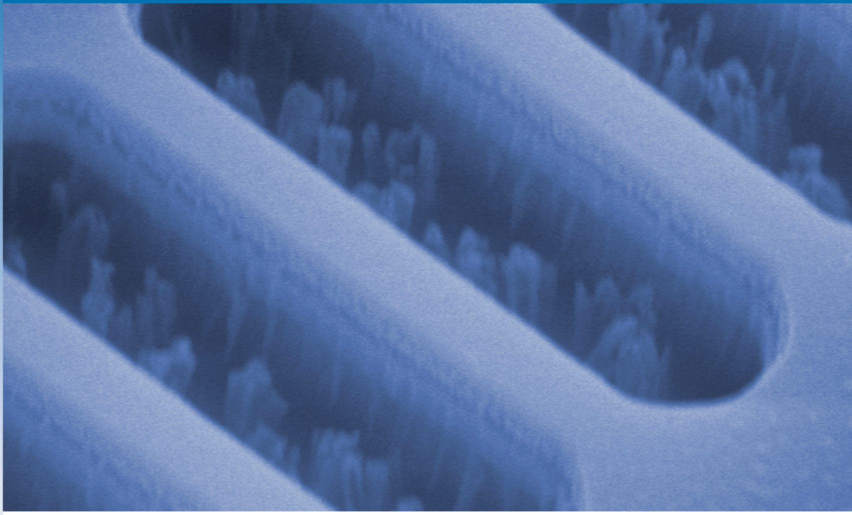

QUANTAX API

Basic interface functions for Bruker
Esprit EDS software.

● **Reference Manual**

EDS

Innovation with Integrity

Bruker Nano GmbH
Schwarzschildstrasse 12
12489 Berlin

Tel.	+49 (30) 670 990-0
Fax	+49 (30) 670 990-30
E-mail	service-berlin@bruker-nano.de
Internet	www.bruker.com

Order no. DOC-M82-EXX019 V4
© 2011 Bruker Nano GmbH, Berlin, Germany.

Contents

1 Overview

This API is intended for control of basic acquisition and quantification functions of the Bruker Esprit EDS software.

User software can use functions to control Bruker spectrometry and imaging hardware, to start and stop spectrum and image acquisition and to quantify acquired or loaded spectra.

2 Installation

2.1 Included Files

▪ description.rtf	- this document
▪ rtifcclient.dll	- interface library to QUANTAX or QM100
▪ \SampleProgram\InterfaceTest.*	- project files for sample program
▪ \SampleProgram\mainunit.*	- source files for sample program

2.2 Installation

Copy folder 'SampleProgram' anywhere to your local disk. To recompile the sample program Delphi 5 is recommended. To run the sample it is necessary that a QuanTax or QM100 client is running on the same machine and a copy of the file ‘rtifcclient.dll’ is in the same directory as the sample program. The same goes for the final application.

2.3 Sample program

It consists of an executable program ‘interfacetest.exe’ and the source code ‘mainunit.pas’. The program links the interface functions of rtifcclient.dll statically. This means that rtifcclient.dll must be located in the same directory as the sample program or the final application program respective. The sample program comprises only a limited selection of possible interface functions.

3 Conventions and Definitions

Calling convention is stdcall. String parameters are always referenced by pointers (type PChar). Input parameter may refer to literals of buffers. Empty input parameter are to be notified by nil. In case of answer strings buffers with enough room for the complete answer string are to be provided (buffer overflow are tagged by an error code). Some strings contain multiple lines which are separated by <CRLF>= #0D,#0A. The terminator of the last line is optional. String contents may be method specific and defined elsewhere.

QuanTax/QM100 is a Client/Server system. ‘Client’ refers to the part of the system that controls normally the user screen and runs on the local computer; ‘Server’ is the program that controls the hardware. The server can control 1 to 4 spectrometers and possibly an imaging system. Upon start-up a client connects to a chosen server.

If more clients connect to the same server (computer) only the first client can access hardware.

Any running client application is assigned to a user, however one user can start more then one client application, even when connected to the same server.

The user name must be one of a list of pre-assigned user for the server to be connected. The application program can start own clients or connect to a client already running on that computer.Starting an application requires a valid password. References to a connection takes place always via an identification code (CID) delivered by the OpenClient function.

4 Type declarations

4.1 Standard types

longint	:	signed 4 byte
longword	:	unsigned 4 byte
double	:	8 byte floating point
boolean	:	1 byte boolean (0=false)
PChar	:	Pointer to first char of a zero terminated string

5 Error codes

Functions normally return with an error code as function result. Error code = 0 always mean function executed successfully. Error codes may be function specific and meaningful only to the programmer. However, some general error codes are commonly defined:

ERROR IN_EXECUTION	=	-1
ERROR WRONG_PARAMETER (execution)	=	-2
ERROR SPECTRUM_BUFFER_EMPTY	=	-3
ERROR PARAMETER_MISSED	=	-11
ERROR WRONG_PARAMETER (interface)	=	-101
ERROR FILE_NOT_EXIST	=	-102
ERROR NO_CONNECTION	=	-103
ERROR NO_ANSWER	=	-104

6 Functions

6.1 Start / Stop the Client

function	**QueryServers**	**// Query list assigned servers**
	(pServerList:	PChar;
	BufSize:	longint): longint;

function	**QueryUser**	**// Query list of active clients**
	(pServer	PChar;
	pUserList:	PChar;
	BufSize:	longint): longint;

function	**QueryInfo**	**// Delivers infos about a client/server instance**
	(CID:	longword;
	pInfo:	PChar;
	BufSize:	longint): longint;

function	**OpenClient**	**// start a new client application**
	(pServer:	PChar;
	pUserName:	PChar;
	pPassword:	PChar;
	StartNew:	boolean;
	GUI:	Boolean;
	var CID:	longword): longint;

function	**CloseClient**	**// Close a client application**
	(CID:	longword): longint;

function	**CheckConnection**	**// checks interface and delivers error code**
	(CID:	longword): longint;

Parameters

pServerList:	Pointer to string buffer for list of names of all assigned QuanTax/QM100 servers
BufSize:	Size of string buffer for regarding list, must be sufficient for complete list
pServer:	Pointer to a server name (if empty the local/default server is referenced)
pUserList:	Pointer to string buffer for list of names of all active clients for given server
pUserName:	Pointer to User name assigned for referenced server (may be empty)
pPassword:	Pointer to Password of the named user (ignored if user is already active)
StartNew:	If true function starts always a new client instance
GUI:	When true client screen is shown, otherwise hidden (only valid at first start of client)
CID:	Identification code (handle) for a actual server/client instance
Function results:	Error code (0 = no error)

ServerList

Each line contains the name of an assigned server for this workstation. The server may be local or connected via LAN/WAN. In case the local server is to be connected or only one server is assigned the server list need not be fetched, since this server can be referred to by an empty server name.

Example:

```
Localmachine<CRLF>
Room AAA<CRLF>
Room BBB<0CRLF>
```

UserList
Each line contains the name of a user that is connected to the given server. The first entry is the main user that is in control of the hardware. Multiple occurrence of a name is possible.

Example:

```
Smith<CRLF>
Jonny<CRLF>
Smith<CRLF>
Guest<CRLF>
```

Info
Contains available infos (server name, user name, access code etc.) for referenced client instance.

Example:

```
server=Localmachine <CRLF>
user=smith<CRLF>
hardwareaccess=true <CRLF>
<CRLF>
```

Remarks
Prior to using the interface functions a client connection is to be opened (OpenClient). This means a new client instance is started or an existing client is connected. For further reference OpenClient delivers an identification code (CID).

With QueryServers on can fetch a list of available servers for that computer. QueryUser delivers a list of names of users that have opened a client with connection to the named server. Independently thereof, the default server (or only server assigned) can be referenced by an empty server name (nil). Likewise the main user on a given server (which has control of the hardware) can be referred to by a empty user name, if already active.

When a user name is given with OpenClient, one can decide if in case this user is already active a second client instance is to be started (StartNew=true) or the existing client is merely to be connected (StartNew=false). If no user with the given name is active always a new client is started. A valid password is only required if a new client is started. In this case one can decide if the client screen shall be visible (GUI=true) or invisible (GUI=false).

6.2 Acquisition Control

function	**StartSpectrumRealTimeMeasurement**	**//start spectra accumulation with real time**
	(CID:	Longword; *preselection*
	Device:	longint;
	RealTime:	Longword): longint;

function	**StartSpectrumLifeTimeMeasurement**	**//start spectra accumulation with live time**
	(CID:	longword; *preselection*
	Device:	longint;
	LifeTime:	Longword): longint;

function	**StopSpectrumMeasurement**	**//stop spectra accumulation immediately**
	(CID:	longword;
	Device:	longint): longint;

function	**GetSpectrumMeasureState**	**//query accumulation state and pulse rate**
	(CID:	longword;
	Device:	longint;
	var Running:	boolean;
	var State:	double;
	var PulseRate:	double): longint;

function	**ReadSpectrum**	**//read spectrum from spectrometer into buffer**
	(CID:	longword;
	Device:	longint):longint;

Parameters

CID:	Identification code (handle) for server/client to be addressed
Device:	Index of spectrometer (1 for first/only one of a given server)
RealTime:	real time preselection in ms (0 : infinite measurement time)
LifeTime:	life time preselection in ms
Running:	Spectra accumulation running or stopped
State:	State of acquisition in % (100 % means ready)
PulseRate:	Input pulse rate at spectrometer
Function results:	Error code (0 = no error)

Remarks
In case more than one spectrometer are connected to one server they are discriminated by the Device parameter (Note: Device does not index different servers, see ConnectClient). Starting a measurement clears the spectrometer (MCA) and starts spectra accumulation. Accumulation is terminated after preselected time (real time or live time). Acquisition can also be stopped by command at any time. Spectrum transfer to buffer in server computer can be initiated independently of running or stopped accumulation. If multiple spectrometers exists, they can be operated independently and are assigned to separate buffers.

6.3 Spectrum Transfer

function	**GetSpectrum**	**//read spectrum from buffer into application prog.**
	(CID:	longword;
	Buffer:	longint;
	SpectrumBuf:	PRTSpectrumHeaderRec;
	BufSize:	longint): longint;

function	**LoadSpectrum**	**//read spectrum from file into buffer**
	(CID:	longword;
	pFileName:	PChar): longint;

function	**SaveSpectrum**	**//save spectrum from buffer to file**
	(CID:	longword;
	Buffer:	longint;
	pFileName:	PChar): longint;

function	**ShowSpectrum**	**//shows spectrum on GUI**
	(CID:	longword;
	Buffer:	longint;
	pName:	PChar): longint;

Parameters

CID:	Identification code (handle) for server/client to be addressed
Buffer:	Index of buffer in server (equals device index, buffer 0 reserved for load function)
SpectrumBuf:	Pointer to data buffer for transmitted spectrum
BufSize:	Size of actual data buffer. Must be sufficient to hold complete spectrum and header (>= 16 465 byte).
pFileName:	Pointer to complete path, filename and extension, referring to local or mapped disks
Function results:	Error code (0 = no error)

Spectrum Header

```
TRTSpectrumHeaderRec = packed record
     Identifier:         char[25];           // Bruker AXS XRay spectrum'
     Version:            longint;            // Version information
     Size:               longint;            // Size of header in byte
     DateTime:           double;             // Delphi 5.0 version of date and time
     ChannelCount:       longint;            // number of channels
     ChannelOffset:      longint;            // Index of first channel (0 or 1)
     CalibrationAbs:     double;             // Energy of first channel
     CalibrationLin:     double;             // Energy step in keV per channel
     SigmaAbs:           double;             // Energy resolution at 0 keV (sigma2)
     SigmaLin:           double;             // Energy resolution slope (sigma2/keV)
     end;

PRTSpectrumHeaderRec = ^TRTSpectrumHeaderRec;
```

Remarks

For each spectrometer a separate buffer is assigned. All buffer contents can be saved, evaluated, or transferred to the application program. For LoadSpectrum a separate buffer is assigned (Index=0). File format is determined from file extension (e.g. '.spx' for QuanTax/QM100 standard format). Show spectrum has no effect if no GUI is shown. Name is optional.

6.4 Spectrum Evaluation

function	**GetQuantificationMethods**	**// query list of available evaluation methods**
	(CID:	longword;
	AutomaticOnly:	boolean;

	pMethods:	**PChar;**
	BufSize:	longint): longint;

function	**QuantifySpectrum**	**// process spectrum from buffer**
	(CID:	longword;
	Buffer:	longint;
	pMethodName:	PChar:
	pParams:	PChar;
	pResultBuf:	PChar;
	BufSize:	longint): longint;

Parameters

CID:	Identification code (handle) for server/client to be addressed
AutomaticOnly	If true, only methods that completes without user interaction are listed
pMethods	Pointer to string buffer for list of available methodes
Buffer:	Index of buffer in server (equals device index, buffer 0 reserved for load function)
pMethodName:	Pointer to name of a available predefined quantification method
pParams:	Pointer to string of parameters, string must contain CRLF separated parameters in
pResutBuf:	Pointer to string buffer for evaluation results
BufSize:	Size of method or result string buffer. Must be sufficient to hold complete string
Function results:	Error code (0 = no error)

Params

Example:

```
     ResultType=nettocounts<CRLF>          // Net counts as results requested
     ResultType=quantification<CRLF>       // additionally quantification results requested
     ParamType= elements<CRLF>             // ROIs defined by element names
          Element=Cu<CRLF>                 // list of elements of interest
          Element=Pb<CRLF>
               Element=Sn<CRLF>
```

Results

Example:

```
     Netto=Sn,L-Serie,5119<CRLF>           // Results of net count estimation
     Netto=Pb,M-Serie,2036<CRLF>
     Netto=Cu,K-Serie,8762<CRLF>

     Quant=Sn,L-Serie,22<CRLF>             // Results of quantification
     Quant=Pb,M-Serie,9<CRLF>
     Quant=Cu,K-Serie,67<CRLF>
```

6.5 Hardware Profiles

function	**GetHardwareProfiles**	**// query list of available settings**
	(CID:	longword;
	pProfiles:	PChar;
	BufSize:	longint): longint;

function	**SetHardwareProfile**	**// activate a predefined setting**
	(CID:	longword;
	pProfile:	PChar): longint;

Parameters

CID:	Identification code (handle) for server/client to be addressed
pProfiles:	Pointer to string buffer for list of names of available hardware profiles
BufSize:	Size of names string buffer. Must be sufficient to hold complete list.
pProfile:	Name of hardware profile to be put into effect
Function results:	Error code (0 = no error)

Remarks

Hardware settings cannot be directly controlled from the application program via QuanTax/QM100. However, any one of a number of predefined and stored profiles can be put into effect. Profiles were created by means of the QuanTax/QM100 user screen. A hardware profile comprises relevant settings for all connected spectrometer.

6.6 Direct Hardware Access

function	**SendRCLCommand**	
	(CID:	longword;
	Device:	longint;
	pCommand:	Pchar;
	pAnswer:	PChar;
	BufSize:	longint): longint;

Parameters

CID:	Identification code (handle) for server/client to be addressed
Device:	Index of spectrometer (1 for first/only one of a given server)
pCommand	Pointer to command string
pAnswer	Pointer to answer string buffer
BufSize:	Size of answer string buffer. Must be sufficient to hold complete string.
Function results:	Error code (0 = no error)

Remarks
Hardware commands are bypassed by QuanTax/QM100 directly to the hardware driver. Syntax is defined by the hardware protocol of the regarding device. At current state, most devices are compatible to the RCL 2.2 protocol (defined in the “Programmers Manual RCL2.2”). Commands are transferred literally, but without the line delimiter (CR).

Note: Hardware commands that respond with binary data ($SS, $SR, $SD) or send continuous data (e.g. $SU) are not supported by this function.