VersaSTAT Developers Kit (VDK) Manual

Version 1.42

2/3/2010

1.0 Overview

The VersaSTAT Developers Kit (VDK) contains a .NET Assembly (Control) that provides control and data retrieval functionality of all VersaSTAT based instruments. This control can be used with any programming language that supports .NET 3.5 Assemblies, such as Microsoft Visual Studio and LabView. Examples for both are provided and explained below.

As the VersaSTAT Developers Kit (VDK) is a free product, Princeton Applied Research can only provided limited support. The VDK is provided “AS IS” and does not include any warranty of any type.

2.0 Programmers Reference

Below is the complete list of methods and properties provided by the control.

Instrument class (allows connection to instrument):

Methods:

int FindNext(int iStartIndex) - returns the serial number of the next available instrument. The iStartIndex parameter is used if more than one instrument is attached to the USB. To request the Serial Number for the first instrument, iStartIndex should be 0.

bool Connect(int iSerialNumber) – Connects to the instrument specified by the iSerialNumber parameter. Returns true if successful, false if not.

void Close() – Closes the connection to the instrument.

string GetSerialNumber() – Returns the serial number of the currently connected instrument.

string GetModel() – Returns the model of the currently connect instrument (ie VersaSTAT 3 – 400).

string GetOptions() – Returns the options of the currently connected instrument (ie FRA).

Properties:

Immediate Immediate – Returns a reference to the Immediate Class

Experiment Experiment – Returns a reference to the Experiment Class

Immediate Class (allows immediate control of the instrument)

Methods:

void SetCellOn() – Turns the instruments Cell On
void SetCellOff() – Turns the instruments Cell Off

void SetCellExternal() – Sets the instrument to use the External Cell
void SetCellInternal() – Sets the instrument to use the Internal Cell

void SetModePotentiostat() – Sets the instrument to potentiostatic mode
void SetModeGalvanostat() – Sets the instrument to galvanostat mode

void SetIRange_2A() – Sets the current range to 2A
void SetIRange_200mA() – Sets the current range to 200mA
void SetIRange_20mA() – Sets the current range to 20mA
void SetIRange_2mA() – Sets the current range to 2mA
void SetIRange_200uA() – Sets the current range to 200uA
void SetIRange_20uA() – Sets the current range to 20uA
void SetIRange_2uA() – Sets the current range to 2uA
void SetIRange_200nA() – Sets the current range to 200nA
void SetIRange_20nA() – Sets the current range to 20nA (if instrument is able)
void SetIRange_4nA() – Sets the current range to 4nA (if instrument is able)

void SetDCPotential(float fVoltage) – Sets the output DC potential to the amount provided with the fVoltage parameter, in Volts. Note this value must be within the instruments capability.

void SetDCCurrent(float fCurrent) – Sets the output DC current to the amount provided with the fCurrent parameter, in Amps. Calling this method also changes to Galvanostat mode and sets the current range to the correct value. WARNING: Once cell is enabled after setting the DC current, do not change to potentiostatic mode or change the current range, these will affect the value being applied to the cell. Note this value must be within the instruments capability.

void SetACFrequency(float fFrequency) – Sets the output AC Frequency to the value provided with the fFrequency parameter, in Hz. Note this value must be within the instruments capability.

void SetACAmplitude(float fRMSAmplitude) – Sets the output AC Amplitude to the value provided with the fRMSAmplitude parameter, in RMS Volts. Note this value must be within the instruments capabilities.

void SetACWaveformOn() – Enables the AC Waveform
void SetACWaveformOff() – Disables the AC Waveform

void UpdateStatus() – This method retrieves the status information from the instrument (it also causes the instrument to auto-range the current if an experiment sequence is not in progress). Call this prior to calling the status methods below.

double GetE() – Returns the latest stored E value
double GetI() – Returns the latest stored I value
int GetOverload() – Returns the latest overload information. 0 indicates no overload, 1 indicates I (current) Overload, 2 indicates E, Power Amp or Thermal Overload has occurred.
bool GetBoosterEnabled() – Returns the status of the booster switch, if enabled, true is returned, otherwise false is returned.
bool GetCellEnabled() – Returns the status of the cell, if enabled, true is returned, otherwise false is returned.
string GetIRange() – Returns the current range the instrument is on. Values will be either 2A, 200mA, 20mA, 2mA, 200uA, 20uA, 2uA, 200nA, 20nA or 4nA.
void SetIsolationModeOn() – For the VersaSTAT 3F ONLY
void SetIsolationModeOff() – For the VersaSTAT 3F ONLY
void SetACNotchFilter_None() - For the VersaSTAT 3F ONLY
void SetACNotchFilter_50Hz() - For the VersaSTAT 3F ONLY
void SetACNotchFilter_60Hz() - For the VersaSTAT 3F ONLY
void SetACFilters_Normal() - For the VersaSTAT 3F ONLY
void SetACFilters_Aggresive() - For the VersaSTAT 3F ONLY
void SetACFilters_MoreAggressive() - For the VersaSTAT 3F ONLY

void SetAutoIRangeOn() – Enables (default is enabled) automatic current ranging while an experiment is not running.

void SetAutoIRangeOff() –Disables automatic current ranging while an experiment is not running. This is useful when wanting to apply a DC current in immediate mode.

Experiment Class (sets up action sequences and retrieves data)

Methods:

string GetActionList() – Returns a list of comma delimited action names that are supported by the instrument that is currently connected.

void Clear() – Clears the current sequence of actions void Start() – Starts the sequence of actions in the instrument that is currently connected.

void Stop() – Stops the sequence of actions that is currently running in the instrument that is currently connected.

void Skip() – Skips the currently running action and immediately starts the next action. If there is no more actions to run, the sequence is simply stopped.

bool IsSequenceRunning() – Returns true if a sequence is currently running on the connected instrument, false if not.

int GetNumPointsAvailable() – Returns the number of points that have been stored by the instrument after a sequence of actions has begun. Returns -1 when all data has been retrieved from the instrument.

Float GetLastMeasuredOC() – Returns the last measured Open Circuit value. This value is stored at the beginning of the sequence (and updated anytime the “AddMeasureOpenCircuit” action is called).

- The following Action Methods can be called in order to create a sequence of Actions. For example, AddOpenCircuit(string) could be called, then AddEISPotentiostatic(string) called. This would create a sequence of two actions, when started, the open circuit experiment would run, then the impedance experiment.

bool AddOpenCircuit(string strParameters) – Adds the Open Circuit experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddLinearScanVoltammetry(string strParameters) – Adds the Linear Scan Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddCyclicVoltammetry(string strParameters) – Adds the Cyclic Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddMultiCyclicVoltammetry(string strParameters) – Adds the Multi-cycle Cyclic Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddStaircaseLinearScanVoltammetry(string strParameters) – Adds the Staircase Linear Scan Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddStaircaseCyclicVoltammetry(string strParameters) – Adds the Staircase Cyclic Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddStaircaseMultiCyclicVoltammetry(string strParameters) – Adds the Staircase Multi-cycle Cyclic Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddChronoamperometry(string strParameters) – Adds the Chronoamperometry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddChronopotentiometry(string strParameters) – Adds the Chronopotentiometry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddChronocoulometry(string strParameters) – Adds the Chronocoulometry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddRecurrentPotentialPulses(string strParameters) – Adds the Recurrent Potential Pulses experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddRecurrentGalvanicPulses(string strParameters) – Adds the Recurrent Galvanic Pulses experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddFastPotentialPulses(string strParameters) – Adds the Fast Potential Pulses experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddFastGalvanicPulses(string strParameters) – Adds the Fast Galvanic Pulses experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddSquarewave(string strParameters) – Adds the Squarewave experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddDiffPulseVoltammetry(string strParameters) – Adds the Differential Pulse Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddNormalPulseVoltammetry(string strParameters) – Adds the Normal Pulse Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddReverseNormalPulseVoltammetry(string strParameters) – Adds the Reverse Normal Pulse Voltammetry experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddZeroResistanceAmmeter(string strParameters) – Adds the Zero Resistance Ammeter (ZRA) experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddGalvanicCorrosion(string strParameters) – Adds the Galvanic Corrosion experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddCyclicPolarization(string strParameters) – Adds the Cyclic Polarization experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddLinearPolarization(string strParameters) – Adds the Linear Polarization experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddTafel(string strParameters) – Adds the Tafel experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddPotentiostatic(string strParameters) – Adds the Potentiostatic experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddPotentiodynamic(string strParameters) – Adds the Potentiodynamic experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddGalvanostatic(string strParameters) – Adds the Galvanostatic experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddGalvanodynamic(string strParameters) – Adds the Galvanodynamic experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddEISPotentiostatic(string strParameters) – Adds the EIS Potentiostatic experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddEISGalvanostatic(string strParameters) – Adds the Galvanostatic experiment to the Action Sequence. Returns true if action added successfully. See Table #1 for information on the required parameters.

bool AddMeasureOpenCircuit() – Forces the VersaSTAT to measure the open circuit, to do this the cell will be turned off momentarily.

bool AddLoop() – Allows a single or group of actions to be repeated n number of times.

- The following Data Retrieval Methods can be called at any time after an Action Sequence has started. Use the “GetNumPointsAvailable()” method to determine the number of points to retrieve. Data is always returned as an array of doubles. Parameter iStart is the starting index location of the data to be retrieved, parameter iNumPoints is the number of points to retrieve. For more detailed information on type, please refer to the instruments manual.

double [] GetDataPotential(int iStart, int iNumPoints)
double [] GetDataCurrent(int iStart, int iNumPoints)
double [] GetDataElapsedTime(int iStart, int iNumPoints)
double [] GetDataPoint(int iStart, int iNumPoints)
double [] GetDataCharge(int iStart, int iNumPoints)
double [] GetDataSyncADCInput(int iStart, int iNumPoints)
double [] GetDataCurrentRange(int iStart, int iNumPoints)
double [] GetDataForwardCurrent(int iStart, int iNumPoints)
double [] GetDataReverseCurrent(int iStart, int iNumPoints)
double [] GetDataDeltaCurrentFR(int iStart, int iNumPoints)
double [] GetDataDeltaCurrentRF(int iStart, int iNumPoints)
double [] GetDataAppliedPotential(int iStart, int iNumPoints)

```
double [] GetDataFrequency( int iStart, int iNumPoints )
double [] GetDataImpedanceMagnitude( int iStart, int iNumPoints )
double [] GetDataImpedanceReal( int iStart, int iNumPoints )
double [] GetDataImpedanceImaginary( int iStart, int iNumPoints )
double [] GetDataImpedancePhase( int iStart, int iNumPoints )
double [] GetDataAdmittanceMagnitude( int iStart, int iNumPoints )
double [] GetDataAdmittanceReal( int iStart, int iNumPoints )
double [] GetDataAdmittanceImaginary( int iStart, int iNumPoints )
double [] GetDataAdmittancePhase( int iStart, int iNumPoints )
double [] GetDataCapacitanceMagnitude( int iStart, int iNumPoints )
double [] GetDataCapacitanceReal( int iStart, int iNumPoints )
double [] GetDataCapacitanceImaginary( int iStart, int iNumPoints )
double [] GetDataVACMagnitude( int iStart, int iNumPoints )
double [] GetDataVACReal( int iStart, int iNumPoints )
double [] GetDataVACImaginary( int iStart, int iNumPoints )
double [] GetDataVACPhase( int iStart, int iNumPoints )
double [] GetDataIACMagnitude( int iStart, int iNumPoints )
double [] GetDataIACReal( int iStart, int iNumPoints )
double [] GetDataIACImaginary( int iStart, int iNumPoints )
double [] GetDataIACPhase( int iStart, int iNumPoints )
double [] GetDataAAIChn0( int iStart, int iNumPoints )
double [] GetDataAAIChn1l( int iStart, int iNumPoints )
double [] GetDataAAIChn2( int iStart, int iNumPoints )
double [] GetDataAAIChn3( int iStart, int iNumPoints )
double [] GetDataSegment( int iStart, int iNumPoints )
double [] GetDataEGain( int iStart, int iNumPoints )
double [] GetDataIGain( int iStart, int iNumPoints )
```

Table #1 (Action Parameters)

Action	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	Parameters																													
Common	RDE	L1T	L1D	L1V																										
Open Circuit	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	CTU	BW																
LSV	IP	Vs	FP	Vs	SR	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW									
CV (Single)	IP	Vs	VP	Vs	VH	AH	FP	Vs	SR	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW					
CV (Multi)	IP	Vs	VP	Vs	VH	AH	VP	Vs	VH	AH	SR	C	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW		
Staircase LSV	IP	Vs	FP	Vs	SH	ST	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW							
Staircase CV (Single)	IP	Vs	VP	Vs	VH	AH	FP	Vs	SH	ST	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW			
Staircase CV (Multi)	IP	Vs	VP	Vs	VH	AH	VP	Vs	VH	AH	SH	ST	C	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW
Chronoamperometry	IP	Vs	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW										
Chronopotentiometry	IC	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	EM	EF	IF	LCO	CTU	BW															
Chronocoulometry	IP	Vs	PE	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW									
Recurrent Pot Pulses	PP	Vs	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW										
Recurrent Gal Pulses	PC	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	EM	EF	IF	LCO	CTU	BW															
Fast Potential Pulses	PP	Vs	Dur	PP	Vs	Dur	PP	Vs	Dur	PP	Vs	Dur	NP	TPP	C	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	BW	
Fast Galvanic Pulses	PC	Dur	PC	Dur	PC	Dur	PC	Dur	NP	TPP	C	L1T	L1D	L1V	L2T	L2D	L2V	AM	EM	EF	IF	LCO	CTU	BW						
Sq. Wave Voltammetry	IP	Vs	FP	Vs	PH	SH	F	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW							
Diff. Pulse Voltammetry	IP	Vs	FP	Vs	PH	PW	SH	SW	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW						
Norm. Pulse Voltammetry	IP	Vs	FP	Vs	PW	SH	SW	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW							
Rev. Norm. Pulse Volt.	IP	Vs	FP	Vs	PW	SH	SW	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	LCO	CTU	iRC	UD	UP	BW							
ZRA	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	CTU	BW																
Galvanic Corrosion	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	EM	EF	IF	CTU	BW																
Cyclic Polarization	IP	Vs	VP	Vs	FP	Vs	SH	ST	T	SL	TL	IR	AM	EM	EF	IF	LCO	CTU	iRC	BW										
Linear Polarization	IP	Vs	FP	Vs	SH	ST	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	BW									
Tafel	IP	Vs	FP	Vs	SH	ST	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	BW									
Potentiostatic	IP	Vs	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	BW											
Potentiodynamic	IP	Vs	FP	Vs	SH	ST	L1T	L1D	L1V	L2T	L2D	L2V	IR	AM	EM	EF	IF	LCO	CTU	iRC	BW									
Galvanostatic	IC	TPP	Dur	L1T	L1D	L1V	L2T	L2D	L2V	AM	EM	EF	IF	LCO	CTU	BW														
Galvanodynamic	IC	Vs	FC	Vs	SH	ST	L1T	L1D	L1V	L2T	L2D	L2V	AM	EM	EF	IF	LCO	CTU	BW											
Potentiostatic EIS	SF	EF	AP	PS	NP	DQ	MD	IP	Vs	IR	EM	LCO	CTU																	
Galvanostatic EIS	SF	EF	AC	PS	NP	DQ	MD	IC	EM	LCO	CTU																			
Loop	NB	I																												

** Note, see Table #2 for Acronym Definitions

Table #2 (Parameter Acronym Definition)

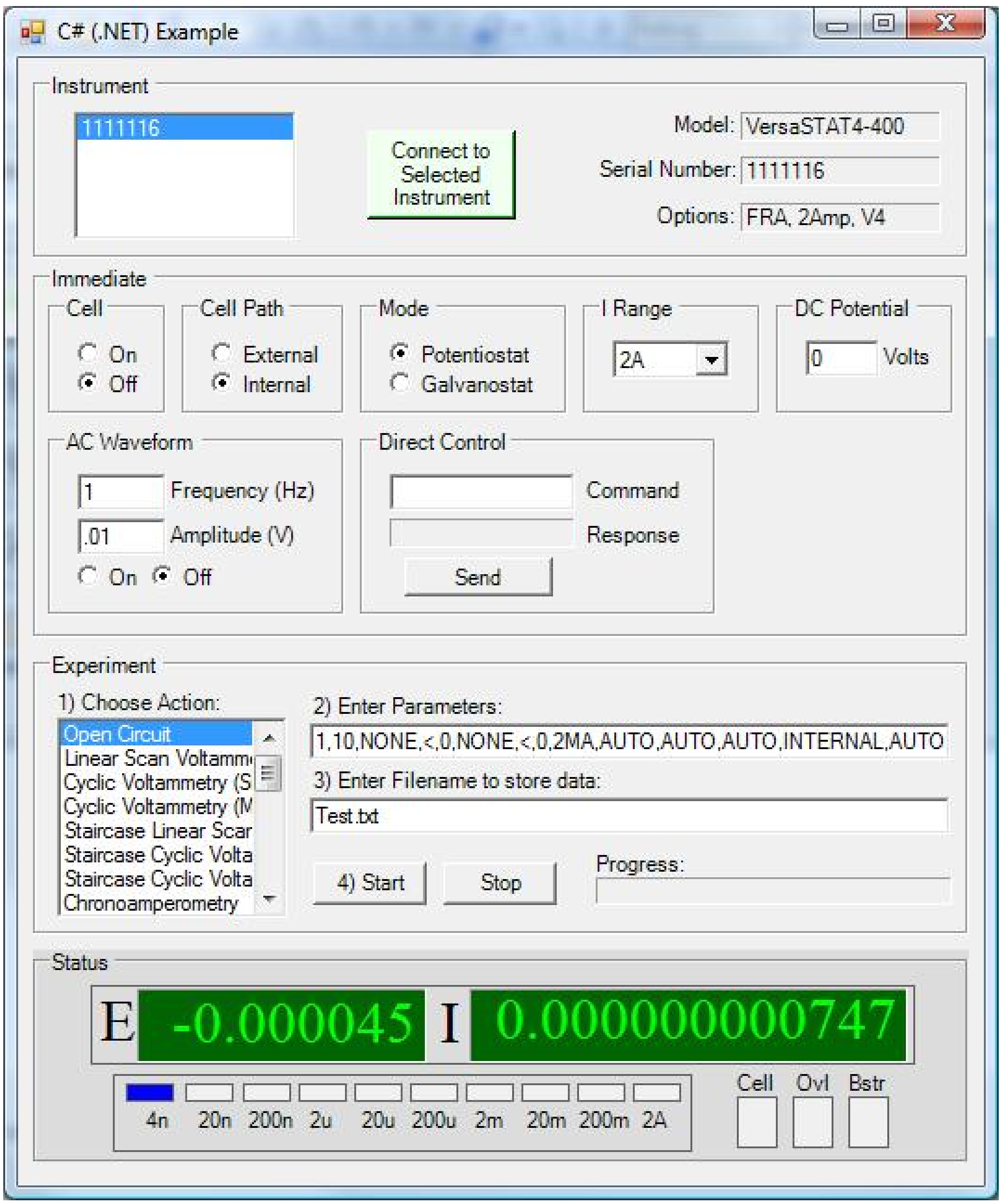
Parameter Acronym	**Definition**	**Units**		**Parameter Values**
IP	Initial Potential	V		User value -10 to 10
IC	Initial Current	A		User value -.650 to .650
Vs	Versus			VS OC, VS REF or VS PREVIOUS
VP	Vertex Potential	V		User value -10 to 10
VH	Vertex Hold	s		User value
AH	Acquire data during Vertex Hold			YES or NO
FP	Final Potential	V		User value -10 to 10
TPP	Time Per Point	s		User value .00001 to ?
Dur	Duration	s		User value .00001 to ?
C	Cycles	#		User value
SH	Step Height	V		User value
ST	Step Time	s		User value
L1T	Limit 1 Type			NONE, CURRENT, POTENTIAL or CHARGE
L1D	Limit 1 Direction			< or >
L1V	Limit 1 Value			User value
L2T	Limit 2 Type			NONE, CURRENT, POTENTIAL or CHARGE
L2D	Limit 2 Direction			< or >
L2V	Limit 2 Value			User value
IR	Current Range		*	AUTO, 2A, 200MA, 20MA, 2MA,200UA,20UA,2UA,200NA
EM	Electrometer			AUTO, SINGLE ENDED or DIFFERENTIAL
EF	E Filter		**	AUTO, NONE, 200KHZ, 1KHZ, 1KHZ, 100HZ 10HZ, 1HZ
IF	I Filter		**	AUTO, NONE, 200KHZ, 1KHZ, 1KHZ, 100HZ, 10HZ, 1HZ
LCO	Leave Cell On			YES or NO
CTU	Cell To Use			INTERNAL or EXTERNAL
iRC	Enable/Disable iR Compensation			ENABLED or DISABLED
UD	User defined the amount of iR Comp	ohms		User value
UP	Use previously determined iR Comp			YES or NO
PE	Pre-Electrolosis time	s		User value
PP	Pulse Potential	V		User value
CP	Pulse Current	A		User value
PH	Pulse Height	V		User value
SR	Scan Rate	V/s		User value
F	Frequency	Hz		User value
PW	Pulse Width	s		User value
SW	Step Width	s		User value
T	Threshold Enable			DISABLED or ENABLED
SL	Start Level	V		User value
TL	Threshold Level	A		User value
AM	Acquisition Mode			AUTO, NONE, 4/4 or AVERAGE
SF	Start Frequency	Hz		User value
EF	End Frequency	Hz		User value
AP	Amplitude (Potential)	V		User value
AC	Amplitude (Current)	A		User value
PS	Point Spacing			LOGARITHMIC or LINEAR
NP	Number of Points (Frequencies to test)			User value (When Log value is Points per Decade)
DQ	Data Quality			User value 0 to 10 (default to 1)
MD	Measurement Delay	s		Delay between impedance points
BW	Bandwidth		***	AUTO, HIGH STABILITY, 1MHZ, 100KHZ, 1KHZ
RDE	RDE Speed	V		Provides a voltage on the DAC Out, located on rear of inst
NA	Number of actions back			Number of actions back in the sequence (1 would repeat t

***** 20NA and 4NA only available on some instruments**
*****100Hz, 10Hz and 1Hz only available on some instruments**
******* High Stability, 1MHz, 100kHz and 10kHz only available on some instruments**

Programming Examples

2.1 Visual Studio

All sources to a C# .NET example is provided in the “\C# Example” folder. This example can be compiled and ran, as is. To add the “VersaSTATControl” to a different project, simply add a reference to the VersaSTATControl file. Then create an instance of the class.

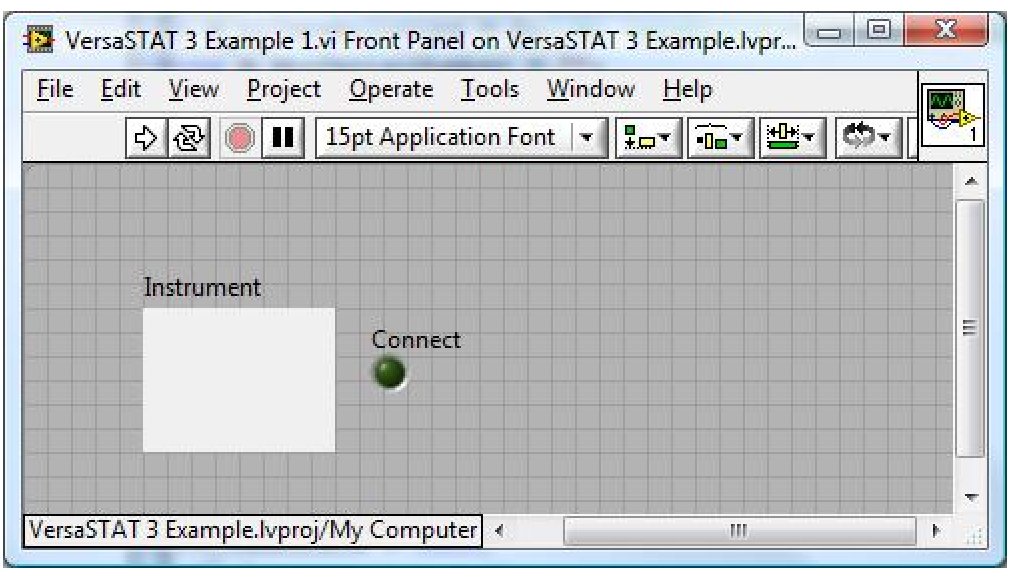


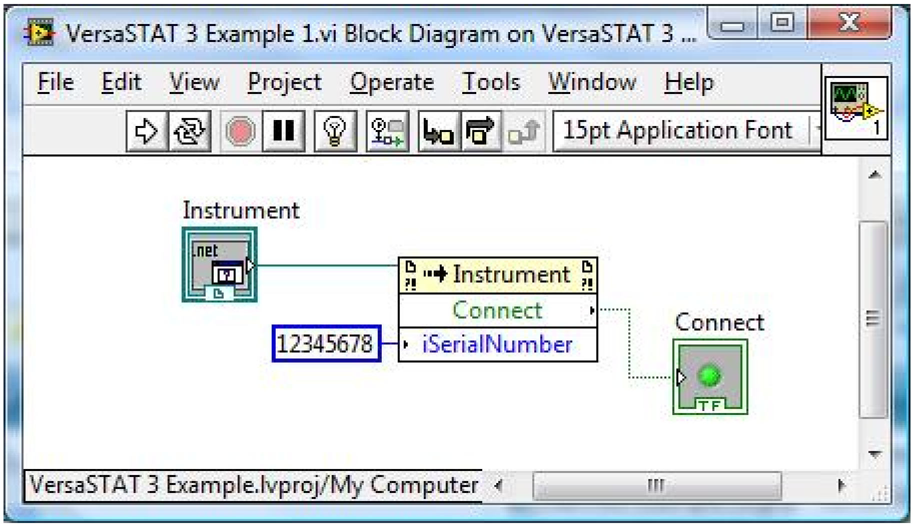
2.2 LabView

All sources to LabView examples are located in the "LabView Example" folder. Example #1 is to show how to add the .NET Control to a .vi. Example #2 shows how to run an experiment where the user defines some of the required parameters via text controls. Example #3 shows how to run "any" experiment, but parameters are entered as a string of text. This example also shows how to use the immediate commands to control the instrument.

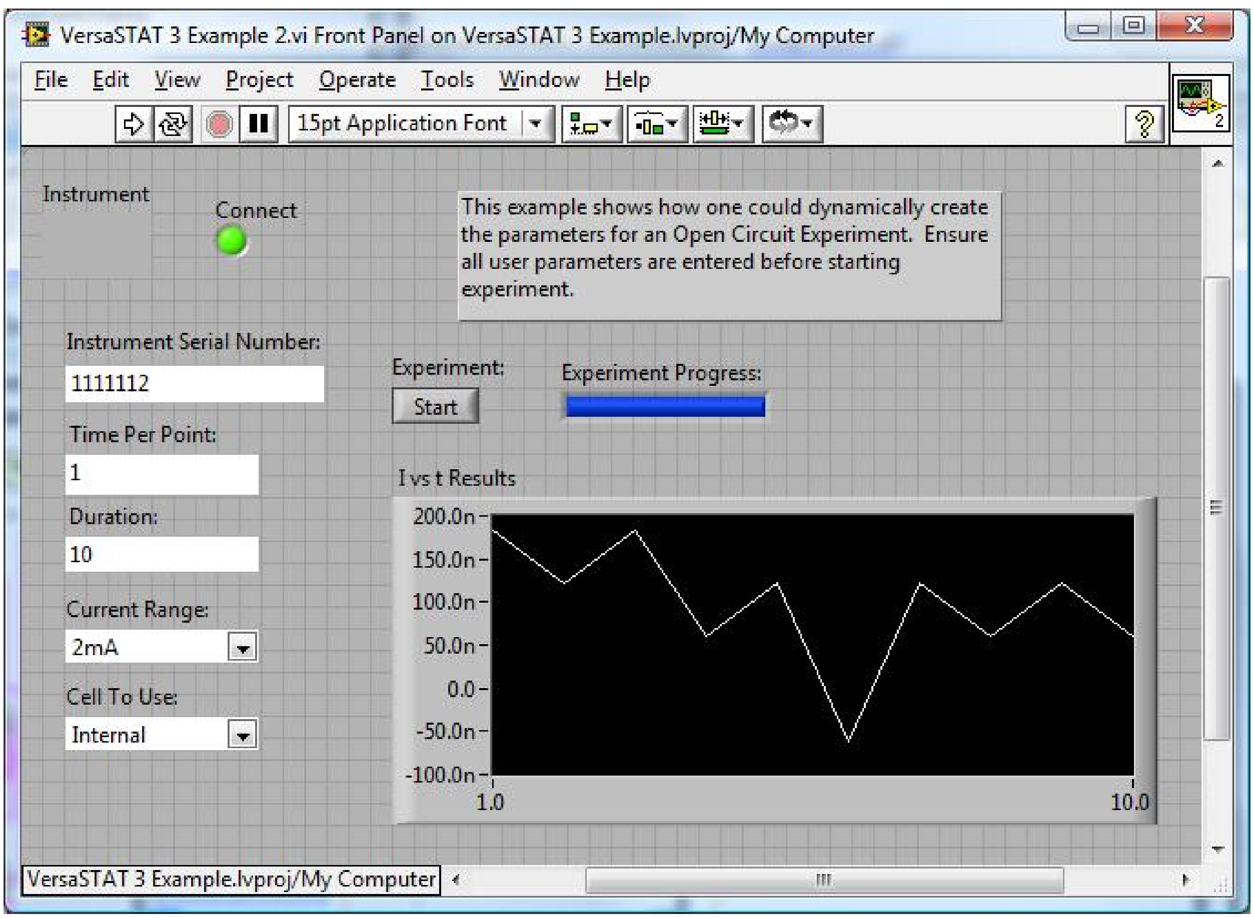
Example #1

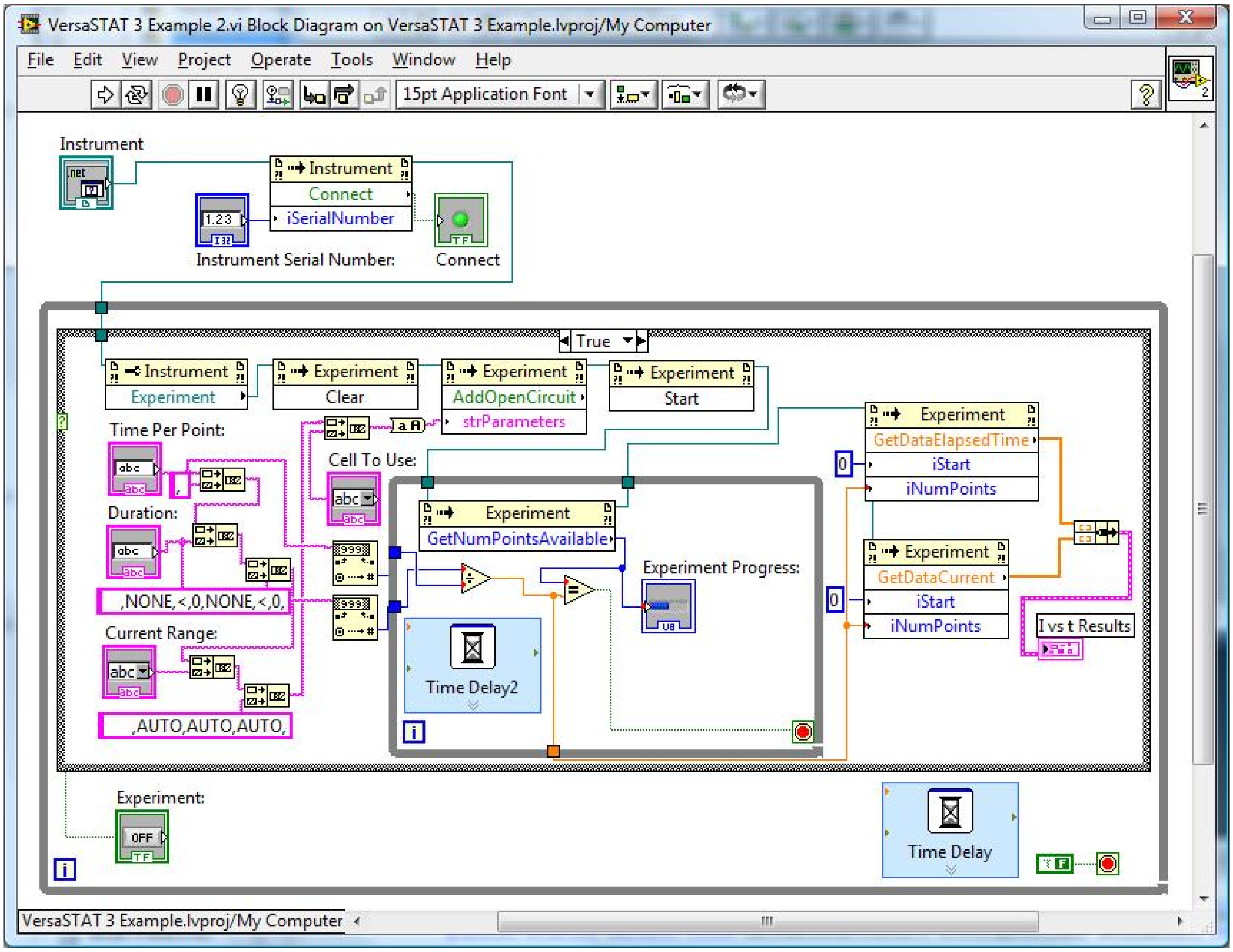
1. Create a new or open a .vi.
2. Add a ".NET Container Control" to the .vi.
3. Right click on the .NET Container and select "Insert .NET Control".
4. Click the "Browse" button.
5. Locate the "VersaSTATControl.dll" file and select it, press OK button.
6. Select "Instrument", press OK button.
7. Open "Block Diagram" window.
8. Right click on "Instrument", select "Create", then "Method for V3Control.Instrument class".
9. Select the "Connect(int32 iSerialNumber)" method.
10. Wire from ".NET Instrument" to the top right (reference) of the method Node.
11. Right click on left side of "iSerialNumber", select "Create", then select "Constant".
12. Enter Serial Number of the Instrument to communicate with.
13. Right click on the right side of "Connect", select "Create", then select "Indicator".
14. Run the .vi, if the Instrument is powered on and attached to the PC, the connect LED should light.




VersaSTAT 3 Example 1.vi Block Diagram on VersaSTAT 3 ...
File Edit View Project Operate Tools Window Help
15pt Application Font
Instrument
Instrument
Connect
12345678
iSerialNumber
Connect
VersaSTAT 3 Example.lvproj/My Computer

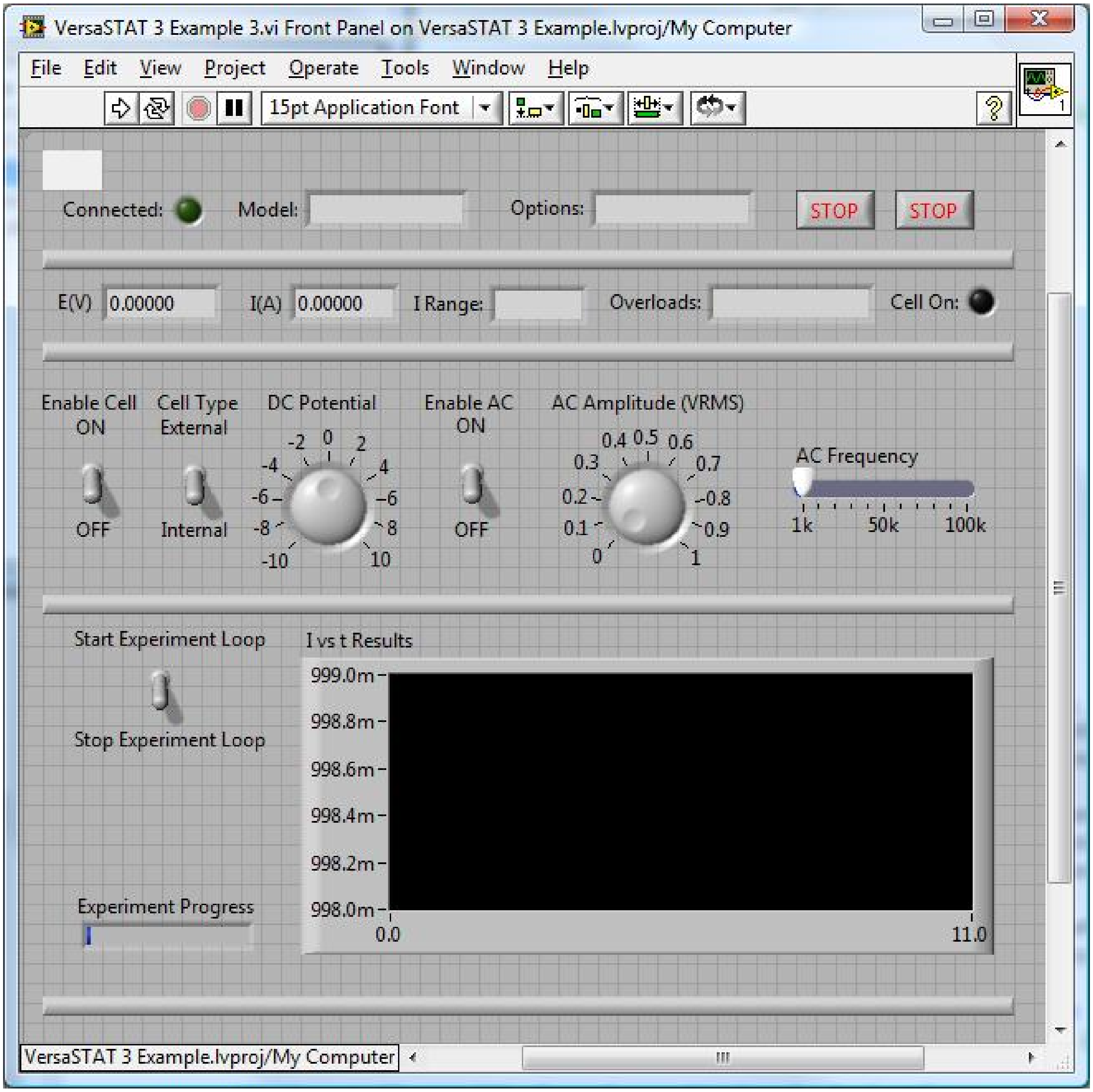
Example #2

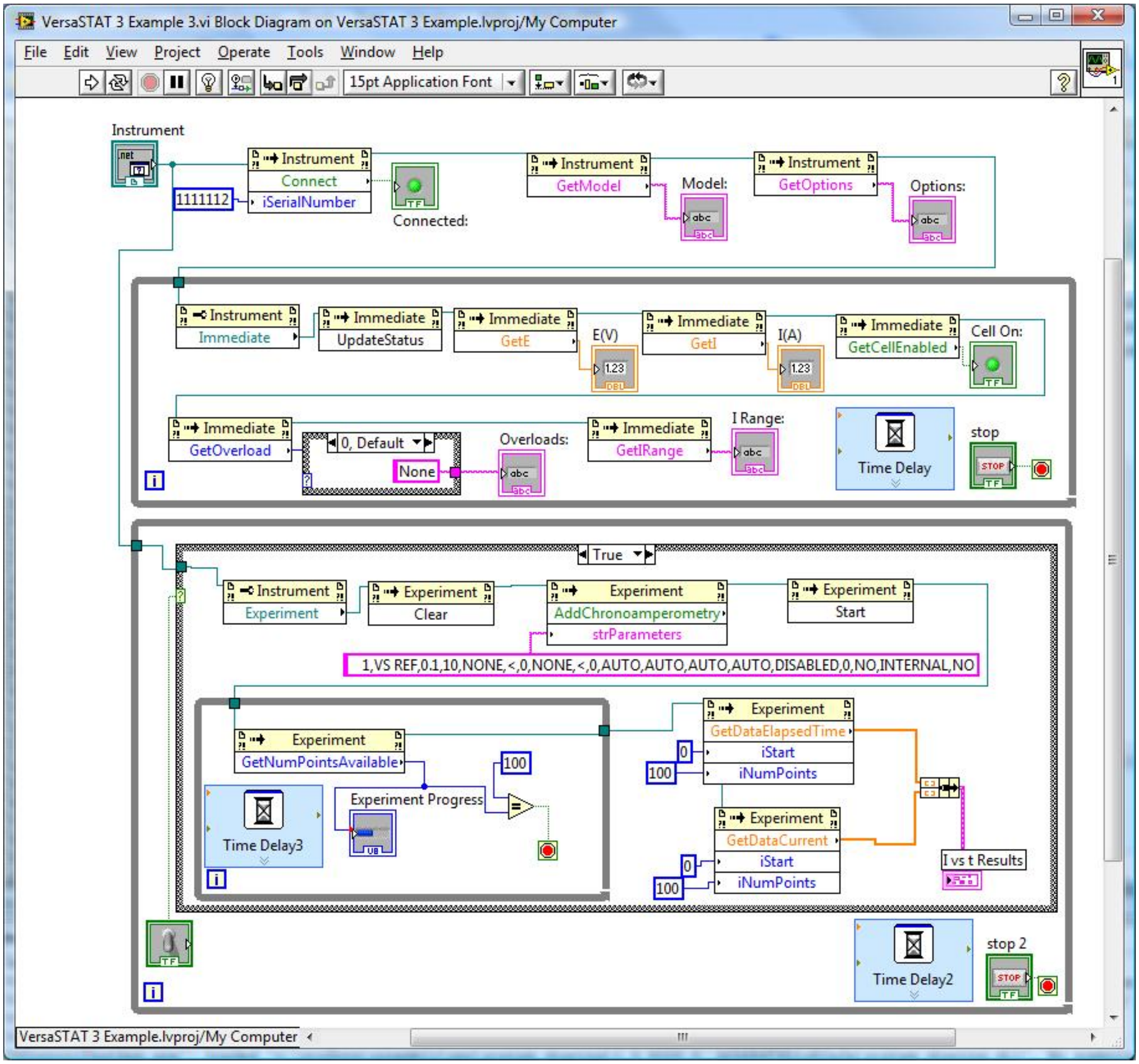

VersaSTAT 3 Example 2.vi Front Panel on VersaSTAT 3 Example.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Instrument
Connect
This example shows how one could dynamically create the parameters for an Open Circuit Experiment. Ensure all user parameters are entered before starting experiment.
Instrument Serial Number:
1111112
Experiment:
Start
Experiment Progress:
Time Per Point:
1
Duration:
10
Current Range:
2mA
Cell To Use:
Internal
I vs t Results
200.0n
150.0n
100.0n
50.0n
0.0
-50.0n
-100.0n
1.0
10.0
VersaSTAT 3 Example.lvproj/My Computer



VersaSTAT 3 Example 2.vi Block Diagram on VersaSTAT 3 Example.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Instrument
Instrument
Connect
iSerialNumber
Instrument Serial Number:
Connect
True
Instrument
Experiment
Experiment
Clear
Experiment
AddOpenCircuit
strParameters
Experiment
Start
Time Per Point:
Duration:
,NONE,<,0,NONE,<,0,
Current Range:
,AUTO,AUTO,AUTO,
Cell To Use:
Experiment
GetNumPointsAvailable
Experiment Progress:
Time Delay2
Experiment
GetDataElapsedTime
iStart
iNumPoints
Experiment
GetDataCurrent
iStart
iNumPoints
I vs t Results
Experiment:
Time Delay
VersaSTAT 3 Example.lvproj/My Computer

Example #3


VersaSTAT 3 Example 3.vi Front Panel on VersaSTAT 3 Example.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Connected:
Model:
Options:
STOP
STOP
E(V) 0.00000
I(A) 0.00000
I Range:
Overloads:
Cell On:
Enable Cell ON
OFF
Cell Type External
Internal
DC Potential
-2 0 2 -4 4 -6 6 -8 8 -10 10
Enable AC ON
OFF
AC Amplitude (VRMS)
0.4 0.5 0.6 0.3 0.7 0.2 0.8 0.1 0.9 0 1
AC Frequency
1k 50k 100k
Start Experiment Loop
Stop Experiment Loop
I vs t Results
999.0m 998.8m 998.6m 998.4m 998.2m 998.0m
0.0 11.0
Experiment Progress
VersaSTAT 3 Example.lvproj/My Computer


VersaSTAT 3 Example 3.vi Block Diagram on VersaSTAT 3 Example.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Instrument
Instrument
Connect
1111112
iSerialNumber
Connected:
Instrument
GetModel
Model:
Instrument
GetOptions
Options:
Instrument
Immediate
Immediate
UpdateStatus
Immediate
GetE
E(V)
Immediate
GetI
I(A)
Immediate
GetCellEnabled
Cell On:
Immediate
GetOverload
0, Default
None
Overloads:
Immediate
GetIRange
I Range:
Time Delay
stop
True
Instrument
Experiment
Experiment
Clear
Experiment
AddChronoamperometry
strParameters
Experiment
Start
1,VS REF,0.1,10,NONE,<,0,NONE,<,0,AUTO,AUTO,AUTO,AUTO,DISABLED,0,NO,INTERNAL,NO
Experiment
GetNumPointsAvailable
100
Experiment Progress
Time Delay3
Experiment
GetDataElapsedTime
0
iStart
100
iNumPoints
Experiment
GetDataCurrent
0
iStart
100
iNumPoints
I vs t Results
Time Delay2
stop 2
VersaSTAT 3 Example.lvproj/My Computer