    a. Go to: *C:\Program Files (x86)\IVI Foundation\IVI\Drivers\wx218x*

        i. Open the “WX218x” file:



Name	Date modified	Type	Size
Examples	10/8/2015 9:17 AM	File folder	
Source	10/8/2015 9:17 AM	File folder	
ConfigStore	10/20/2014 2:52 PM	XML Document	4 KB
Readme	12/11/2014 4:19 PM	Text Document	13 KB
WX218x	12/18/2014 9:33 AM	Compiled HTML ...	2,087 KB
wx218x.fp	12/18/2014 8:42 AM	FP File	372 KB
wx218x.sub	11/5/2014 11:47 AM	SUB File	76 KB

ii. Follow the path as shown below:



WX218x IVI Driver Reference

Created with Pacific MindWorks Nimbus

IIviFgenStandardWaveform.Configure Method

IIviFgenStandardWaveform See Also

```
        double  Amplitude,
        double  DCOffset,
        double  Frequency,
        double  StartPhase
)
```

Visual C++

```
HRESULT Configure(
        BSTR  ChannelName,
        IviFgenWaveformEnum  Waveform,
        double  Amplitude,
        double  DCOffset,
        double  Frequency,
        double  StartPhase
);
```

Parameters

ChannelName
The ChannelName parameter may be a string defined by the driver or supplied as a virtual name in the configuration store. For single output instruments, the driver may define the empty string as valid ChannelName.For valid values, see the Channel repeated capability.

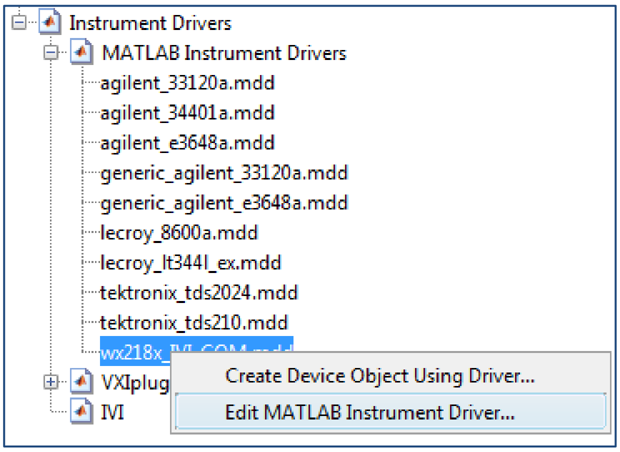
Waveform
Specifies the standard waveform. This value sets the Waveform property

Amplitude
Specifies the waveform amplitude. This value sets the Amplitude property

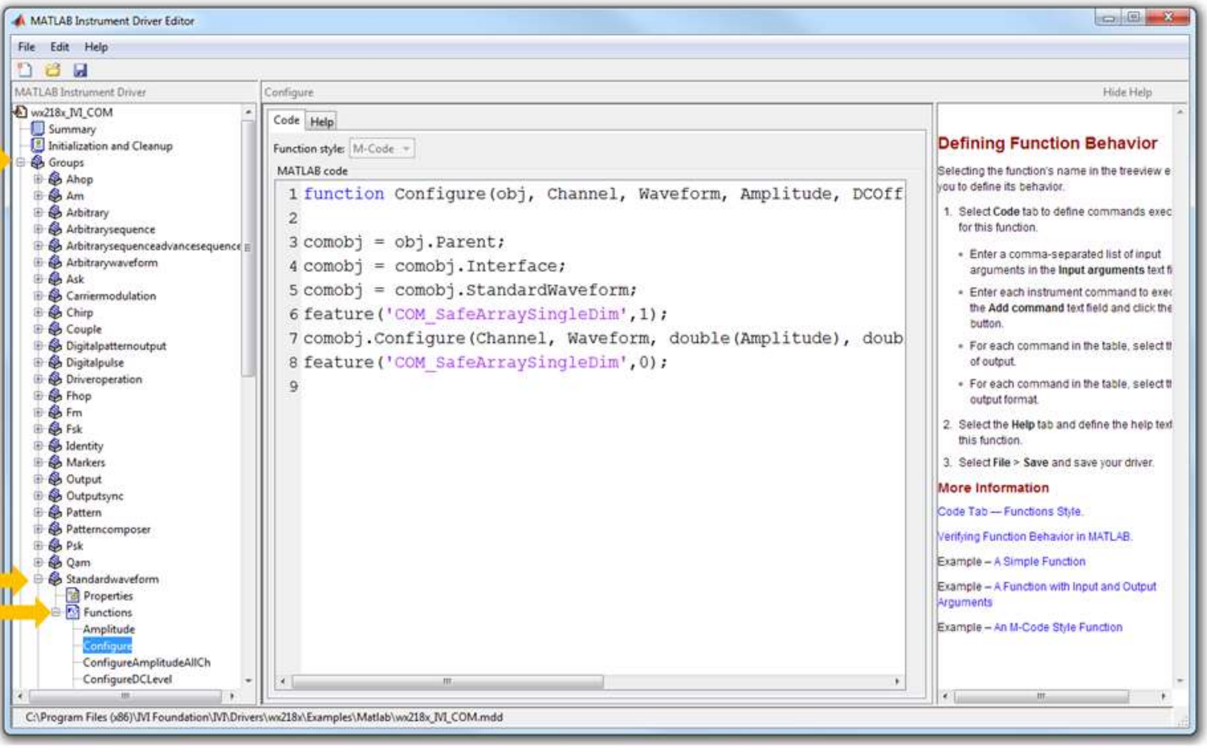
DCOffset
Specifies the waveform's DC offset. Tthis value sets the DC Offset attribute.

b. In MATLAB's Test & Measurement Tool, you can also have a look at the functions syntax by:

i. right clicking on "wx218x_IVI_COM.mdd"and choosing "Edit MATLAB Instrument Driver".



ii. To see the functions syntax extend:
‘Groups’->chose group of interest->’Functions’->chose function of interest



```
function Configure(obj, Channel, Waveform, Amplitude, DCOff

comobj = obj.Parent;
comobj = comobj.Interface;
comobj = comobj.StandardWaveform;
feature('COM_SafeArraySingleDim',1);
comobj.Configure(Channel, Waveform, double(Amplitude), doub
feature('COM_SafeArraySingleDim',0);
```

Defining Function Behavior

Selecting the function's name in the treeview e
you to define its behavior.

1. Select **Code** tab to define commands exec
for this function.
    - Enter a comma-separated list of input arguments in the **Input arguments** text fi
    - Enter each instrument command to exe the **Add command** text field and click the button.
    - For each command in the table, select t of output.
    - For each command in the table, select t output format.
2. Select the **Help** tab and define the help tex this function.
3. Select **File** > **Save** and save your driver.

More Information

Code Tab — Functions Style.

Verifying Function Behavior in MATLAB.

Example – A Simple Function

Example – A Function with Input and Output Arguments

Example – An M-Code Style Function

C:\Program Files (x86)\IVI Foundation\IVI\Drivers\wx218x\Examples\Matlab\wx218x_IVI_COM.mdd

In the next four tutorials of the series “How to Control Tabor AWGs with MATLAB”, we will show four practical MATLAB coding examples.

For More Information

To learn more about how to use MATLAB with Tabor instruments, visit our website Support & Tutorials zone. For more of Tabor’s solutions or to schedule a demo, please contact your local Tabor representative or email your request to info@tabor.co.il. More information can be found at our website at www.taborelec.com