

**Data Acquisition Interface Module (Model 9062-XX)**

**USER GUIDE**

**Explanation and Utilization of the API**

**August 2004**

## Table of Contents

1	<b>Introduction</b> .....	-4-
2	<b>API Installation</b> .....	-5-
	2.1 Requirements .....	-5-
	2.2 Installation .....	-5-
	2.3 Opening the VI's .....	-5-
3.	<b>Description of the Library of Virtual Instruments (VI's)</b> .....	-7-
	3.1 Demonstration VI's .....	-7-
	3.1.1 9062USB VI Tree .....	-7-
	3.1.2 Initialize .....	-7-
	3.1.3 Example .....	-7-
	3.1.4 GetStarted120V → 9062-X0 Module GetStarted220-240V → 9062-X5 Module .....	-7-
	3.2 Description of the VI's of the Data Acquisition Interface Module .....	-8-
	3.2.1 InitVariable .....	-8-
	3.2.2 Exit .....	-8-
	3.2.3 DetectModule .....	-8-
	3.2.4 SetTimeOut .....	-9-
	3.2.5 SetFrequency .....	-9-
	3.2.6 GetFrequency .....	-9-
	3.2.7 InitRelay (WriteDigital.vi) .....	-10-
	3.2.8 RelayState (ReadGlobalRangeUSB.vi) .....	-10-
	3.2.9 AcquisitionUSB .....	-11-
	3.2.10 TransfertUSB .....	-12-
	3.2.11 WriteAnalog (2) .....	-13-
	3.2.12 TestTrig .....	-14-
	3.2.13 ResetDevice .....	-15-
4.	<b>Demonstration of the Features of the Application Programming Interface (API)</b> .....	-16-
	4.1 Required Material .....	-16-
	4.2 Circuit Setup .....	-16-
	4.3 Execution of the Example VI .....	-18-
	4.4 Exploring the Sections of the Control Panel .....	-20-
	4.4.1 Module Detection .....	-20-
	4.4.2 Set Frequency .....	-20-
	4.4.3 Set Trigger .....	-21-
	4.4.4 Set Input Range .....	-21-
	4.4.5 USB Acquisition .....	-22-
	4.4.6 USB Transfer .....	-22-
	4.4.7 Analog Outputs .....	-23-
5.	<b>Utilization of the API</b> .....	-24-
	5.1 VI Calling Order .....	-24-
	5.1.1 Initialization .....	-24-
	5.1.2 Acquisition/Transfer .....	-24-
	5.1.3 Exit .....	-24-

5.2	Processing the Acquired Points .....	-24-
5.2.1	Array <i>IpBufInfo32</i> Arrangement .....	-25-
5.2.2	Scaling of the Acquired Values According to the Selected Input Range .....	-25-
5.2.3	Sequence of the Channels in Arrays <i>ValMax</i> and <i>ValMin</i> .....	-26-
5.2.4	Scaling of the Maximum and Minimum Values .....	-26-
Appendix A	<b>Equivalence Between the C++ and LabVIEW Data Types</b> .....	-27-

## 1. Introduction

To permit the use of the Data Acquisition Interface module (Lab-Volt Model 9062) with a software different than that coming with this module (LVDAM-EMS), an Application Programming Interface (API) is now available.

The API consists of the following:

- 1) a dynamic link library (DLL) named *LV9062USB.dll*;
- 2) a library of virtual instruments (VI's) built from within the LabVIEW<sup>®</sup> software and named *LV9062USB.lib*.

The DLL, which is written in C++, contains all the functions permitting the use of the Data Acquisition Interface module, models 9062-1X and 9062-BX (X being a number designating a particular language). For each function of the DLL, a virtual instrument (VI) has been created in LabVIEW to facilitate the calling of this function. Thus, the parameters of a function correspond to the connectors of the VI. The user simply has to connect the proper controls and indicators to the different connectors of the VI.

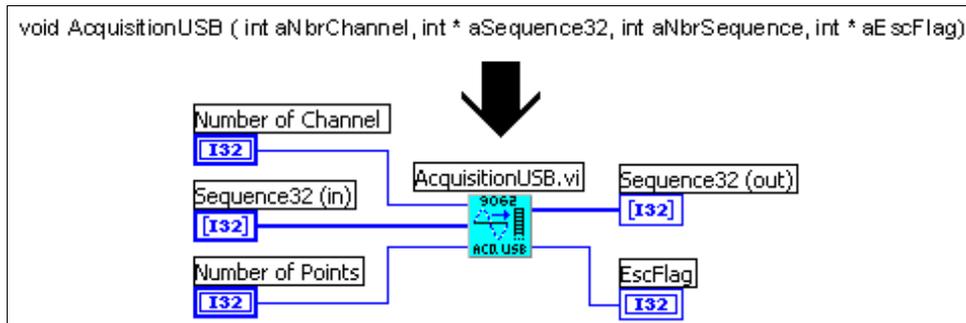


Figure 1. Example of a VI with five parameters.

The API was developed to facilitate its use with LabVIEW. However, it is possible to use the DLL with other instrumentation softwares that permit external code calling, or to develop applications that use the DLL.

The present document is intended for use mainly by instructors and technicians. It permits familiarization with the various VI's specific to the Data Acquisition Interface module (Model 9062) and, therefore, with the various functions of the DLL.

Firstly, a brief explanation of how to install the API is provided, then each VI contained in the DLL is described. Next, the features of the API are described by means of an application example that uses a basic dual-channel oscilloscope. Finally, explanations specific to the basic structure of an application and to the processing of the acquired data are provided.

## 2. API Installation

### 2.1 Requirements

The dynamic link library (.dll) requires a functional installation of the Lab-Volt LVDAM-EMS software, version 3.0 or higher. Refer to the documentation provided with this software to install it.

**Note:** *The dynamic link library is incompatible with the Data Acquisition Interface modules that use a serial port, that is, with Models 9062-0X and 9062-AX (X being a number designating a particular language).*

The library of VI's (.lib) requires a functional installation of the *National Instrument LabVIEW* software, version 6.0 or higher. Refer to the documentation provided with this software to install it.

### 2.2 Installation

Copy files *LV9062USB.dll* and *LV9062USB.lib* in a same directory. The suggested directory is as follows: C:\Program Files\National Instruments\LabVIEW 6\user.lib\.

**Note:** *To be able to use the API, the LVDAM-EMS software must be closed since both use the same USB communication with the Data Acquisition Interface module.*

### 2.3 Opening the VI's

If the libraries have been installed in the directory suggested in Section 2.2, it is possible to access all the virtual instruments (VI's) from within LabVIEW, through accessing the software *paletteMenu*, available under the *User Libraries* group of this software (figure 2).

To do so, launch LabVIEW and then click on *New VI*. Then, click on the right mouse button in the *Diagram* window to make the *Functions* palette appear, then click on the  button to display the *User Libraries* group. Point to the  button to make the *paletteMenu* appear with all VI's. The  button in this palette also permits the display of all VI's in *Diagram* window.

If the libraries have not been installed in the directory suggested in Section 2.2, a VI can be opened by specifying the directory where the VI's library is stored (figure 3).

To do so, launch LabVIEW and then click on *Open VI*. This will bring up the *Choose the VI to open:* dialog box. In this dialog box, select the drive and folder where file *LV9062USB.lib* is stored, select this file and then click on *Open*, which will bring up the *File Dialog* box with a scroll-down list of the VI's. In this list, select the VI named *9062USB VI Tree* and then click OK. All the VI's are contained in the *Diagram* window.

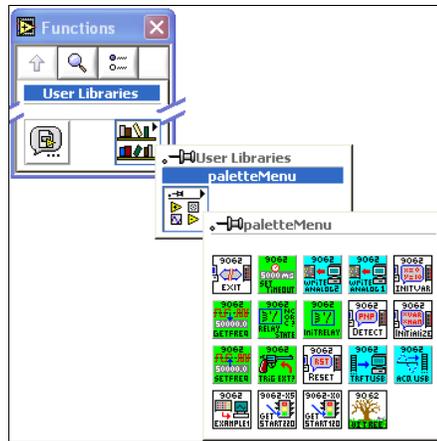


Figure 2. Selecting VI's from the *User Libraries* group.

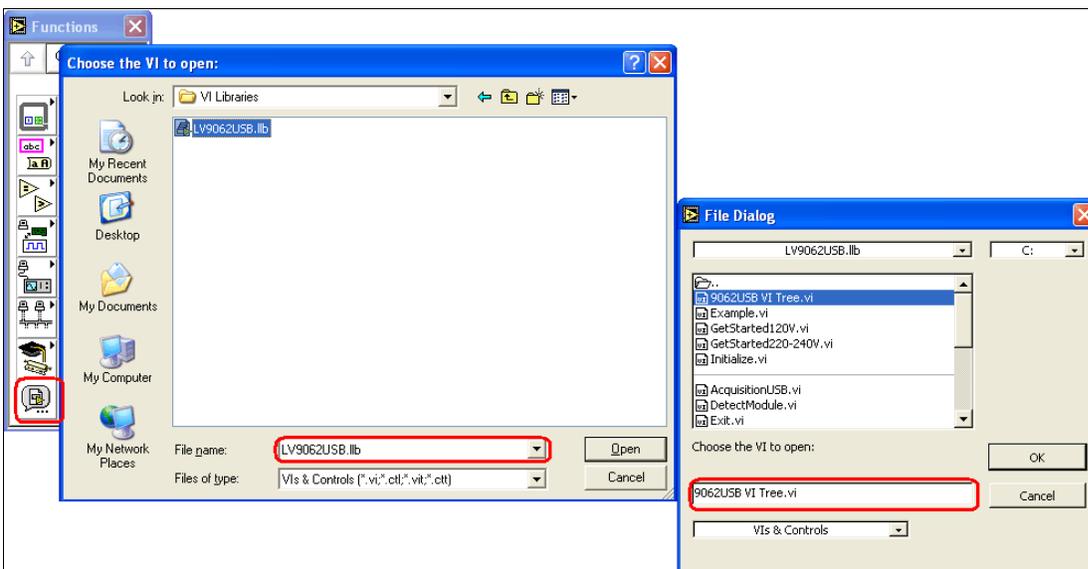


Figure 3. Selecting VI's from the *LV9062USB* library.

### 3. Description of the Library of Virtual Instruments (VI's)

#### 3.1 Demonstration VI's

This section presents the LabVIEW commands (VI's) created for the purpose of showing the various possibilities of the Application Programming Interface (API). These commands can be accessed from the *paletteMenu*, available under the *User Libraries* group of LabVIEW.

##### 3.1.1 9062USB VI Tree

Displays all the VI's available in the library.



##### 3.1.2 Initialize

Initializes the USB communication through the use of the following VI's: *InitVariable*, *SetTimeOut*, and *DetectModule*.



##### 3.1.3 Example

Contains the VI's required to perform the acquisition and the transfer of the measurement points. The USB communication must have been previously initialized.



##### 3.1.4 GetStarted120V → 9062-X0 Module GetStarted220-240V → 9062-X5 Module

Brings up an oscilloscope screen made out from the following VI's: *Initialize*, *Example*, and *Exit*. The number of channels that can be observed simultaneously is limited to two. Channel 1 is intended for the monitoring of voltage inputs E1, E2, and E3 only. Channel 2 is intended for the monitoring of current inputs I1, I2, and I3 only.



### 3.2 Description of the VI's of the Data Acquisition Interface module

This section provides detailed information on all the VI's created for utilization of the Data Acquisition Interface module. Each VI calls a function of the DLL. Consequently, most of the VI's have input parameters with specific type of data, and which can also return a value. Appendix A of this document provides a table that lists the types of data of LabVIEW and their equivalent in C++.

#### 3.2.1 InitVariable

Must be the **first function called** in LabVIEW since it initializes several global variables. This function is included in the *Initialize VI*.



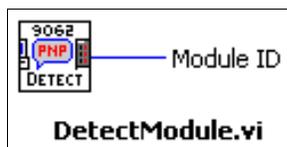
#### 3.2.2 Exit

Must be the **last function called** since it closes Handles required for control of the USB communications.



#### 3.2.3 DetectModule

Detects the module connected to the computer. This function is included in the *Initialize VI*.

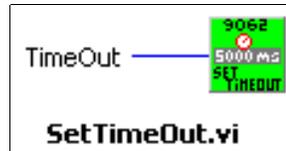


**Module ID** is an identification number indicating the supply voltage of the Data Acquisition Interface module:

- 0: The module is powered by an AC voltage of 120 V/60 Hz.
- 5: The module is powered by an AC voltage of 220 V/50 Hz, or 240 V/50 Hz.
- 1: There is no module connected to the computer.

### 3.2.4 SetTimeOut

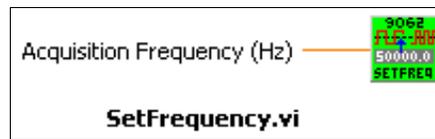
Sets the time delay (in milliseconds) between connection tests with the module.



**TimeOut** is a dwell time between 0 and 60 000 milliseconds.

### 3.2.5 SetFrequency

Sets the acquisition frequency, in hertz (Hz).



**Acquisition Frequency** is a value comprised between 0.0 and 180 000.0 Hz.

### 3.2.6 GetFrequency

Returns the approximate value of the current acquisition frequency, in hertz.



**Acquisition Frequency** is a value between 0.0 and 180 000.0 Hz.

### 3.2.7 InitRelay (WriteDigital.vi)

Sets the status of the relays used to change the gain at the voltage inputs (E1, E2, and E3), and at the current inputs (I1, I2, and I3).



#### I32

**Data** represents a binary value of 6 bits, each bit corresponding to the status (open/closed) of the relay associated with a specific voltage or current input. This value is comprised between 0 and 63 or, in binary, between 000000 and 111111. The bit assignment is as follows:

MSB					LSB
Current Input I3	Current Input I2	Current Input I1	Voltage Input E3	Voltage Input E2	Voltage Input E1

Where 0 = High range gain  
1 = Low range gain

Table 1. Data bit arrangement.

#### I32

**EscFlag** is an error code:

- 0 = No error detected.
- 1 = The *Escape (ESC)* key has been pressed.
- 2 = An error has occurred.

### 3.2.8 RelayState (ReadGlobalRangeUSB.vi)

Returns the status of the relays used to change the gain at the current and voltage inputs.



**I32**

**Range** is a decimal value between 0 and 63 corresponding to a binary value between 000000 and 111111. Each bit indicates the status of the relay associated with a specific input, as indicated below.

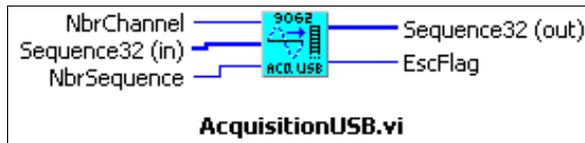
<b>MSB</b>					<b>LSB</b>
Current Input I3	Current Input I2	Current Input I1	Voltage Input E3	Voltage Input E2	Voltage Input E1

Where 0 = High range gain  
1 = Low range gain

**Table 2. Range bit arrangement.**

### 3.2.9 AcquisitionUSB

Permits the acquisition of all points (*NbrChannel* x *NbrSequence*) in the channel order stated by list *Sequence32 (in)*. Points represented by 12-bit binary values are stored in a 32-kilobyte memory on the acquisition board.



**I32**

**NbrChannel** is the number of channels through which data is to be acquired (between 0 and 16).

**I32**

**Sequence32 (in)** states the order in which data must be acquired through each channel (channel acquisition order) before the VI is executed. The size of the array corresponds to the number of channels specified by *NbrChannel*. The possible values for each element are comprised between 0 and 15, according to the arrangement below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E1	E2	E3	I1	I2	I3	T	N	AI1	AI2	AI3	AI4	AI5	AI6	AI7	AI8

**Table 3. Sequence32 (in).**

**I32**

**NbrSequence** is the number of acquisition points for each channel (between 0 and 8192).

[I32]

**Sequence32 (out)** indicates the order in which data must be acquired through each channel (channel acquisition order) following the execution of the VI. The specifications are the same as those mentioned for *Sequence32 (in)*.

[I32]

**EscFlag** is an error code.

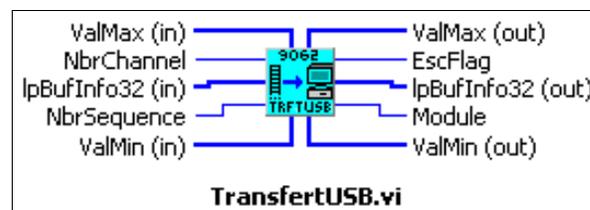
0 = No error detected.

1 = The *Escape (ESC)* key has been pressed.

2 = An error has occurred.

### 3.2.10 TransfertUSB

Permits the transmission, via the USB communication link, of all the acquired points on the form of 64-bit data packets. The points can be accessed through array *IpBufInfo32 (out)*. The lists *ValMax (out)* and *ValMin (out)* can be used to calculate RMS values or to achieve automatic selection of the input range. Refer to Section 5.2 for information on how the values in arrays *IpBufInfo32 (out)*, *ValMax (out)*, and *ValMin (out)* are processed.



[I32]

**NbrChannel** is the number of channels through which data is to be acquired (between 0 and 16).

[I32]

**NbrSequence** is the number of points to acquire on each channel (from 0 and 8192).

[I32]

**IpBufInfo32(in)** corresponds to the array containing all the acquisition points before the VI is executed. The maximum size of the array is 8192 elements. The value of an element is comprised between 0 and 4095.

[I32]

**ValMax (in)** represents the list of the maximum values for each channel. The size of the array corresponds to the number of channels specified by *NbrChannel* (between 0 and 16). The value of an element is comprised between -2047 and 2048.

[I32]

**ValMin (in)** represents the list of the minimum values for each channel. The size of the array corresponds to the number of channels specified by *NbrChannel* (between 0 and 16). The value of an element is comprised between -2047 and 2048.

[I32]

**EscFlag** is an error code.

0 = No error detected.

1 = The *Escape (ESC)* key has been pressed.

2 = An error has occurred.

[I32]

**Module** is a number indicating the supply voltage of the Data Acquisition Interface module:

0: The module is powered by an AC voltage of 120 V/60 Hz.

5: The module is powered by an AC voltage of 220 V/50 Hz, or 240 V/50 Hz.

-1: There is no module connected to the computer.

[I32]

**IpBufInfo32(out)** corresponds to the array containing all the acquisition points of each channel transferred during execution of the VI. The specifications are the same as those mentioned for *IpBufInfo32(in)*.

[I32]

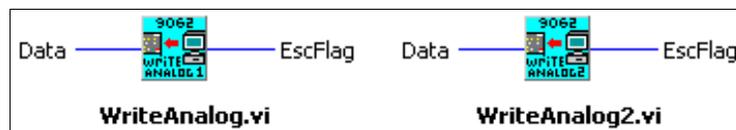
**ValMax (out)** indicates the maximum values of each channel in the order stated by *Sequence32*. The specifications are the same as those mentioned for *ValMax (in)*.

[I32]

**ValMin (out)** indicates the minimum values of each channel in the order stated by *Sequence32*. The specifications are the same as those mentioned for *ValMin (in)*.

### 3.2.11 WriteAnalog (2)

Permits the writing of a 12-bit binary value received from the *Data* input into a register associated with the analog output. This binary value is converted into a voltage comprised between -10 and +10 V before it is applied across the analog output terminals.



I32

**Data** corresponds to the magnitude of the voltage present at the analog output, according to the scale below.

DATA	VOLTAGE
4095	+10 V
2048	0 V
0	-10 V

**Table 4. Voltage versus Data.**

I32

**EscFlag** is an error code.

0 = No error detected.

1 = The *Escape (ESC)* key has been pressed.

2 = An error has occurred.

### 3.2.12 TestTrig

Used to activate the *Sync. Input* to permit synchronization with an external signal.



TF

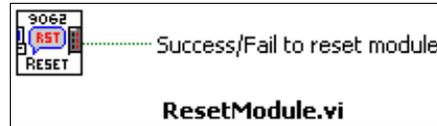
**Activate Sync. Input** indicates the selected type of synchronization.

True: External synchronization using the signal present at the *Sync. Input*.

False: Internal synchronization.

### 3.2.13 ResetDevice

Reinitializes the communication between the Data Acquisition Interface module and the computer. Used after an interruption in USB communication or AC power supply to the module.



**Success/Fail to reset module** indicates whether or not the reinitialization has been successfully performed.

True: Successful reinitialization  
False: Reinitialization has failed

#### 4. Demonstration of the Features of the Application Programming Interface (API)

To permit a better understanding of the VIs' interaction, a complete example involving the use of the API is provided within the library of VIs. This example uses the Data Acquisition Interface module as a programmable dual-channel oscilloscope. The measured signals are generated by an AC series-parallel circuit which you have to connect as indicated below.

##### 4.1 Required Material

- One Lab-Volt EMS Workstation with a set of connection leads
- One Power Supply, Lab-Volt Model 8821
- One Resistive Load module, Lab-Volt Model 8311
- One Data Acquisition Interface module, Lab-Volt Model 9062
- One USB cable

##### 4.2 Circuit Setup

Figure 4 shows the AC series-parallel circuit to connect. This circuit consists of a single-phase, variable AC power supply connected to a resistive load. The resistor value of R1, R2, and R3 must be selected for the AC line voltage you are using, as indicated in Table 5. E1, E2, and E3, as well as I1, I2, and I3 correspond to the inputs of the Data Acquisition Interface module, Model 9062-XX.

AC LINE VOLTAGE (V)	R1 ( $\Omega$ )	R2 ( $\Omega$ )	R3 ( $\Omega$ )
120	300	600	300
220	1100	2200	1100
240	1200	2400	1200

Table 5. Resistor values as a function of the AC line voltage.

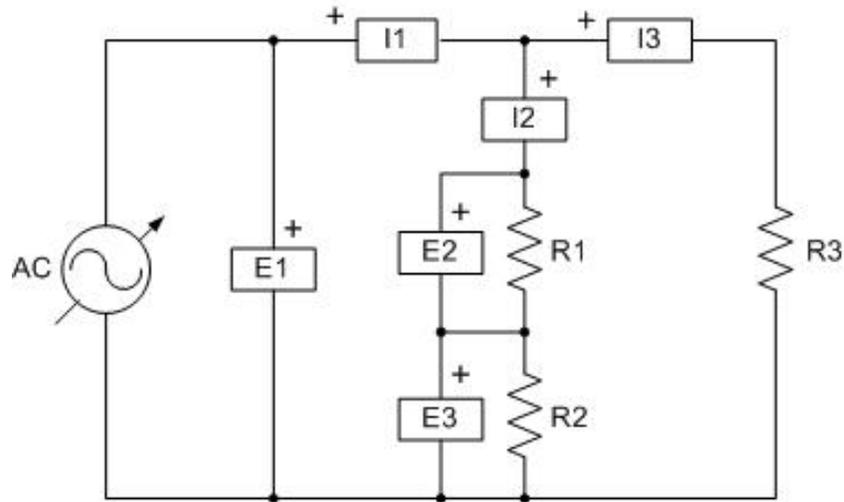


Figure 4. AC series-parallel circuit.

1. Connect the circuit shown in Figure 4.
2. Connect the Data Acquisition Interface module to the computer through the USB cable, then turn on this module.
3. Verify your circuit setup using the LVDAM-EMS software. To do so, use the *Metering* application of this software and set the display refresh to *Continuous Refresh*.
4. Turn on the Power Supply and adjust its main control voltage knob to 100%.
5. Make sure that the following circuit parameters can be measured by the *Metering* application: voltages E1, E2, and E3, as well as currents I1, I2, and I3. Table 6 below lists the theoretical values for these voltages and currents.

AC LINE VOLTAGE	R1		R2		R3	
	VOLTAGE (V)	CURRENT (A)	VOLTAGE (V)	CURRENT (A)	VOLTAGE (V)	CURRENT (A)
120	40	0.1333	80	0.1333	120	0.4
220	73.33	0.0666	146.66	0.0666	220	0.2
240	80	0.0666	160	0.0666	240	0.2

Table 6. Theoretical values for the voltages and currents.

6. Turn off the Power Supply and the Data Acquisition Interface module.

### 4.3 Execution of the Example VI

Before using the API within LabVIEW, make sure the LVDAM-EMS software is closed. To access the example VI, launch LabVIEW and open the library *LV9062USB.lib* previously installed. In the File Dialog box (Figure 5), select the proper VI file between *GetStarted120V* or *GetStarted220V-240V*, depending on the supply voltage of the Data Acquisition Interface model you are using, and then click on OK.

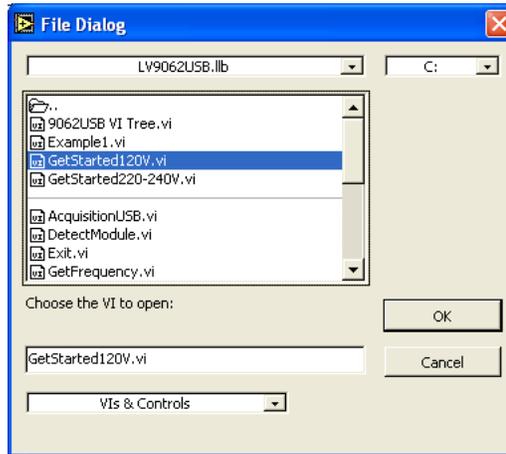


Figure 5. Opening a VI file from the LV9062USB.lib library.

The control panel window shown in Figure 6 should appear, allowing you to set the various parameters for the signal acquisition.

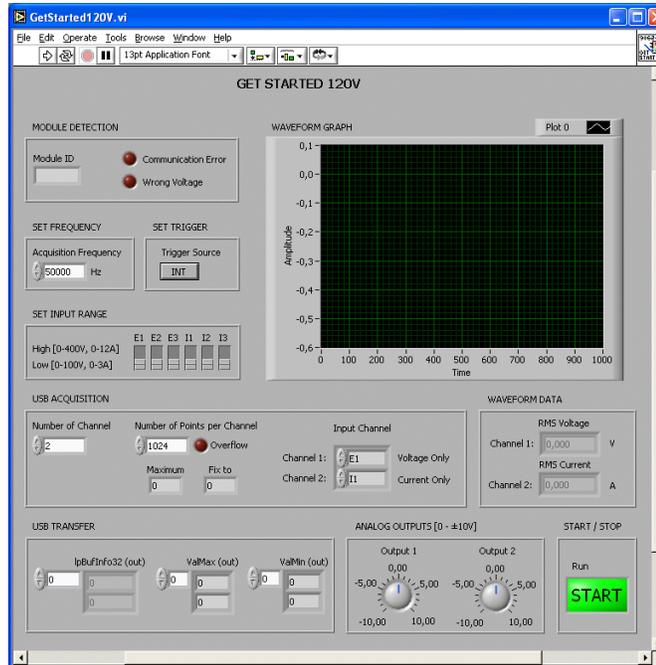


Figure 6. Control panel window.

1. Turn on the Data Acquisition Interface module and wait for at least 5 seconds in order for this module to be detected by the system.
2. In the Control panel window, click on the  button (*Run Continuously*) to execute the application. The model number of your Data Acquisition Interface module should appear in the section *MODULE DETECTION*. Otherwise, an LED in this section will turn on to indicate the source of the problem.
3. Leave the various controls in the Control panel window set as they are (default settings). Click on the **START** button in this window to start the acquisition.
4. Power on the AC series-parallel circuit. To do so, adjust the main control voltage knob of the Power Supply to 50%. Two sine waves should appear in the *WAVEFORM GRAPH* section of the Control panel window. The waveform acquired through channel 1, which corresponds to the voltage at the E1 input, is displayed in white; the waveform acquired through channel 2, which corresponds to the current at input I1, is displayed in red.

**Note:** The waveform of the I1 current displayed on the *WAVEFORM GRAPH* seems to have a weak amplitude, due to the fact that the graph scale is set by the signal of highest amplitude, in that case, the signal on channel 1 (voltage at the E1 input).

## 4.4 Exploring the Sections of the Control Panel

### 4.4.1 Module Detection

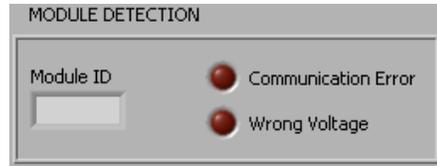


Figure 7. *MODULE DETECTION* section.

The model number of the Data Acquisition Interface module you are using appears in the *Module ID* field once the communication is established. The LED *Communication Error* will turn on if the communication cannot be established. The LED *Wrong Voltage* indicates that the example VI file you have opened does not correspond to the supply voltage of the Data Acquisition Interface module.

### 4.4.2 Set Frequency

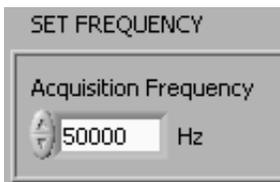


Figure 8. *SET FREQUENCY* section.

The acquisition frequency can be varied to modify the number of cycles displayed on the graph. For example, to observe a single cycle of a signal having a frequency of 60 Hz, the acquisition frequency is set to around 120 000 Hz, assuming that two channels with 1024 acquisition points are used (or to around 100 000 Hz if the signal has a frequency of 50 Hz).

**Note:** *When data acquisition is multiplexed over several channels (up to 16 channels), the actual acquisition frequency corresponds to the set frequency divided by the number of channels specified in the USB ACQUISITION section.*

#### 4.4.3 Set Trigger



Figure 9. SET TRIGGER section.

The source of synchronization can be an external signal having a TTL level applied to the SYNC INPUT.

**Note:** With the example currently studied, internal synchronization on one of the channels is not possible.

#### 4.4.4 Set Input Range

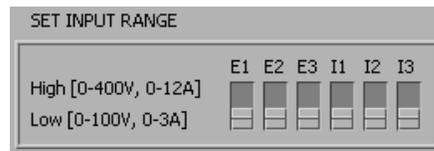


Figure 10. SET INPUT RANGE section.

The input range of the voltage inputs (E1, E2, and E3) and current inputs (I1, I2, and I3) can be changed according to the magnitude of the signals applied to these inputs. For example, you can set the Power Supply output voltage to 100% and then place the SET INPUT RANGE switch of channel 1 (E1 voltage input) to the HIGH position. The measured voltage should be approximately equal to the AC line voltage.

**Note:** When the SET INPUT RANGE switch of one of the voltage inputs is set to HIGH, the scale of the graph will automatically be 400 V (or 800 V for the 200-240 V modules) for all the other voltage inputs, regardless of the setting of their range switch. For example, if the E1-input voltage is displayed on the graph and the range switch for this input is set to LOW, but the range switch for the E2 input is set to HIGH, the scale of the graph will be 400 V, not 100 V (LOW scale). The same limitation applies for the selection of the input range of current inputs I1 through I3.

#### 4.4.5 USB Acquisition

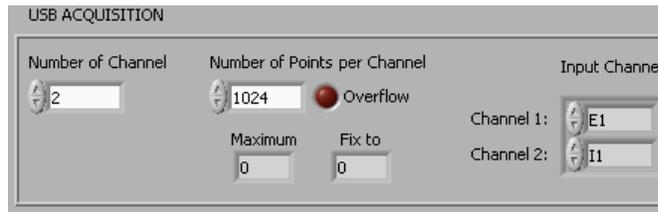


Figure 11. ACQUISITION USB section.

The number of channels used in the current example cannot be changed. This number is limited to two, due to the non-dynamic processing of the points contained in the buffer memory. The number of points per channel can be specified. The maximum number of points is dependent upon the number of channels used, since the buffer memory has a capacity of 32 kilobytes. The signals that can be observed through channel 1 are limited to those at voltage inputs E1, E2, and E3. The signals that can be observed through channel 2 are limited to those at current inputs I1, I2, and I3. This limitation is due to the fixed input-range programming for the two channels (voltage or current). You may want to further experiment with this example by using channels 1 and 2 to perform measurements and observations at different other points of the AC series-parallel circuit.

**Note:** When the number of points is modified, the scale of the X axis (Time) of the graph must also be changed in order for all points to be displayed.

#### 4.4.6 USB Transfer

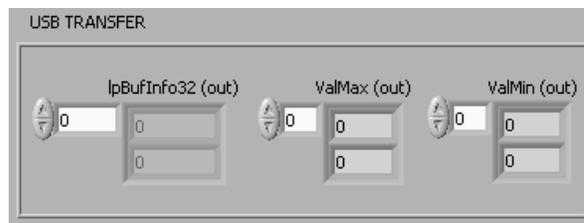


Figure 12. USB TRANSFER section.

The decimal values representing the acquired points are displayed in the array *IpBufInfo32*. The value at index 0 corresponds to the first point of channel 1, while that at index 1 corresponds to the first point of channel 2. Thus, the display of the values alternates between channel 1 and channel 2. The index of the array can be changed to know the values of the other points. Moreover, the maximum and minimum values of each channel are displayed in arrays *ValMax* and *ValMin*. Index 0 corresponds to the extreme values of channel 1, while index 1 corresponds to those of channel 2.

#### 4.4.7 Analog Outputs

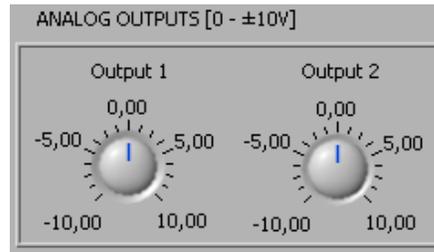


Figure 13. ANALOG OUTPUTS section.

The adjustment of the voltage for each analog output is performed by means of a control knob. This permits adjustment of the DC output voltage between -10 and +10 V. The voltage adjustment can be performed only when the acquisition is stopped.

## 5. Utilization of the API

The development of an application with the Data Acquisition Interface module requires a basic structure of the VI's in order to be able to perform the acquisition and transfer of the acquired data. Moreover, the transferred data must be processed in order for the measurements to be adequately represented.

### 5.1 VI Calling Order

The Diagram window for the example VI previously studied, accessible through selecting Show Diagram from the Window menu, reveals that this VI is made up of three main steps, which are: initialization, acquisition/transfer, and exit.

#### 5.1.1 Initialization

This first step is performed within *Initialize.vi* in the example VI. Through the use of *InitVariable.vi*, the DLL variables required for the acquisition and transfer of data are initialized. Moreover, upon detection of the Data Acquisition Interface module with *DetectModule.vi*, a Handle representing the Data Acquisition Interface module is created by the DLL to permit communication with the computer.

#### 5.1.2 Acquisition/Transfer

Once the communication has been established, it is necessary to set the acquisition frequency (*SetFrequency.vi*), then the status of the relays (*WriteDigital.vi*), and then the source of synchronization (*TestTrig.vi*) before the acquisition can be performed. There is no delay required between the acquisition and the transfer of data since the DLL already creates a delay.

**Note:** *Parameters such as Sequence32, lpBufInfo32, ValMax, and ValMin require arrays (Array) whose number of enabled cells correspond to the required size. Otherwise, an error will occur when a call is made to the DLL.*

#### 5.1.3 Exit

This last step ensures that Handles used for the USB communication are deleted from the memory.

### 5.2 Processing of the Acquired Points

The points sampled on each channel, as indicated in array *Sequence32* of *Acquisition USB*, are stored in array *lpBufInfo32(out)* of *TransfertUSB*. They are represented by 12-bit binary values, which corresponds to decimal values between 0 and 4095. Moreover, the maximum and minimum values of each channel are available in arrays *ValMax (out)* and *ValMin (out)*.

### 5.2.1 Array *IpBufInfo32* Arrangement

The way in which the sequence of points of a channel is stored in array *IpBufInfo32* is dependent upon the number of channels used for the acquisition (*NbrChannel*) and on the order in which data is acquired through each channel (*Sequence32*). This can be compared to the way in which cards are distributed to players before a card game. The number of players corresponds to the number of channels, and the seat (or position) of each player corresponds to the order in which data is acquired through each channel (channel acquisition order).

For example, if channels E1, I2, and E2 are used in order, the points of E1 will be at indexes 0, n, 2n, ..., where n is the number of elements contained in the array. The acquisition points of I2 will be at indexes 1, n+1, 2n+1, and so on. The acquisition points of E2 will be at indexes 2, n+2, 2n+2, and so on.

### 5.2.2 Scaling of the Acquired Values According to the Selected Input Range

To correspond to the actual measurements, the acquired values must be scaled according to the selected input ranges (see table 7 below). This is performed according to the process shown in Figure 14.

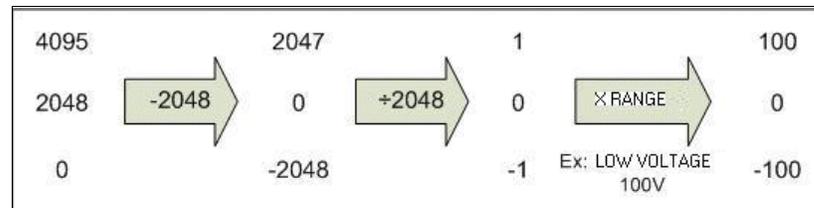


Figure 14. Scaling of the values of array *IpBufInfo32*.

	RANGE	120-V MODEL	220/240-V MODEL
VOLTAGE INPUT	Low	± 100 V	± 200 V
	High	± 400 V	± 800 V
CURRENT INPUT	Low	± 3 A	± 1.5 A
	High	± 12 A	± 6 A
ANALOG INPUT		± 10 V	± 10 V
AUXILIARY ANALOG INPUT		± 10 V	± 10 V

Table 7. Input ranges for the Data Acquisition Interface modules, models 120 V and 220/240 V.

### **5.2.3 Sequence of the Channels in Arrays *ValMax* and *ValMin***

The sequence of the maximum and minimum in the arrays correspond to the channel acquisition order (*Sequence32*). Consequently, assuming the following channel sequence: E1, I2, and E2, index 0 is for E1, index 1 represents the maximum and minimum values of I2, and index 2 indicates the extreme values of the voltage at input E2.

### **5.2.4 Scaling of the Maximum and Minimum Values**

As earlier mentioned for array *lpBufInfo32*, the acquired values must be scaled according to their selected input range in order for these values to correspond to the actual values. The scaling process is the same as that described in Section 5.2.2, except that the first step, which consists in a -2048 translation, is not required since it has already been done by the DLL. The next steps consist in performing a division by 2048 and then a multiplication by the range of the selected channel.

## Appendix A Equivalence Between the C++ and LabVIEW Data Types

	LabVIEW TYPE		C++ TYPE	
		CONTROL		INDICATOR
<b>8 BITS</b>	Boolean			bool
<b>32 BITS</b>	Long signed integer			int, long
<b>32 BITS</b>	Array of long integer			int*
<b>32 BITS</b>	Long unsigned integer			unsigned int, unsigned long
<b>64 BITS</b>	Double-precision floating-point			double

Table 8. Equivalence between the C++ and LabVIEW data types.

