Technical Note T3

Software Development Driver

Software development driver: **Control the instrument with your own software, apply process control and automation.**

With IviumSoft Version 1.1+, you can develop your own software to control the IviumStat and CompactStat instruments. Developing programs is easy, because IviumSoft will do all the hard work. The supplied driver will allow you to integrate the functionality in your own program. You can still use the convenience of the IviumSoft software and program your specific tasks with a few program lines in the desired language: VB, Delphi, C, etc.

Advantages:

- Faster development of applications than programming from scratch. IviumSoft takes care of most overhead: communication, error handling, graphic plotting, data storage, 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 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.

Method:

- Import the provided DLL in your program: IVIUM_remdriver.dll
- Embed the control functions in your software
- Start IviumSoft and your own program

Functions:

The DLL allows you to import and execute most basic functions of the Ivium device. A list of available functions is given in the table below.

Imported Function	Description
GENERAL	
IV_open	Opens the driver
IV_close	Closes the driver
IV_selectdevice (int)	Select device, applicable for multi-device configurations, default=1
IV_getdevicestatus	Returns status of device: -1=no IviumSoft; 0=not connected; 1=available_idle; 2=available_busy
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
IV_version	Returns the version number of the IVIUM_remdriver.dll the active IviumSoft needs

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, bit 8= cell on
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 *Note that the above is for single instruments, but when an IviumBoost1040 is connected, the number of available connect states is less and the code definition changes to: Select configuration, 0=off; 1=EStat4EL(default), 2=IStat4EL Therefore in the presence of a 40A Booster, to switch to galvanostatic operation, code 2 should be sent instead of 7. If code 7 would be sent, it is ignored.*
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
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 20msec
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 20msec
IV_getpotentialtrace (int ,double,*double)	Returns a sequence of measured potentials at defined samplingrate (npoints, interval, array of double): npoints<=256, interval: 10us to 20msec
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 (*char1,*char2)	Modify method parameter, with char1=parameter_name, char2=new value
IV_Ndatapoints (int)	Returns actual available number of datapoints: indicates progress during a run
IV_getata (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 bit 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 Amperes, potential in Volts, time in seconds, and frequency in Hertz
- 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_getdata 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 parametername and the parametervalue:
  - Parametername: this must correspond exactly to the spelling on the method-tabsheet.
  - Parametervalue: textual expression of the parametervalue. The format of the supplied value must correspond with the type of the selected parametername. 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 is modified, first set the Method, then 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.

IVIUM
TECHNOLOGIES

www.ivium.com