Control via other software

IviumSoft can be controlled from other software, such as programs in C++, Delphi, VB, Labview. To facilitate this a dll with a library of functions is available: software development driver dll. As an example, the procedure and an example VI is given for Labview.

Software development driver DLL

Ivium supplies a software development driver dll that is standard included with each IviumSoft installation package. This dll supports interaction between IviumSoft and your own program in your choice of language: VB, Delphi, C, Labview, etc. The dll offers stadard commands that are executed in IviumSoft, thus providing easy software programming, while the algorithms, safety catches and settings of IviumSoft remain in place. Your software through the dll communicates with IviumSoft that is running in the background (your software will not communicate directly with the Ivium instrument, but with IviumSoft). This implicates that the IviumSoft-software should always be running in the background when the dll to operate the device.

Advantages:

- Faster development of applications than programming from scratch. IviumSoft takes care of most overhead: communication, error handling, graphic plotting, datastorage etc. You may mix modes, for example first set the device in a desired state with IviumSoft, and let your own program take control from that point.
- More flexible than using a scripting language. You can use all the programming power of the higher programming language of your choice. It is possible to customize data processing, or react on specific results and events.
- Control multiple Ivium devices at the same time, and simultaneously control/read other types of devices: pumps, valves, thermostats, motors, sensors, etc.

Versions:
Over the years of its existance, functionality has been added to the IVIUM_remdriver.dll. From IviumSoft version 2.200 (with firmware version 417) it is possible to check which version of IVIUM_remdriver.dll it corresponds with: in the top menu select "About", it will open a pop-up window. At the bottom left the IviumSoft release version is given, behind it between brackets [..] the IVIUM_remdriver.dll version it corresponds with. The most recent dll is always included in the IviumSoft installation package.

Method:
1) Import the provided dll in your program: **IVIUM_remdriver.dll**
2) Embed the control-functions in your software
3) Start IviumSoft and your own program

The dll allows you to import and execute most basic functions of the Ivium device:

Imported function	Description
GENERAL	
IV_open	Opens the driver
IV_close	Closes the driver
IV_abort	Will abort the running method/measurement
IV_selectdevice(int)	Select device, applicable for multi-device configurations, default=1

***The command IV_selectdevice does not actually select a device but an IviumSoft instance. Multiple devices can only be operated with a separate IviumSoft instance. When you open in sequence for example 3 IviumSoft instances, IV_selectdevice=1 communicates with the first IviumSoft instance, IV_selectdevice=2 communicates with the second IviumSoft instance, etc. IV_connect then instructs the IviumSoft to connect with the first available instrument in the connect list of instruments, i.e. the top one. The list is alphabetical for instrument serial number.
The maximum of simultaneous operated IviumSoft instances is 32.
The command IV_readSN will return the serial number to which the IviumSoft instance is connected.***

IV_getdevicestatus	Returns status of device: -1=no IviumSoft; 0=not connected; 1=available_idle; 2=available_busy; 3=no device available
IV_readSN(*char)	Returns serial number of selected device, empty string if not connected
IV_connect(int)	Connect to selected device, int=1 for connect, int=0 for disconnect; To establish a know situation, using the connect function sets the Automatic E-ranging of the instrument to "off"
IV_version	Returns the version number of the IVIUM_remdriver.dll the active IviumSoft needs
IV_SelectChannel(int)	Sending the integer value communicates with Multichannel control: if not yet active, the [int] number of tabs is automatically opened and the [int] tab becomes active; if Ivium-n-Soft is active already, the [int] tab becomes active. Now the channel/instrument that is connected to this tab can be controlled. If no instrument is connected, the next available instrument in the list can be connected (IV_connect) and controlled.
DIRECT MODE	
IV_getcellstatus(int)	Returns cell status: bit 2=I_ovl, bit 4 =Anin1_ovl, bit 5 = E_ovl, bit 7 = CellOff_button pressed
IV_setconnectionmode(int)	Select configuration, 0=off; 1=EStat4EL(default), 2=EStat2EL, 3=EstatDummy1,4=EStatDummy2,5=EstatDummy3,6=EstatDummy4 7=Istat4EL, 8=Istat2EL, 9=IstatDummy, 10=BiStat4EL, 11=BiStat2EL
IV_setcellon(int)	Set cell on off to close cell relais (0=off;1=on)
IV_setpotential(double)	Set cell potential
IV_setpotentialWE2(double)	Set BiStat offset potential
IV_setcurrent(double)	Set cell current (galvanostatic mode)
IV_getpotential(double)	Returns measured potential
IV_setcurrentrange(int)	Set current range, 0=10A, 1=1A, etc,
IV_setcurrentrangeWE2(int)	Set current range for BiStat, 0=10mA, 1=1mA, etc,
IV_getcurrent(double)	Returns measured current
IV_getcurrentWE2(double)	Returns measured current from WE2 (bipotentiostat)
IV_setfilter(int)	Set filter, for int :0=1MHz, 1=100kHz, 2=10kHz, 3=1kHz, 4=10Hz
IV_setstability(int)	Set stability, for int 0=HighSpeed, 1=Standard, 2=HighStability
IV_bistat_mode(int)	Select mode for BiStat, for int 0=standard, 1=scanning This bistat_mode function also can be used to control the Automatic E-ranging function of the instrument; 0=AutoEranging off; 1=AutoEranging on
IV_setdac(int,double)	Set dac on external port, int=0 for dac1, int=1 for dac2
IV_getadc(int,double)	Returns measured voltage on external ADC port, int=channelnr. 0-7
IV_setmuxchannel(int)	Set channel of multiplexer, int=channelnr. starting from 0(default)
IV_setdigout(int)	Set digital lines on external port, int is bitmask
IV_getdigin(int)	Returns status of digital inputs from external port, int is bitmask
IV_setfrequency(double)	Set ac frequency, double in Hz
IV_setamplitude(double)	Set ac amplitude, double in Volt
IV_getcurrenttrace (int ,double,*double)	Returns a sequence of measured currents at defined samplingrate (npoints, interval, array of double): npoints<=256, interval: 10us to 20ms
IV_getcurrentWE2trace (int ,double,*double)	Returns a sequence of measured WE2 currents at defined samplingrate (npoints, interval, array of double): npoints<=256, interval: 10us to 20ms
IV_getpotentialtrace (int ,double,*double)	Returns a sequence of measured potentials at defined samplingrate (npoints, interval, array of double): npoints<=256, interval: 10us to 20ms
IV_we32setchannel(int)	Select active WE32 channel (chan)
IV_we32setoffset(int,double)	Set WE32 offset (chan,value), value -2 to +2V. Use chan=0 to apply the same offset to all channels.
IV_we32getoffsets(int,*values)	Returns actual WE32 offset values (Nchan,values), with Nchan the number of channels (1..32)
IV_we32readcurrents(*values)	Returns array with 32 WE32 current values, that are measured simultaneously
METHOD MODE	
IV_readmethod(*char)	Loads method procedure from disk, with char as filename
IV_savemethod(*char)	Saves method procedure to disk, with char as filename
IV_startmethod(*char)	Start method procedure. If char is empty then presently loaded procedure is used, else the procedure is loaded from disk.
IV_abort	Abort the ongoing method procedure
IV_savedata(*char)	Saves actual result data to disk, with char as filename
IV_setmethodparameter	Modify method parameter, with char1=parameter_name,

(*char1,*char2)	char2=new value
IV_Ndatapoints(int)	Returns actual available number of datapoints: indicates progress during a run
IV_getdata(int,d1,d2,d3)	Read datapoint with index int, returns 3 doubles (d1/d2/d3) that represent measured values depending on the used technique, for example LSV/CV methods return (E/I/0) Transient methods return (time/I,E/0), Impedance methods return (Z1,Z2,freq) etc.
IV_getdatafromline (int,int2,d1,d2,d3)	Same as IV_readdata, but with the additional int2 parameter which specifies the scannr. This function will allow reading data from non-selected (previous) scans.

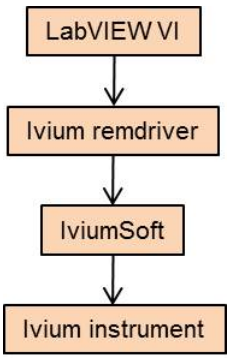
Programming considerations:

- All imported functions return an integer as result (32 bits signed number), 0=successfully executed, -1= no device, 1=illegal command, 2=argument out of range. Note that IV_getdevicestatus return codes are different.
- Arguments are passed by reference.
- Pchar= zero-terminated string; int=32 bit signed integer; double=8 byte floating point number
- Current is expressed in Ampere, potential in Volt, time in s, and frequency in Hz
- The driver must be opened with the IV_open function before the control functions can be used.
- When the driver is opened, it automatically connects the first connected device. For single-device users, nothing needs to be done to select it. For multi-device configurations, use the IV_selectdevice() command to select each different device.
- After starting a method, IV_getdevicestatus will indicate whether a scan is ready. During a scan, progress can be monitored with the IV_Ndatapoints function.
- When reading datapoints with the IV_readdata command during an ongoing scan, be sure to check whether data is available with the IV_Ndatapoints function, before attempting to access the data.
- The IV_setmethodparameter(pchar1,pchar2) function will change method-parameters of the currently loaded procedure. If subsequently a scan is started, the new values will be used. It requires 2 arguments, the parameter name and the parameter value:
    - Parametername: this must correspond exactly to the spelling on the method-tabsheet.
    - Parametervalue: textual expression of the parameter value. The format of the supplied value must correspond with the type of the selected parameter name. If the selected parameter is a checkbox, a value of 'true' will correspond to the checked condition, anything else will uncheck the box. Numerical text strings must be of the correct format.
    - When the technique should be modified, first set the Method, thereafter the Technique. For example when selecting Standard Cyclic Voltammetry:
        - setmethodparameter('Method','CyclicVoltammetry')
        - setmethodparameter('Technique','Standard')
    - If wrong or unavailable Parameter names are selected, or when unavailable parametervalues are entered, the commands are ignored without an error message. When a parametervalue with improper format is supplied, the command is ignored and an error message is shown. Please note that the parameter availability depends on the chosen Method and Technique.
    - In principle only top level parameters can be changed, that is: only parameters that are visible in the method tab should be changed; parameters behind a check box or in a pop-up should not be changed.

Labview interfacing

1. Order of communication hierarchy

For optimum results when using Labview to control an Ivium device it is very important to realise the order of hierarchy for communication between the various software and hardware. In this case Labview (the VI) communicates with the Ivium remdriver, which in turn communicates with the IviumSoft-software. Only the IviumSoft communicates directly with the Ivium device. This implicates that the IviumSoft-software should always be running in the background when using a Labview VI to operate the device.
The IviumSoft can be minimised when operating Labview.

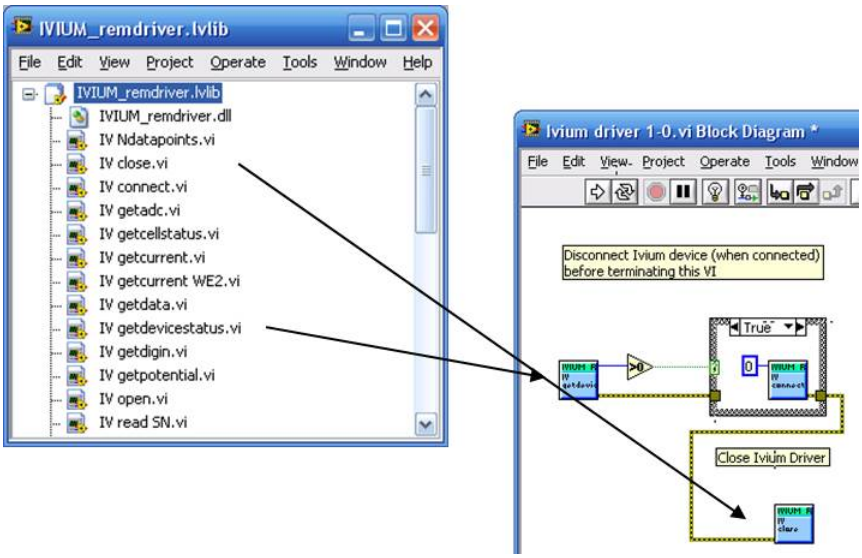


2. Constructing the LabVIEW vi library

Import the Ivium remdriver (dll) into LabVIEW. LabVIEW will automatically create a library with a vi for each available function in the Ivium remdriver. A list of available functions can be found in the Software development driver DLL.

3. Building you LabVIEW vi

To use the functions in Labview open the ' IVIUM_remdriver.lvlib' (in the Labview directory) in the getting started window of Labview. Then drag the desired function from the lvlib onto the Block Diagram of the VI that you are building.



When a VI is build to control an Ivium device make sure that the driver is opened before operating the device and closed again before the VI is stopped.

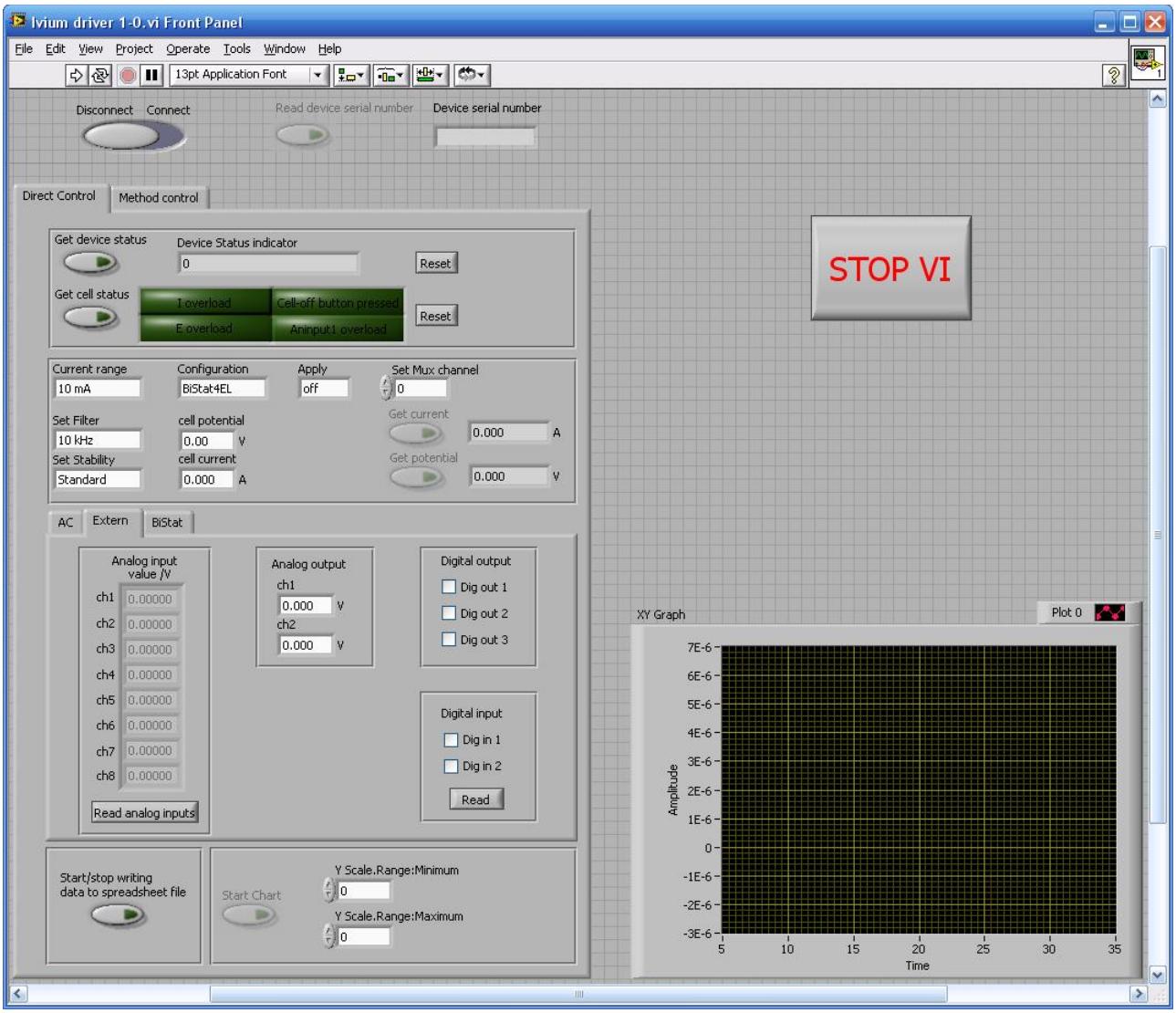
Boundary conditions
Some of the settings of the Ivium device that can be controlled via Labview change an actual setting in the Ivium device, like for example the current range. When one (or some) of these settings are changed outside Labview, i.e. in the IviumSoft or by the device itself when for example using the AutoCR setting in a method, this change of setting is NOT read back into Labview. That implicates that the value in the User Interface in the Front Panel may not be the actual setting anymore.

When using the method mode a number of strings (file names and locations) are communicated to the IviumSoft software. These can only be communicated when the device is CONNECTED. If the device is not connected it will result in an error.

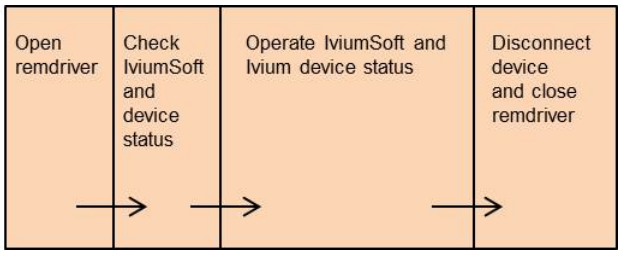
NOTE: The IviumSoft-software has many conditions and safeties incorporated. When using Labview to control an Ivium device the programmer her/himself will be responsible for supplying valid input parameters.

4. Example

As an example how Labview may be used to control an Ivium device, a Labview VI is build. This example is available in your ..\IviumStat\Labview folder. In this example mostfunctions that can be controlled through Labview are included. This example has been set up in such a way that the user interface resembles the user interface of the IviumSoft. Note that the example is build using Labview 8.2, if you are using a later version of LabVIEW, not all functions may work and you will need to contruct your own vi library.

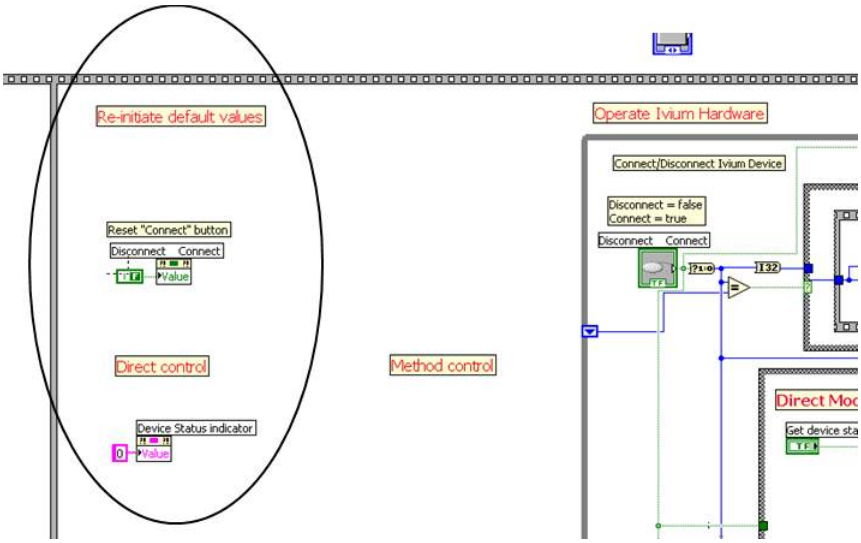


Set up of Ivium Driver 1.0
The Ivium Driver 1.0 has been set up using a flat sequence to ensure that the order of events is correct.

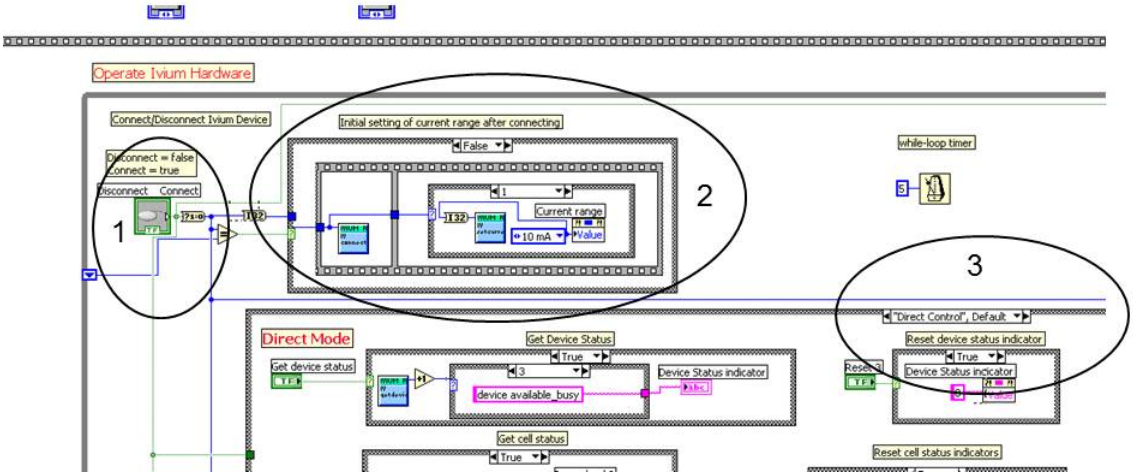


- The first operation is to open the Ivium_remdriver.dll so the IviumSoft and device can be operated.
- Next a check is performed to see whether the IviumSoft is running. If not the VI is stopped. If IviumSoft is running a check is performed to see if the device is connect, and if so it will be disconnected.
- Third sequence is the actual operation of the Ivium device, either in direct-mode or in method-mode.
- Finally, upon closing the VI, the Ivium device is disconnected and the Ivium_remdriver.dll is closed.

Operating sequence of Ivium Driver 1.0: Block Diagram set-up
To ensure the continuance of operating the Ivium device the actual running-sequence of the VI is incorporated in a while-loop. However, before the while-loop commences an initiation of the default values is executed for all the controls for the direct-mode operation. This has been done to ensure that the values shown in the user interface or Front Panel are the actual values. After this has been done the while-loop starts.



First in the while loop (refer to Block Diagram) is the possibility of connecting with the Ivium device (1). Upon connecting, a flat sequence is carried out (2) connecting the device and subsequently setting the current range, again to ensure that the shown value is the actal one.

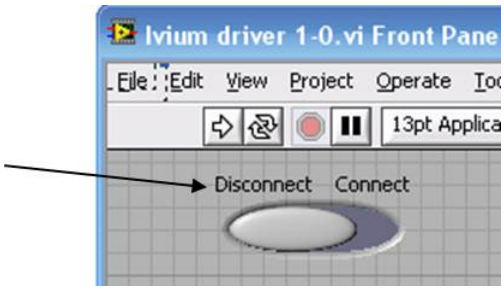


The rest of the while-loop is reserved for a tab controlled case structure (3). This case structure operates depending on the choice of 'direct' or 'method' mode.

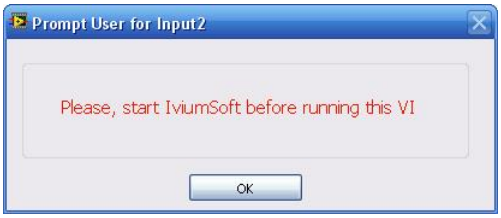
The 'direct' case includes all operating parameters, as well as the tab controlled case structure for the AC/Extern/BiStat controls. It also includes the writing-data-to-file operation and the simple chart.

The 'method' case includes all the operating parameters for reading, starting, changing and saving experimental methods.

Operating the Ivium Driver 1.0
Operating the Ivium Driver 1.0 mostly resembles the IviumSoft software. First start the VI from Labview. Clicking the 'Disconnect - Connect' button at the top of the user interface above the tab control connects the Ivium device.

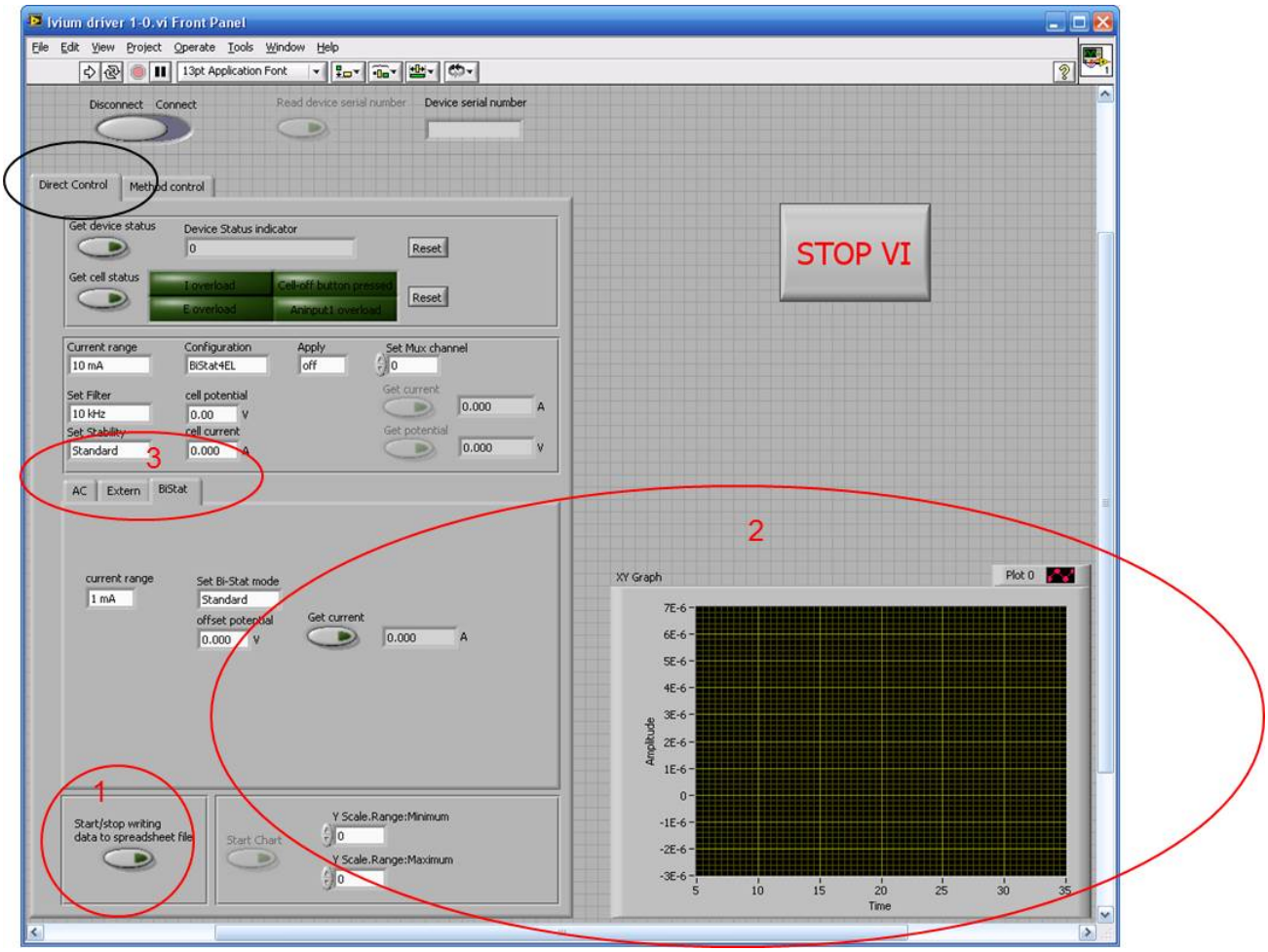


If the device is not connected, none of the direct operating controls or the method controls, work. After connecting, all functions can be operated. This includes the reading of the device serial number. If the IviumSoft is not running a pop-up window will show mentioning this. Upon clicking the 'OK'-button the Ivium Driver 1.0-VI will stop running.



When connected the Ivium device can be simply operated by using the controls on the various tab pages.

Direct Mode
In the 'direct' mode tab page the device and cell status can be called and all parameters of the Ivium device can be controlled directly. For some of these parameters a property node is called to enable/disable the controls when relevant. This has been done to eliminate the possibility of sending illegal or conflicting commands that might result in an error and subsequent ceasing of Labview.

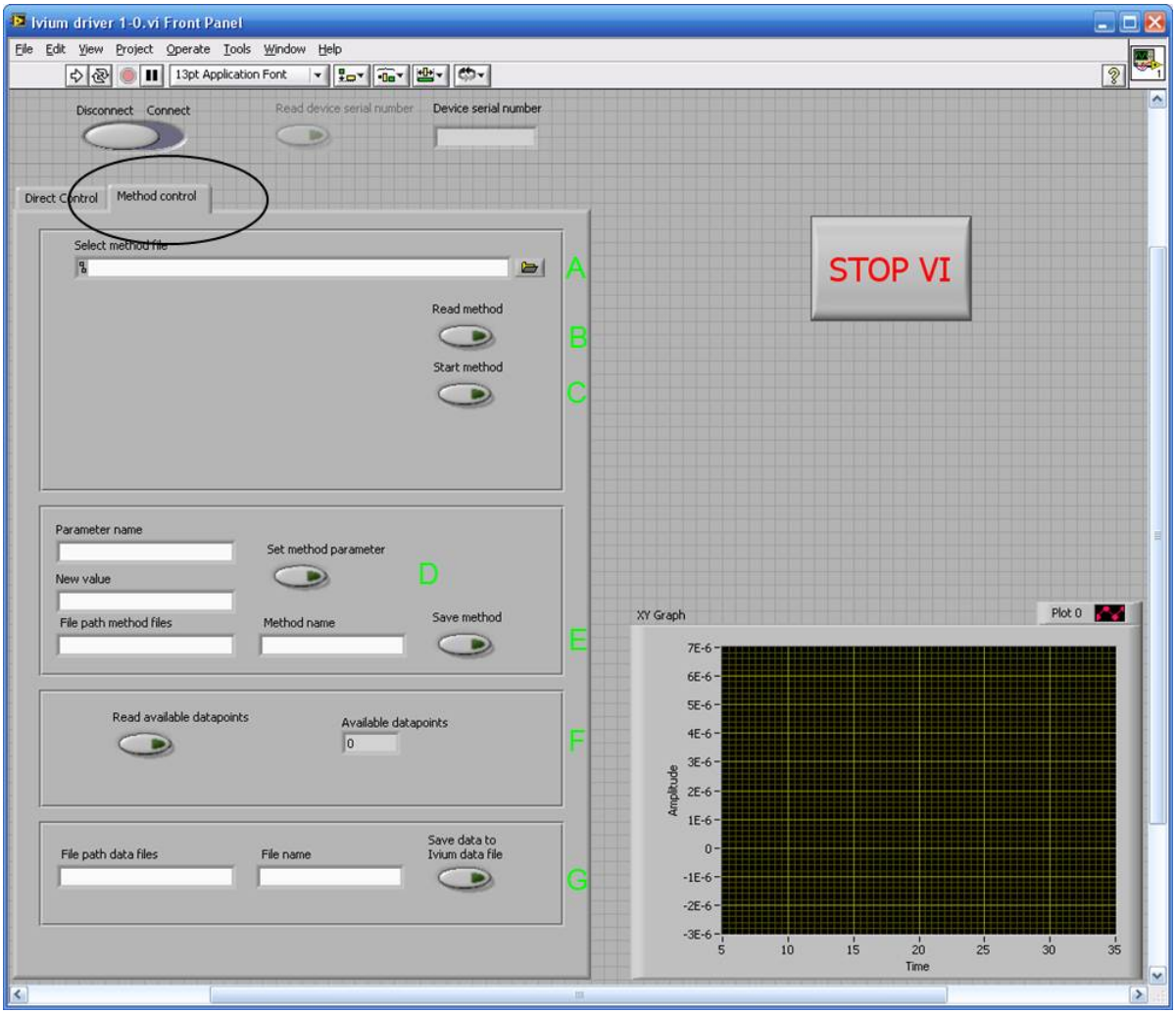


In the direct mode, as an example, an operation for writing data to spread-sheet file (1) and a simple chart (2) have been incorporated. The chart plots data from the data file and thus only operates when the data is being written to file. The name of the data file is automatically genrated as 'test+date/time'. This filename can be altered in the Block Diagram.

The 'direct' mode also shows an extra number of tab pages: AC/Extern/BiStat (3).

- The 'AC' page allows to control a simple AC experiment.
- The 'Extern' page allows to read and control all external ports.
- The 'BiStat' page only shows when the configuration 'BiStat4EL' or 'BiStat2EL' has been chosen. It then allows BiStat control.

Method mode
In the 'method' mode tab page a method can be selected (A). This method can then be read into the Ivium software and the device (B). Then the method can be started (C). When 'start method'is clicked and no method was read into the IviumSoft, the method currently selected in the IviumSoft is started.



Furthermore, separate parameters of method can be changed (D) and this new method can be saved (E). Care should be taken that the entered method-parameters to be changed are spelled exactly as in the IviumSoft, otherwise it will not register.

During running of a method the number of available data points can be requested (F). In Labview there is no possibility to abort a method once started.

Finally, upon completion of executing a method the then in the IviumSoft available data can be stored as a Ivium data file (.idf) (G).