



Using MATLAB[®] to Configure and Control Tabor's Wonder Wave Series with IVI Drivers

Application Note

Alexey Farovich – Senior Software Engineer

Tabor Electronics Ltd
PO Box 404, Tel Hanan, Israel
support@taborelec.com

Trademarks

MATLAB[®] is a trademark of The MathWorks, Inc.
NI-VISA is a trademark of National Instrument Corp.
NI-IVI is a trademark of National Instrument Corp.

March 5, 2007

TABOR ELECTRONICS Ltd.
www.taborelec.com

Table of Contents

1.0	Introduction	2
2.0	What is IVI?	3
3.0	Installing the requirement software	4
3.1	Installing the ww257x IVI driver	4
4.0	Using the ww257x IVI driver in MATLAB	6
4.1	Example Script	7
5.0	Literature	10

1.0 Introduction

MATLAB® is a well known software environment for generating waveforms for instruments, acquiring and analyzing measurements, and building test systems. It can control and configure the Tabor Wonder Wave Series of arbitrary waveform generators using MATLAB's [Instrument Control Toolbox](#). Instrument Control Toolbox supports IVI instrument drivers. This guide will show you how to configure an IVI instrument driver, created by Tabor, to enable MATLAB to communicate with Tabor's Wonder Wave Series.

This application note is common to the entire Wonder Wave Series. Model WW2572 was selected as an example. For all other models you will have similar file names.

2.0 What is IVI?

The Interchangeable Virtual Instrument (IVI) specification provides a standard for instrument drivers that address the performance issues demanded by test system developers and provides a robust framework for interchanging instruments.

IVI was developed to address problems test software developers faced when writing code to programmatically communicate with their instruments. Specifically, each instrument in their system required a different set of commands to perform a specific action, with no simple, standardized communication protocol in place to program their instruments. Companies wasted valuable time and money training their developers to write software for instruments. Moreover, when instruments were upgraded, developers could not reuse the software; they often had to learn a new protocol or a different set of commands.

The IVI specification subdivides instruments into classes such as digital multimeters or oscilloscopes. They then create a specification for that class based on the most common features and functionality of the most popular instruments within that class. The IVI Foundation, the standards body overseeing the development of these specs, has already defined specifications for eight instrument classes. The table below provides a list of all these classes.

IVI Instrument Classes	
Digital Multimeter	Switch
Oscilloscope	Power Meter
DC Power Supply	Spectrum Analyzer
Arbitrary Waveform/Function Generator	RF Signal Generator

If you are developing test systems or software that communicates with instruments, IVI can provide you with many benefits. IVI is not limited to one instrument manufacturer or one type of communication bus (e.g., GPIB, VXI, or computer-based) and therefore does not limit your choice of instrument.

The IVI Foundation provides a comprehensive online list of IVI drivers at www.ivifoundation.org/Drivers.htm. This list provides information on more than 100 IVI drivers available for download from various instrument and software vendors' Web sites.

3.0 Installing the required software

The ww257x IVI driver uses the NI-VISA for communication with the instrument and the NI-IVI Compliance Package, therefore you need to install them first. The installing sequence is important: first you need to install the NI-VISA and afterwards the NI-IVI Compliance Package.

NOTE: All the installations afore mentioned can be found both in the Tabor web site and on the CD which is each of our instruments. You can also download the latest version directly from the NI web site.

Use the following Links:

- www.ni.com/visa (download the full version of NI-VISA)
- www.ni.com/ivi (download the NI-IVI Compliance Package)

3.1 Installing the ww257x IVI driver

As was mentioned above, the installation program available on a CD that is supplied with the instrument. Insert the CD to your CD drive and follow the instructions in the installer. You may then verify the integrity of the installation of the ww257x IVI driver in the NI-VISA or the MATLAB environment:

- **NI-VISA environment**
To verify if the driver is installed properly, use the Measurement & Automation Explorer (MAX). To run the MAX, select **National Instruments >> Measurement & Automation** from the **Start** menu as seen below in Figure 1.

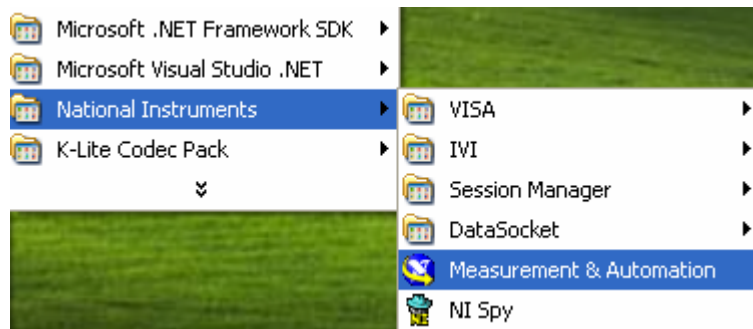


Figure 1 – Finding the Measurement & Automation Explorer.

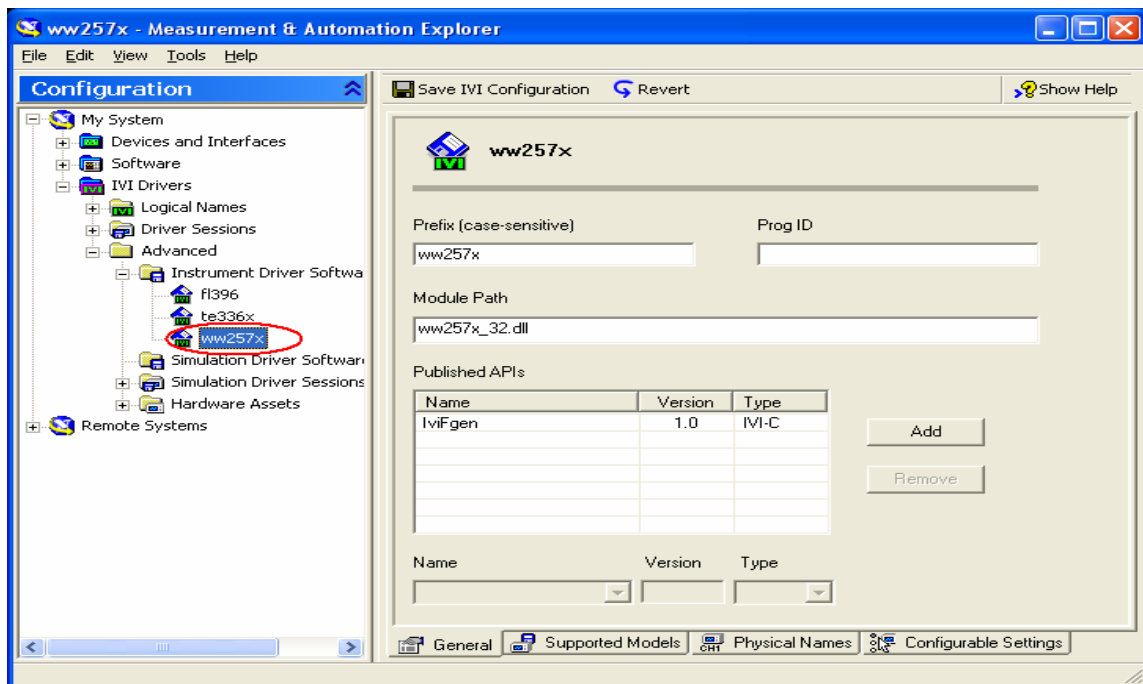


Figure 2 - Viewing the ww257x IVI driver in MAX.

- **MATLAB environment**

To verify if the driver is installed properly, launch MATLAB and use the **instrhwinfo** command. The **instrhwinfo** command takes an optional argument, which can be a type of instrument driver. For example, for IVI drivers the argument is

```
driverinfo = instrhwinfo('ivi')
```

```
driverinfo =
```

```

    LogicalNames: {}
    ProgramIDs: {}
    Modules: {'f1396' 'te336x' 'ww257x'}
    ConfigurationServerVersion: '1.3.2.4'
    MasterConfigurationStore: [1x51 char]
    IVIRootPath: 'C:\Program Files\IVI\'

```

The output of the command is a structure with fields. The **IVIRootPath** field identifies the main IVI root install path. The **Modules** field lists all of the IVI drivers that are installed in the root path.

NOTE: MATLAB's Instrument Control Toolbox now provides a graphical tool, called Test & Measurement Tool, which allows you to detect, control, configure, and exchange data with instruments without writing code. Execute the **tmtool** command to launch this tool. You may use this tool to verify that an instrument driver is installed or to configure an instrument. However, using this tool is beyond the scope of this application note.

4.0 Using the ww257x IVI driver in MATLAB

All MATLAB instrument drivers, regardless of whether they are stand-alone or wrap an underlying manufacturer's driver, are used to instantiate an **icdevice** object in MATLAB. This object is assigned to a MATLAB variable that has a set of properties and methods for communicating with the instrument.

Some properties and methods are common across all objects independent of the driver type and instrument model, for example, the driver name and connect/disconnect methods. In addition, the object will have properties and methods that are unique to the instrument based on the information in the driver. In the following example we will create an **icdevice** object and connect to the instrument, which is connected to the computer through a GPIB interface. The GPIB primary address is 6.

```
deviceObj = icdevice('tabor_ww257x.mdd', 'GPIB0::6::INSTR');  
connect(deviceObj);
```

You can view some basic information about the **icdevice** object and the associated driver and instrument by looking at the **icdevice** object's display.

```
display(deviceObj)
```

```
Instrument Device Object Using Driver : ww257x
```

```
Instrument Information
```

```
  Type:          Function Generator  
  Manufacturer:  Tabor Electronics Ltd  
  Model:        TABOR WW257X Waveform Generator
```

```
Driver Information
```

```
  DriverType:    MATLAB IVI  
  DriverName:    ww257x  
  DriverVersion: 1.0
```

```
Communication State
```

```
  Status:        open
```

Most of the instrument functionality is typically contained below the root level of the object in groups related to specific functionality. These groups are properties of the object and have their own properties and functions. The following code examines the methods of the **Configuration** group for this driver.

```
groupObj = get(deviceObj, 'Configuration');  
methods(groupObj)
```

```
Driver specific methods for class icconfiguration:  
configurefilter      configureoperationmode  configureoutputenabled  
configureoutputimpedance  configureoutputmode    configurerefclocksource  
configuresyncsignal
```

Often, the default method names are somewhat vague or condensed. The **instrhelp** command, called on the object with a property or method name, returns help information that is contained in the driver file.

```
instrhelp(deviceObj.Configuration, 'configureoutputenabled')
```

This function configures whether the signal the function generator produces appears at the output channel you specify. The driver sets the WW257X_ATTR_OUTPUT_ENABLED attribute to the value you pass in the Output Enabled parameter.

Executing a method on the object is very similar to using the COM interface in MATLAB. The invoke method is used here. For example:

```
invoke(groupObj, 'configureoutputenabled', 'CHAN_A', 1);
```

This command enables the output of channel A.

NOTE: You can find additional information on group and functions in the ww257x IVI driver online help, use the following shortcut (**Start>>Programs>>Tabor Electronics>>WW-257X**). In MATLAB, you can use the **midedit** command to view functions of the tabor_ww257x.mdd driver wrapper. For example:

```
midedit 'tabor_ww257x.mdd'
```

You can find additional information on group objects, getting and setting properties, and invoking methods in the toolbox [documentation](#).

4.1 Example Script

The following script demonstrates how to create a sequence with the Model ww2571/2A waveform generator at IP address 192.168.0.240, using the ww257x IVI driver. Before running this script make sure that the ww2571/2A is available via your network; instruction below helps you to do this :

- Power 'On' the instrument.
- Check that the network cable is connected to the instrument.
- To verify the IP address that was assigned to the instrument press the **TOP** button, then press the **Utility** button and scroll down to the **Remote Setup** command. Press **Enter** as shown in Figure 3.

TIP: Learn how to select an active interface and how to set up interface parameters in Chapter 2 of the operating Manual, entitled **Configuring the Instrument**.

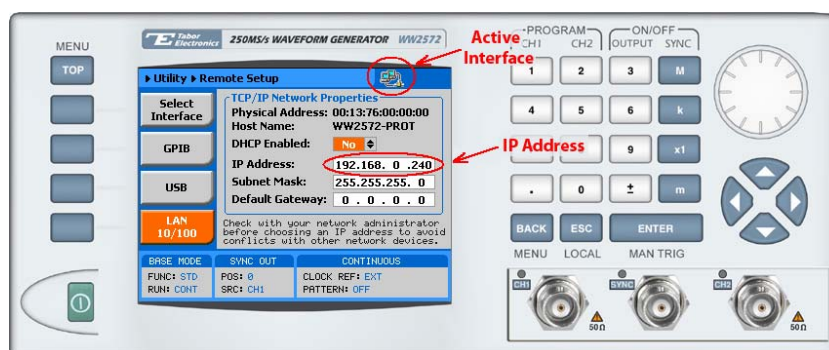


Figure 3 - Viewing the remote setup in ww2571/2A

- To verify the communication between the instrument and your PC, use the **ping** utility. To

run the **ping** utility, select the **Run...** command from the **Start** menu and type **CMD** in the **Open:** field, as shown in Figure 4. Click on OK and expect the MS-DOS window to appear.

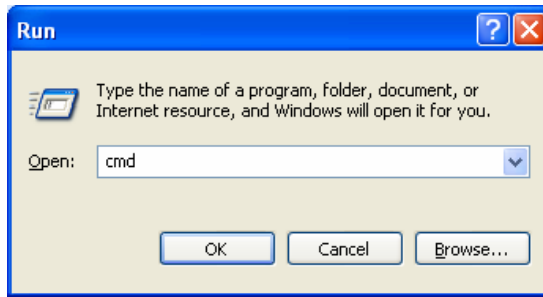


Figure 4 – Run the **MS-DOS Prompt**

Type **ping 192.168.0.240** at the command line, as shown below in Figure 5.

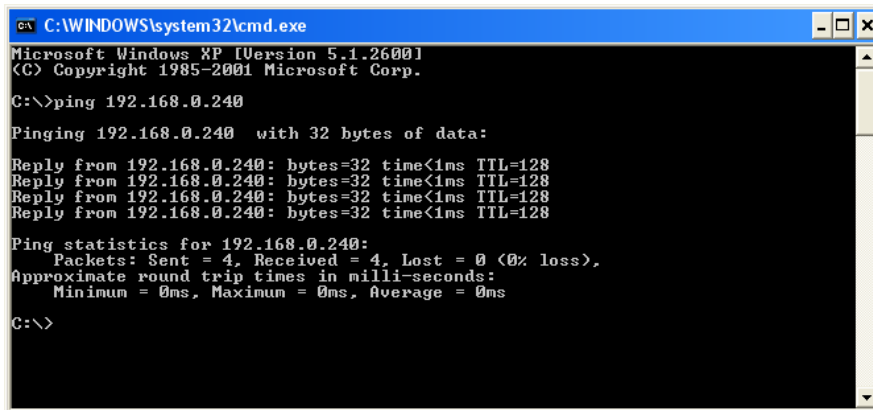


Figure 5 – The Ping utility

NOTE: For USB interface you need to make sure that it has been activated on the instrument. If you want to run this example, use the following descriptor '**ASRL<Port Number>::INSTR**'. To detect the USB port number that the instrument obtains from windows, you need to run the Windows **Device Manager** and select item **Ports (COM & LPT)** from the devices tree as shown in Figure 6:

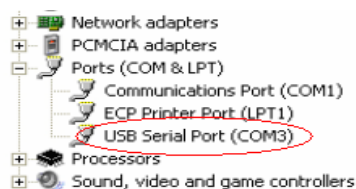


Figure 6 – The Device Manager

For figure 6 the descriptor should be '**ASRL3::INSTR**'.

Below is an example of a MATLAB script used to create a complex waveform to be sent to the Tabor arbitrary waveform generator:

```

% Open connection to instrument with IP address
dev = icdevice('tabor_ww257x.mdd', 'TCPIP::192.168.0.240::23::SOCKET')

connect(dev);

% Reset device
groupCnf = get(dev, 'Utility');
invoke(groupCnf, 'reset')

% Set the sample clock frequency
groupArb = get(dev, 'Arbitraryoutput');
invoke(groupArb, 'configuresamplerate', 30E6)

% Create three segments in channel 1 and download waves into them
groupArb = get(dev, 'Arbitrarywaveforminformation');
wfmHandle1 = invoke(groupArb, 'loadarbwfmfromfile', 'CHAN_A', 'sine_1kpts.wav');
wfmHandle2 = invoke(groupArb, 'loadarbwfmfromfile', 'CHAN_A', 'dc_256pts.wav');
wfmHandle3 = invoke(groupArb, 'loadarbwfmfromfile', 'CHAN_A', 'am_ramp_2kpts.wav');

wfmHandles = [wfmHandle1, wfmHandle2, wfmHandle3];

% Create a sequence in the active channel, currently the active
% channel is 1 because the 'loadarbwfmfromfile' function used with 'CHAN_A'
% parameter
%
% Sequence Description:
% Step #           Segment #           Repeats Count
% [ 1 ]           [ Segment 1 ( sine_1kpts.wav ) ]           [ 2 ]
% [ 2 ]           [ Segment 2 ( dc_256pts.wav ) ]           [ 3 ]
% [ 3 ]           [ Segment 3 ( am_ramp_2kpts.wav ) ]           [ 1 ]
%
groupArb = get(dev, 'Arbitrarysequenceinformation');
seqHandle = invoke(groupArb, 'createarbsequence', 3, wfmHandles, [2,3,1]);

% Create the sequence in the active channel, currently the active
groupCnf = get(dev, 'Configuration');

% Output Modes:
% 0 - Standard
% 1 - Arbitrary
% 2 - Sequence
% 3 - Modulated
invoke(groupCnf, 'configureoutputmode', 2)

% Output enable
invoke(groupCnf, 'configureoutputenabled', 'CHAN_A', 1)

% Output SYNC signal enable
%
% SYNC Types:
% 0 - BIT
% 1 - LCOM
sync_type = 1;
invoke(groupCnf, 'configuresyncsignal', wfmHandle1, sync_type, 1, 0)

% Close the connection with the instrument
disconnect(dev);
delete(dev);

```

The result of this MATLAB example can be observed if you connect the instrument to an oscilloscope, similar to the waveform shown in Figure 7:

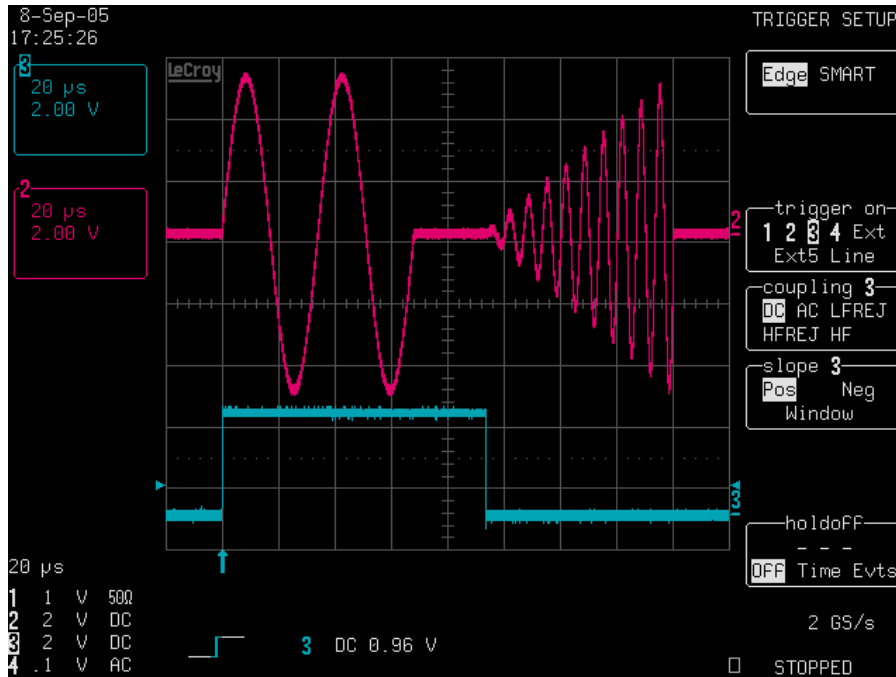


Figure 7 – Waveform Sequence Example

5.0 Literature

For more information about programming as well as for documentation of the remote control commands, refer to the instrument's user manual.

For additional information on using MATLAB or its Instrument Control Toolbox, refer to the on-line documentation available on The MathWorks website:

MATLAB: www.mathworks.com/products/matlab

Instrument Control Toolbox: www.mathworks.com/products/instrument