HAMAMATSU
PHOTONICS DEUTSCHLAND GMBH

HiPic/HPD-TA RemoteEx Programmers Handbook

HiPic/HPD-TA Version 8.3.0 or later

WT, HPD, Document Date 25.06.2008

Table of content

Introduction

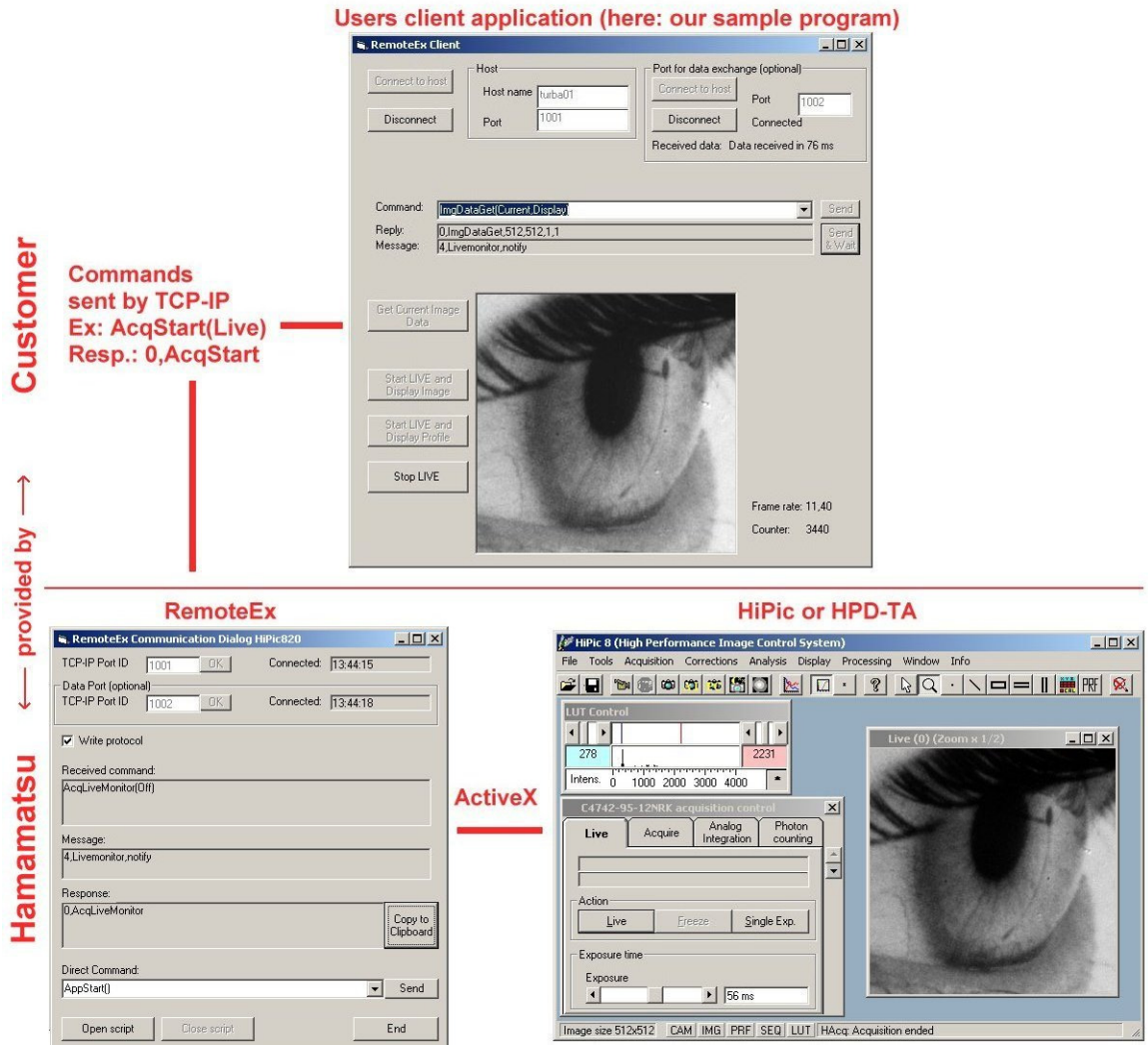
About this manual

Normal explanation text is written in the Font “Times New Roman”.
Commands and responses are written in “Courier New” and “Courier Bold”.
If a text is written which should be used as is written standard version of “Courier New” is used.
If the word is written instead of several possibilities (in a programming language we would talk of a variable) italics are used.

Getting the system working

To get the system working proceed with the following steps:

1.) Install the HiPic or HPD-TA.
2.) Run the HiPic or HPD-TA once and verify that it operates correctly. This step registers the HiPic or HPDTA executable files correctly as ActiveX components.
3.) Run HiRemoteEx.exe or HiRemoteEx.exe from the application directory
4.) Type „Appstart()“ into the Text box labeled “direct command”. The HiPic or HPDTA should now start up. If it does not or if you get an automation error the HiPic or HPDTA may not be registered correctly.
One possible solution is to uninstall HiPic or HPDTA, Call RegClean to fix registry errors and install HiPic or HPDTA again.
If everything is OK type „Append()“into the Text box labeled direct command. The HiPic or HPDTA should disappear.
5.) The next step is to establish a communication between the client program and the HiRemoteEx.exe or TaRemoteEx.exe. A small sample program is delivered which is called RemoteExClient.exe. This command can communicate with both HiPic and HPDTA. You have to establish a communication via a TCP-IP port. Make sure that on both HiRemoteEx.exe or TaRemoteEx.exe and RemoteExClient.exe the TPC-IP port is specified identical. In our sample it is set to 1001. It is not necessary that also the secondary port (data port) is specified now. It is also important to specify the correct host name. The host name could either be a computers name (as it appears under network neighborhood, see also “Identifying the Host name” for details later) or a TCP-IP address. If you communicate the RemoteEx on the same computer “localhost” can be used as the computers name.
We assume that HiRemoteEx.exe or TaRemoteEx.exe is still running (Not necessarily the main application). Start RemoteExClient.exe and click to „Connect to Host“ on the left side. The Disconnect pushbutton should be enabled and also the „Send“ and „Send & Wait“ pushbuttons should be enabled.
If this is not the case, there are several possibilities:
   - The HiRemoteEx.exe or TaRemoteEx.exe is not running
   - The host name is not specified correctly
     The port numbers are not identical on HiRemoteEx.exe or TaRemoteEx.exe and RemoteExClient.exe
   - The system does not allow to access ports. Sometimes a virus scanner disables the access. Change the system setting accordingly
6.) Type „Appstart()“ in the command text box of the RemoteEx Client window. The program should startup. Type „Acqstart(acquire)“ to acquire an image. Type „Append()“ to end the application.



Schematic diagram of how the RemoteEx works.

The program HiRemoteEx.exe or TaRemoteEx.exe can be started in the autostart folder and can run continuously.

General syntax

The commands used in the RemoteEx application have the following syntax:
`CommandName(parameter1,parameter2,etc.)`
Example:
`appstart()` (Start the application)
A pair of parentheses is used to enclose the parameters. Parameters are separated by comma. Text or parameters should therefore never contain commas. Please make sure to delimit any command by a <CR> character (ASCII=13). In this document the <CR> character will not be shown because it is a non printable character.

Delimiter of commands and responses

At the end of any command the <Carriage Return> character (<CR>, ASCII value 13) has to be used. The RemoteEx also delimits any response by a <CR> character, thus individual responses can be separated by locating the <CR> character.

Case sensitivity

Interpretation of the command is case insensitive thus „**`CommandName`**“ is treated identical to „**`commandname`**“ or „**`COMMANDNAME`**“.

Command response

Every command is replied by an individual response. The command response contains the error code and the command name (not the full command sent to the RemoteEx application). This response should be use as a kind of handshake. A new command should not be sent unless the response for the last one has been detected. Sometimes a response contains one or more other parameters. The number of parameters and its meaning depend on the command.
Syntax of the response is:
EC*,*CommandName
or
EC*,*CommandName*,*parameter1*,*parameter2*,*etc.
where EC is an integer number indicating the Error code. If the command has been executed successfully EC is zero. Once the response has been sent, the system is ready to execute the next command. Though the RemoteEx program has an input FIFO for the command execution it is recommended to individually wait for the command response and react according to the error code and other returned parameters.
Example:
0,appstart (No error, command base name is returned)

Responses to TCP-IP connection

Whenever a client connects successfully to the command or data port of the RemoteEx the RemoteEx sends a response. This makes it easier to the client to find out whether the RemoteEx is available and whether the connection took place successfully.
The response are:

RemoteEx Ready <CR>	Response to the command port
RemoteEx Data Ready <CR>	Response to the data port

Messages and MsgBoxReply

Additionally to the command response which indicates the completion of the command messages are sent to the client program. They normally do not refer to a command and should not be used for command handshake. The same is true for strings which are sent instead of a MessageBoxReply. Messages MessageBoxReply strings can be distinguished from command responses by its error code. Error codes used in combination with Messages are ECMessage(4) and ECMsgBoxReply(5).
Example:
4,Application closed by user (Message,Message text)

Invalid syntax

If the syntax of the command is invalid (e.g. missing parenthesis) the following response is sent:
1,*FullCommand*,Invalid syntax
Example, command:
Appstart(((syntax not correct because right parenthesis is missing)
Response
1,Appstart((,Invalid syntax (Invalid syntax,fullcommand,text)
Normally responses do never contain parentheses. The case of invalid syntax is the only case where this happens because the full command is returned.

Text based communication

Commands and other information are always exchanged on a text base (This is not true in the case that image or other binary data is exchanged by a separate port; see a detailed explanation about data exchange later). Commands are significant expressions and normally can contain several parts. The first part always specifies the main circumstance where following parts give more detailed information. The associated action is always the last part of the command.

Example:
`AppInfo(directory)` (get info about application directory)
Parameters are mostly specified as text based keywords.
Example:
`AcqStart(Live)` **(**Start live mode)
Only if really numerical values are used these are specified in text formatted version.
Example:
`CamParamSet(AI,NrExposures,10)` (Set analog integration count to 10)

Priority commands

Since version 8.2 the structure of the RemoteEx has been simplified for the sake of speed and has no more priority commands (The commands itself which have been priority commands are still available for compatibility reasons).

Error codes

Every response and message which is sent back from the RemoteEx to the client is preceded by a number indicating its status. This status is comparable to the function value of Windows API functions, which normally returns an information if the functions has succeeded or failed. We call it the "error code". There are two situations where a string sent from the RemoteEx does not correspond to the command directly. These two situations are messages (sent during run time) and MessageBox Results (also sent during Runtime) with the ErrorCodes ECMessage and ECMsgBoxReply. Strings with these ErrorCodes are no responses to commands.
All other ErrorCodes are responses to commands. Only in the case of ECNoError the command has been successfully executed.

Error code	Meaning	Response to command
ECNoError (=0)	Command successfully executed	X
ECInvalidSyntax (=1)	Invalid syntax (command must be followed by parentheses and must have the correct number and type of parameters separated by comma)	X
ECUnkownCommandOrParameters (=2)	Command or Parameters are unknown.	X
ECCommandNotPossible (=3)	Command currently not possible	X
ECMessage (=4)	A message during runtime (example: a string indicating the frame rate during live mode)	
ECMsgBoxReply (=5)	Reply value of a message box. The structure of RemoteEx does not allow sending inquiry commands from the RemoteEx to the client. In cases where the standalone program needs to popup a message box to get some information from the user the RemoteEx just continues execution with the default value of this message box. When such case happens a string is sent to the RemoteEx Client informing it about this default value.	
ECMissingParameter (=6)	Parameter is missing	X
ECCannotExecute (=7)	Command cannot be executed	X
ECErrorDuringExecution (=8)	An error has occurred during execution	X
ECCannotSendData (=9)	Data cannot be sent by TCP-IP	X
ECValueOutOfRange (=10)	Value of a parameter is out of range	X

Protocol

The RemoteEx has a protocol feature (available from version 8.2.0 pf5) which writes all important events together with a time stamp to a text file. This feature can be switched on or off with a check box labeled “Write protocol”.
The protocol is written to a file RemoteExProtocol.txt in the directory of the RemoteEx program.
The format is as follows:

```
17479118.988   GEN           RemoteEx started 04-21-2008 14:06:42
17483643.169   DCM           AppStart()
17483997.321   DCR           4,Checking values from INIT
17483998.590   DCR           4,Checking Licence
17493009.192   DCR           4,Check validity of controls
17493013.427   DCR           4,Load main window
17493081.895   DCR           4,
17493548.863   DCR           0,AppStart
17507188.419   GEN           Command port connected
17508233.739   GEN           Data port connected
17523851.562   TCM           AcqStart(Live)
17524483.751   TCR           0,AcqStart
17535216.468   TCR           4,Frame rate 3,00 Hz
17536216.549   TCR           4,Frame rate 3,00 Hz
17536238.750   TCM           AcqStop()
17536363.402   TCR           0,AcqStop
17547479.294   DCR           4,Mouse moved to (584,127), (584 No unit, 127 No unit), Int: 4095
17566516.225   GEN           RemoteEx ended
```

The number in the first column is the timestamp in ms (The values are relative values. Under certain circumstances this denotes the time after booting up the computer system).
The abbreviation in the second column mean:

GEN:	General
DCM:	Direct command
DCR:	Response from direct command
TCM:	command sent by TCP-IP
TCR:	Response received by TCP-IP
DAR:	Data received on second port

The third column describes the text data associated to the command/response

List of commands and parameters

General commands

`Appinfo(type)`

Returns the current application type (**`HiPic`** or **`HPDTA`**). This command is executed even if the application has not been started.
Response
`0,Appinfo,HiPic`

`Status()`

Returns whether or not a command is currently executed
Response
`0,Status,idle`
`0,Status,busy,`*`commandname`*

Note: This command is still available but it is obsolete because it is only executed after the current command has been finished.

`Stop()`

Stops the command currently executed if possible. (Few commands have implemented this command right now)
Response:
`0,Stop`

`Shutdown()`

This command shuts down the application and the RemoteEx program. Response is sent before shutdown.
The usefulness of this command is limited because it cannot be sent once the application has been hang up. Restarting of the remote application if an error has occurred should be done by other means (example: Power off and on the computer from remote and starting the RemoteEx from the autostart).

Response:
`0,Shutdown`

Application Commands

`AppStart(`*`fVisible,sINIFile`*`)`

This command starts the application. If the application has already been started this command returns immediately, otherwise it waits until it has been started completely.
If fVisible is 0 or FALSE the application starts invisible. If this parameter is omitted or if it is others than 0 or FALSE the application starts visible. This parameter is ignored if the application is already running. If you want to make sure that the visible state is set if desired you should first close the application with **`AppEnd()`** and then restart it with the **`AppStart()`** command.

If sINIfile is specified the application starts with the INI-File (new from version 8.3.0). This parameter is also ignored if the application is already running.

AppEnd()

This command ends the application.

AppInfo(*parameter*)

This command returns information about the application.

Where ***parameter*** can be one of the following:

Date	Application date
Version	Application version
Directory	Application directory
Title	Application title
Titlelong	Application title (long version)

Response:
0,AppInfo,info

MainParamGet(*parameter*)

This command gets the values of parameters visible in the main window (new in version 8.2).

Parameter can be one of the following:

ImageSize	Size of an image which if it would be acquired now
Message	Message text
Temperature	Temperature for cameras with cooling whether the temperature can be readout

For the HPD-TA there are the following additional parameters:

GateMode	Gate mode
MCPGain	MCP gain
Mode	Mode
Plugin	Plugin
Shutter	Shutter
StreakCamera	Streak camera
TimeRange	Time range

MainParamInfo(*parameter*)/ MainParamInfoEx(*parameter*)

This command gets information about parameters visible in the main window (new in version 8.2). **MainParamInfoEx** (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **MainParamInfo.**
Example2:
MainParamInfo(Temperature)
Response:
0,MainParamInfo,Temperature,-50,5

The response consists of the following parts separated by commas:

Meaning	Value
Errorcode	**0**
CommandName	**MainParamInfo**
Label	**Temperature**

Current value **-50**
Parameter Type **5**

The Parameter Type can have the following values:
5=Display A string which is displayed only

GenParamGet(*parameter*)

This command gets the values of parameters in the general options (new in version 8.2).
Parameter can be one of the following:

RestoreWindowPos	Restore window positions
UserFunctions	Call user functions

For the HPD-TA there are the following additional parameters

ShowStreakControl	Shows or hides the Streak status/control dialog
ShowDelay1Control	Shows or hides the Delay1 status/control dialog
ShowDelay2Control	Shows or hides the Delay2 status/control dialog
ShowSpectrControl	Shows or hides the Spectrograph status/control dialog

Example:
GenParamGet(RestoreWindowPos)
Response:
0,GenParamGet,1

GenParamSet(*Parameter*,*Value*)

This command sets the value of parameters in the general options (new in version 8.2). Possible values for ***Parameter*** are described above**.**

Example:
GenParamSet(RestoreWindowPos,0)
Response:
0,GenParamSet

GenParamInfo(*Parameter*) / GenParamInfoEx(*Parameter*)

This command gets information about the specified parameter (new in version 8.2).
GenParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **GenParamInfo.**

Example:
GenParamInfo(RestoreWindowPos)
Response:
0,GenParamInfo,RestoreWindowPos,Restore window positions,1,0

The response consists of the following parts separated by commas:
Meaning Value
Errorcode **0**
CommandName **GenParamInfo**
Label **Restore window positions**
Current value **1**
Parameter Type **0**

The Parameter Type can have the following values:

0= Boolean	Can have the values true or false. Valid entries are „true“ (true), „false“ (false), „on“ (true), „off“ (false), „yes“ (true), „no“ (false), „0“ (false), or any other numerical value (true). On output only 0 (false) and 1 (true) is used.

Acquisition commands

AcqStart(*AcqMode*)

This command starts an acquisition.
AcqMode is one of the following:

Live	Live mode
Acquire	Acquire mode
AI	Analog integration
PC	Photon counting

Response:
0,AcqStart

AcqStatus()

This command returns the status of an acquisition.
Response:
0,AcqStatus,idle
or
0,AcqStatus,busy,Live

AcqStop()

This command stops the currently running acquisition. It can have an optional parameter (available from 8.2.0 pf5) indicating the timeout value (in ms) until this command should wait for an acquisition to end. The range of this timeout value is [1…60000] and the default value is 1000 (if not specified).
Example: **AcqStop(5000)** (waits maximum 5 seconds for an acquisition to end until it timeouts)

Response:
0,AcqStop (Successfully stopped)
or
7,AcqStop,timeout (Timeout while waiting for stop)

AcqParamGet(*Parameter*)

This command gets the values of the acquisition options (See the meaning of these options in the manual or help file)
Parameter can be one of the following:

DisplayInterval	Display interval in Live mode
32BitInAI	Creates 32 bit images in Analog integration mode
WriteDPCFile	Writes dynamic photon counting file
AdditionalTimeout	Additional timeout
DeactivateGrbNotInUse	Deactivate the grabber while not in use
CCDGainForPC	Default setting for photon counting mode (new in version 8.2)
32BitInPC	Create 32 bit images in Photon counting mode (new in version 8.2)
MoireeReduction	Strength of Moiré reduction (new in version 8.2)

The following parameter is no longer available from version 8.2:

PCMode	Photon counting mode

Example:
AcqParamGet(32BitInAI)
Response:
0,acqparamget,1 The parameter 32BitInAI is set to true

AcqParamSet(*Parameter*,*Value*)

This command sets the specified parameter of the acquisition options. Possible values for ***Parameter*** are described above**.**
Example:
acqparamset(DisplayInterval,100)
Response:
0,acqparamset

AcqParamInfo(parameter) / AcqParamInfoEx(parameter)

This command gets information about the specified parameter.
AcqParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **AcqParamInfo.**

Example:
acqparaminfo(AdditionalTimeout)
Response:
0,acqparaminfo,AdditionalTimeout [sec]:,0,1,0,1800
or
Example2:
acqparaminfo(PCMode)
Response:
0,acqparaminfo,photon counting method,Gravity,2

The response consists of the following parts separated by commas:

Meaning	Value
Errorcode	**0**
CommandName	**acqparaminfo**
Label	**AdditionalTimeout [sec]:**
Current value	**0**
Parameter Type	**1**
Minimum (numerical type only)	**0**
Maximum (numerical type only)	**1800**

The Parameter Type can have the following values:

0= Boolean	Can have the values true or false. Valid entries are „true“ (true), „false“ (false), „on“ (true), „off“ (false), „yes“ (true), „no“ (false), „0“ (false), or any other numerical value (true). On output only 0 (false) and 1 (true) is used.
1= Numeric	A numerical value. In the case of a numerical value the minimum and maximum value is returned (But not for other parameter types).
2= List	The value is one entry in a list.
3=String	Any string can be used
4= ExposureTime	An expression which evaluates to a time like „5ms“, „1h“, „1s“ etc.

Valid units are ns (nanosecond), us (microsecond), ms (millisecond), s (second), m (minute), h(hour)

Note: In case of a list or an exposure time the number of entries and all list entries are returned in the response of the AcqParamInfoEx command.

AcqLiveMonitor(*MonitorType*)

This command starts a mode which returns infomations on every new image acquired in live mode. Once this command is activated Together with every new live image a message is sent with certain information.

By setting ***MonitorType*** to one of the following values these types of information can be obtained:

Off	No messages are output. This setting can be used to stop live monitoring.
Notify	A message is sent with every new live image. No other information is attached. The message can then be used to observe activity or to get image or other data explicitly.
NotifyTimeStamp	(available from 8.2.0 pf5) A message is sent with every new live image. The message contains the timestamp of the image when it was acquired in ms.
RingBuffer	The data acquired in Live mode is written to a ring buffer inside the RemoteEx application. A message is sent with every new live image. This message contains a sequence number. The imgRingBufferGet command can be used to get the data associated to the specified sequence number. Please see also the description of the imgRingBufferGet command and the description of the sample client program. Note: Because the data is transferred by ActiveX from one application to anther this method cannot be used for systems with very high data rate (like the C9300 camera).
Average	Returns the average value within the full image or a specified area.
Minimum	Returns the minimum value within the full image or a specified area.
Maximum	Returns the maximum value within the full image or a specified area.
Profile	Returns a profile extracted within the full image or a specified area in text form.

The Syntax of the command can be either of the following:

AcqLiveMonitor(*MonitorType*)

This format applies to ***MonitorType*** =**Off/Notify**

AcqLiveMonitor(*MonitorType,NumberOfBuffers*)

This format applies to ***MonitorType*** =**RingBuffer**

NumberOfBuffers specifies the number of buffers allocated inside the RemoteEx.

AcqLiveMonitor(*MonitorType*,FullArea)

This format applies to ***MonitorType***=**Average/Minimum/Maximum.** The specified calculation algorithm is performed on the full image area.

AcqLiveMonitor(*MonitorType*,Subarray,*X*,*Y*,*DX*,*DY*)

This format applies to ***MonitorType*=Average/Minimum/Maximum** The specified calculation algorithm is performed on a sub array specified by X (X-Offset), Y (Y-Offset), DX

(Image width) and DY (Image height).

AcqLiveMonitor(*MonitorType*,*ProfileType*,FullArea)
This format applies to ***MonitorType*=Profile.**
ProfileType can be one of the following:
1=Line profile
2=Horizontal profile (integrated)
3=Vertical profile (integrated)
The profile is extracted from the full image area

AcqLiveMonitor(*MonitorType*,*ProfileType*,Subarray,*X*,*Y*,*DX*,*DY*)
This format applies to ***MonitorType*=Profile.** The profile is extracted from a subarray specified by X (X-Offset), Y (Y-Offset), DX (Image width) and DY (Image height).

The response is:
0,AcqLiveMonitor

During live monitor the following messages can appear:
4,AcqLiveMonitor,notify (Notify)
4,AcqLiveMonitor,notifytimestamp,*timestamp* (Notify timestamp)
4,AcqLiveMonitor,ringbuffer,*Seqnumber,timestamp* (RingBuffer)
4,AcqLiveMonitor,*data* (Average,Minimum,Maximum)
4,AcqLiveMonitor,*data0,data1,...* (Profile)

Camera commands

CamParamGet(*Location,Parameter*)

This command gets the values of the camera options (See the meaning of these options in the manual or help file)

Location can be one of the following:

Setup	Part of the options dialog
Live	Parameters on the Live tab of the acquisition dialog
Acquire	Parameters on the Acquire tab of the acquisition dialog
AI	Parameters on the Analog Integration tab of the acquisition dialog
PC	Parameters on the Photon counting tab of the acquisition dialog

Parameter can be one of the following (Which of these parameters are relevant is dependent on the camera type. Please refer to the camera options dialog):

Setup (options) parameter

TimingMode	Timing mode (Internal / External)
TriggerMode	Trigger mode
TriggerSource	Trigger source
TriggerPolarity	Trigger polarity
ScanMode	Scan mode
Binning	Binning factor
CCDArea	CCD area
LightMode	Light mode
Hoffs	Horizontal Offset (Subarray)
HWidth	Horizontal Width (Subarray)
VOffs	Vertical Offset (Subarray)
VWidth	Vertical Width (Subarray)
ShowGainOffset	Show Gain and Offset on acquisition dialog
NoLines	Number of lines (TDI mode)
LinesPerImage	Number of lines (TDI mode)
ScrollingLiveDisplay	Scrolling or non scrolling live display
FrameTrigger	Frame trigger (TDI or X-ray line sensors)
VerticalBinning	Vertical Binning (TDI mode)
TapNo	Number of Taps (Multitap camera)
ShutterAction	Shutter action
Cooler	Cooler switch
TargetTemperature	Cooler target temperature
ContrastEnhancement	Contrast enhancement
Offset	Analog Offset
Gain	Analog Gain
XDirection	Pixel number in X direction
Offset	Vertical Offset in Subarray mode
Width	Vertical Width in Subarray mode
ScanSpeed	Scan speed
MechanicalShutter	Behavior of Mechanical Shutter
Subtype	Subtype (X-Ray Flatpanel)
AutoDetect	Auto detect subtype
Wait2ndFrame	Wait for second frame in Acquire mode

DX	Image Width (Generic camera)
DY	Image height (Generic camera)
XOffset	X-Offset (Generic camera)
YOffset	Y-Offset (Generic camera)
BPP	Bits per Pixel(Generic camera)
CameraName	Camera name (Generic camera)
ExposureTime	Exposure time (Generic camera)
ReadoutTime	Readout time Generic camera)
OnChipAmp	On chip amplifier
CoolingFan	Cooling fan
Cooler	Coolier
ExtOutputPolarity	External output polarity
ExtOutputDelay	External output delay
ExtOutputWidth	External output width
LowLightSensitivity	Low light sensitivity
DCam3SetupProp_xxxx	A setup property in the Options(setup) of a DCam 3.0 module. The word xxxx stand for the name of the property (This is what you see in the labeling of the property). Blanks or underscores are ignored. Example: Dcam2SetupProp_ReadoutDirection (a parameter for the C10000)

Parameters on the acquisition Tabs of the Acquisition dialog

Exposure	Exposure time
Gain	Analog gain
Offset	Analog Offset
NrTrigger	Number of trigger
Threshold	Photon counting threshold
DoRTBacksub	Do realtime background subtraction
DoRTShading	Do realtime shading correction
NrExposures	Number of exposures
ClearFrameBuffer	Clear frame buffer on start
AmpGain	Amp gain
SMD	Scan mode
RecurNumber	Recursive filter
HVoltage	High Voltage
AMD	Acquire mode
ASH	Acquire shutter
ATP	Acquire trigger polarity
SOP	Scan optical black
SPX	Superpixel
MCP	MCP gain
TDY	Time delay
IntegrAfterTrig	Integrate after trigger
SensitivityValue	Sensitivity (value)
EMG	EM-gain (EM-CCD camera)
BGSub	Background Sub
RecurFilter	Recursive Filter
HighVoltage	High Voltage
StreakTrigger	Streak trigger
FGTrigger	Frame grabber Trigger
SensitivitySwitch	Sensitivity (switch)
BGOffset	Background offset

ATN	Acquire trigger number
SMDExtended	Scan mode extended
LightMode	Light mode
ScanSpeed	Scan Speed
BGDataMemory	Memory number for background data (Inbuilt background sub)
SHDataMemory	Memory number for shading data (Inbuilt shading correction)
SensitivityMode	Sensitivity mode
Sensitivity	Sensitivity
Sensitivity2Mode	Sensitivity 2 mode
Sensitivity2	Sensitivity 2
ContrastControl	Contrast control
ContrastGain	Contrast gain
ContrastOffset	Contrast offset
PhotonImagingMode	Photon Imaging mode
HighDynamicRangeMode	High dynamic range mode
RecurNumber2	Second number for recursive filter (There is a software recursive filter and some camera have this as a hardware feature) (new from 8.3.0)
RecurFilter2	Second recursive filter (There is a software recursive filter and some camera have this as a hardware feature) (new from 8.3.0)
FrameAvgNumber	Frame average number
FrameAvg	Frame average

CamParamSet(Location,Parameter,Value)

This command sets the specified parameter of the acquisition options. Possible values for ***Parameter*** are described above. When specifying a parameter value the value has to be written as it appears in the corresponding control.

CamParamInfo(*Parameter*) / CamParamInfoEx(*Parameter*)

This command gets information about the specified parameter.
CamParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **CamParamInfo.**

Example:
CamParamInfo(AdditionalTimeout)
Response:
0,acqparaminfo,AdditionalTimeout [sec]:,0,1,0,1800
or
Example2:
camparaminfo(setup,binning)
Response:
0,camparaminfo,Binning,2 x 2,2

The response consists of the following parts separated by commas:
Meaning Value
Errorcode **0**
CommandName **camparaminfo**
Label **Binning**

Current value **2 x 2**
Parameter Type **2**
Minimum (numerical type only) **0**
Maximum (numerical type only) **1800**

The Parameter Type can have the following values:

0= Boolean	Can have the values true or false. Valid entries are „true“ (true), „false“ (false), „on“ (true), „off“ (false), „yes“ (true), „no“ (false), „0“ (false), or any other numerical value (true). On output only 0 (false) and 1 (true) is used.
1= Numeric	A numerical value. In the case of a numerical value the minimum and maximum value is returned (But not for other parameter types).
2= List	The value is one entry in a list.
3=String	Any string can be used
4= ExposureTime	An expression which evaluates to a time like „5ms“, „1h“, „1s“ etc. Valid units are ns (nanosecond), us (microsecond), ms (millisecond), s (second), m (minute), h(hour)

Note: In case of a list or an exposure time the number of entries and all list entries are returned in the response of the CamParamInfoEx command.

CamGetLiveBG()

This command gets a new background image which is used for real time background subtraction (RTBS). It is only available of LIVE mode is running. (New from 8.3.0)

External devices commands (HPD-TA only)

These commands refer to the HPD-TA only. They are not available in the HiPic.

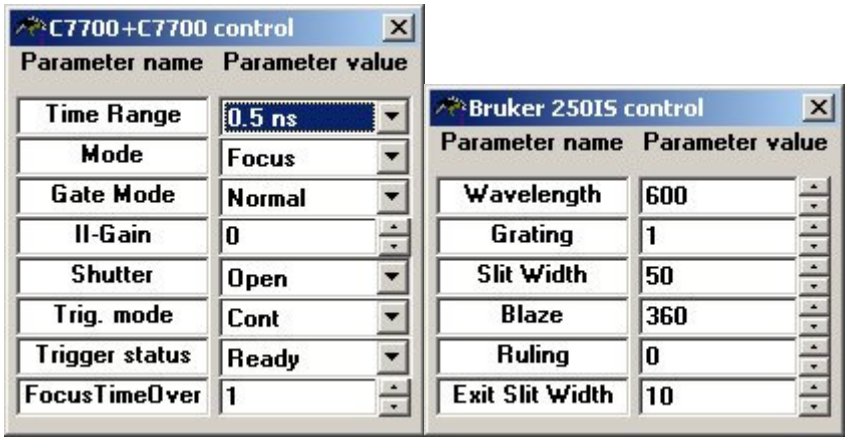
DevParamGet(Location,Parameter)

This command gets the values of the camera parameters (See the meaning of these options in the manual or help file)

Location can be one of the following:

TD	Streak camera
Streak	Streak camera
Streakcamera	Streak camera
Spec	Spectrograph
Spectrograph	Spectrograph
Del	Delaybox 1
Delay	Delaybox 1
Delaybox	Delaybox 1
Del1	Delaybox 1
Del2	Delaybox 2
Delay2	Delaybox 2
DelayBox2	Delaybox 2

Parameter can be every parameter which appears in the external devices status/control box. It should be written as indicated in the Parameter name field.



Example:
DevParamGet(TD,Time Range)
Response:
0,DevParamGet,0.5 ns

or

DevParamGet(Spec,Wavelength)
Response:
0,DevParamGet,600

DevParamSet(Location,Parameter,Value)

This command sets the specified parameter of the acquisition options. Possible values for ***Parameter*** are described above. When specifying a parameter value the value has to be written as it appears in the corresponding control.

Example:
DevParamSet(TD,Mode,Operate)
Response:
0,DevParamSet

or

DevParamSet(Spec,Slit Width,20)
Response:
0,DevParamSet

DevParamInfo(*Device*, *Parameter*) / DevParamInfoEx(*Device*,*Parameter*)

This command gets information about the specified parameter.
DevParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **DevParamInfo.**

Example:
DevParamInfo(TD,Time Range)
Response:
0,**DevParamInfo,Time Range,0.5 ns,2**

The response consists of the following parts separated by commas:

Meaning	Value
Errorcode	**0**
CommandName	**DevParamInfo**
Label	**Time Range**
Current value	**0.5 ns**
Parameter Type	**2**
Minimum (numerical type only)	**0**
Maximum (numerical type only)	**64**

The Parameter Type can have the following values:

1= Numeric	A numerical value. In the case of a numerical value the minimum and maximum value is returned (But not for other parameter types).
2= List	The value is one entry in a list.

Note: In case of a list the number of entries and all list entries are returned in the response of the DevParamInfoEx command.

Example: DevParamInfoEx(TD,Time Range)
Response: 0,DevParamInfoEx,0,0,Time Range,5 ns,2,17,5 ns,10 ns,20 ns,50 ns,100 ns,200 ns,500 ns,1 us,2 us,5 us,10 us,20 us,50 us,100 us,200 us,500 us,1 ms

Meaning	Value
Errorcode	**0**
CommandName	**DevParamInfoEx**
ControlAvailable	0
StatusAvailable	0
Label	**Time Range**
Current value	**0.5 ns**
Parameter Type	**2**
Number of entries	17

Entry 1	5 ns
Entry 2	10 ns
…	
Entry 17	1 ms

DevParamsList(*Device)*

This command returns a list of all parameters of a specified device.
Example: DevParamsList(TD)
Response: 0,DevParamsList,11,Time Range,Mode,Gate Mode,MCP Gain,Shutter,Gate Time,Trig. mode,Trigger status,Trig. level,Trig. slope,FocusTimeOver

Correction commands

CorParamGet(Location,Parameter)

This command gets the values of the correction options (See the meaning of these options in the manual or help file)

Location can be one of the following:

Background	Background Subtraction options dialog
Shading	Shading correction options dialog
Curvature	Curvature correction options dialog
DefectPixel	Defect pixel correction options dialog

Parameter can be one of the following (Which of these parameters are relevant is dependent on the detailed circumstance. Please refer to the respective correction options dialog):

When ***Location*** =**Background** (Background subtraction parameter)

BackgroundSource	Source for Background subtraction
BackFilesForAcqModes	Individual background files for every acquisition mode
LiveFile	Correction file for Live mode
AcquireFile	Correction file for Acquire mode
AIFile	Correction file for Analog Integration mode
Constant	Constant added during background subtraction
ClipZero	Clip values to zero during background subtraction
Deleted: RTBSSource	**This feature has been deleted from version 8.1** Source for real-time background subtraction
AutoBacksub	Auto backsub function

When ***Location*** =**Curvature** (Curvature correction parameter, refers to HPD-TA only)

CorrectionFile	Curvature correction file
AutoCurvature	Auto curvature correction function

When ***Location*** =**DefectPixel** (Defect Pixel correction parameter)

DefectCorrection	Defect pixel correction function
DefectPixelFile	Defect pixel correction file

When ***Location*** =**Shading** (Shading correction parameter)

ShadingFile	Image file used for shading correction
ShadingConstant	Defines how to calculate the constant during shading correction
AutoShading	Auto shading correction function
SensitivityCorrection	Sensitivity correction function
LampFile	Lamp file for Sensitivity correction function

CorParamSet(Location,Parameter,Value)

This command sets the specified parameter of the acquisition options. Possible values for ***Parameter*** are described above. When specifying a parameter value the value has to be written as it appears in the corresponding control.

CorParamInfo(*Parameter*) / CorParamInfoEx(*Parameter*)

This command gets information about the specified parameter.

CorParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **CorParamInfo.**

CorDoCorrection(*Destination,Type*)

This command performs a correction on the specified image.
Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number

Type can be one of the following:

Backsub	Background subtraction
Shading	Shading correction
Curvature	Curvature correction
BacksubShading	Background subtraction + Shading correction
BacksubCurvature	Background subtraction + Curvature correction
BacksubShadingCurvature	Background subtraction + Shading correction + Curvature correction

Example:
CorDoCorrection(Current,Backsub)
Response:
0,CorDoCorrection

Note: The corrections can also be applied to an image containing a sequence. In this case the correction is applied to all images in the sequence.

Defect pixel tool commands

(all these commands are available from version 8.3.0)

These commands can be used to get defect pixel coordinates from dark and light reference files and cooperate together with the defect pixel tool.

DefPixParamGet(Parameter)

This command gets the values of the correction options (See the meaning of these options in the manual or help file)

Parameter can be one of the following:

Method	Defines whether files for hot, dead or hot and dead pixels are used to calculate the coordinates of defects. See also the DefPixSetType command.
ImageHotPixel	Image file for hot pixels (dark/background file). See also DefPixLoadHot command.
AverageHotPixel	Average in the hot pixel file.
StandDevHotPixel	Standard deviation in the hot pixel file.
ThresholdHotPixels	Threshold to apply to the hot pixel file to find single defective pixels.
ThresholdHotLines	Threshold to apply to the hot pixel file to find defect lines or columns. Works only in combination with LineColumnsPercentage.
ImageDeadPixel	Image file for dead pixels (dark/background file), see also DefPixLoadDead command.
AverageDeadPixel	Average in the dead pixel file.
StandDevDeadPixel	Standard deviation in the dead pixel file.
ThresholdDeadPixels	Threshold to apply to the dead pixel file to find single defective pixels.
ThresholdDeadLines	Threshold to apply to the dead pixel file to find defect lines or columns. Works only in combination with LineColumnsPercentage.
NrDefectPixels	Number of defective single pixels found.
NrDefectLines	Number of defective lines found.
NrDefectColumns	Number of defective columns found.
NrDefectOverflowLines	Number of defective overflow lines found.
NrDefectOverflowColumns	Number of defective overflow columns found.
OvlLinColFactor	Correction factor for overflow lines or columns.
NrUncorrectable	Number of uncorrectable pixels found

DefPixParamSet(Parameter,Value)

This command sets the specified parameter of the defect pixel tools. Possible values for ***Parameter*** are described above. When specifying a parameter value the value has to be written as it appears in the corresponding control.

DefPixParamInfo(*Parameter*) / DefPixParamInfoEx(*Parameter*)

This command gets information about the specified parameter.
DefPixParamInfoEx returns more detailed information in case of a list parameter (Parameter type = 2) than **DefPixParamInfo.**

DefPixCalculate()

This command calculates the coordinates of defective single pixels, defective lines or columns or overflow lines or columns as a result of the input values.

DefPixShow()

This command shows the defects found previously as overflow values in a separate image (the modified hot or dead file is used to display the defects).

DefPixSave(sFile)

This command saves the coordinates found previously in an INI file. sFile is the filename of the INI file.

DefPixSaveActivate(sFile)

This command saves the coordinates found previously in an INI file and sets this file to the currently active defect pixel file and activates the defective pixel correction (See also the options in the defective pixel correction options dialog). sFile is the filename of the INI file.

DefPixLoadHot(sFile)

This command loads the specified file as the hot pixel file. Please be sure to specify **Method** first or execute the command DefPixSetType. sFile is the filename of the hot pixel file. Use this command instead of the parameter **ImageHotPixel** if you want to calculate the image properties of the hot pixel file (average, standard deviation) and the suggested thresholds.

DefPixLoadDead(sFile)

This command loads the specified file as the hot pixel file. Please be sure to specify **Method** first or execute the command DefPixSetType. sFile is the filename of the dead pixel file. Use this command instead of the parameter **ImageDeadPixel** if you want to calculate the image properties of the dead pixel file (average, standard deviation) and the suggested thresholds.

DefPixSetType(sType)

This command specifies whether files for hot, dead or hot and dead pixels are used to calculate the coordinates of defects. See also the parameter Method.
If sType contains the word “hot” hot files can be used. If sType contains the word “daed” hot files can be used. If sType contains the word “hot” and “dead”, hot and dead files can be used.

Image commands

ImgParamGet(*Parameter*)

This command gets the values of the image options (See the meaning of these options in the manual or help file)

Parameter can be one of the following:

AcquireToSameWindow	Acquire always to the same window
DefaultZoomFactor	Default zooming factor
WarnWhenUnsaved	Warn when unsaved images are closed
Calibrated	Calibrated (Quickprofiles, Rulers, FWHM)
LowerLUTIsZero	Force the lower LUT limit to zero when executing auto LUT
AutoLUT	AutoLut function
AutoLUTInLive	AutoLut in Live mode function
AutoLUTInROI	Calculate AutoLut values in ROI
HorizontalRuler	Display horizontal rulers
VerticalRuler	Display vertical rulers
FixedITEXHeader	Save ITEX files with fixed header

ImgParamSet(*Parameter*,*Value*)

This command sets the specified parameter of the quick profile options. Possible values for ***Parameter*** are described above**.**

ImgParamInfo(*Parameter*) / ImgParamInfoEx(*Parameter*)

This command gets information about the specified parameter.

ImgParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **ImgParamInfo.**

Example:

ImgParamInfo(Calibrated)

Response:

0,ImgParamInfo,1 (Calibrated was set to true)

ImgSave(*Destination*,*ImageType*,*FileName*,*Overwrite*)

Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number

ImageType can be one of the following:

IMG	ITEX image
TIF	TIFF image
TIFF	TIFF image
ASCII	ASCII file
data2tiff	Data to tiff
data2tif	Data to tiff
display2tiff	Display to tiff
display2tif	Display to tiff

FileName can be any valid filename. This function can also save images on a network device, so it can transfer image data from one computer to another computer.

Overwrite can be either true or false. This is an optional parameter. If this is set to true (or 1) the

file is also saved if it exists. If the parameter is omitted or is set to false (or 0) the file is not saved if it already exists and an error is returned.

ImgLoad(*ImageType,FileName*)

ImageType and ***FileName*** are values described above. Please not that not all file types which can be saved can also be loaded. Some file types are intended for export only.
Note: This load functions loads the image always into a new window independently of the setting of the option **AcquireToSameWindow.** If the maximum number of windows is reached an error is returned.
Response:
0,ImgLoad,*ImageNumber*
ImageNumber is the image number of the image loaded.

ImgDelete(*Destination*)

Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number
All	Deletes all images

Note1: This function deletes the specified images independent whether their content has been saved or not. If you want to keep the content of the image please save the image before executing this command.
Note2: This function does not delete images on hard disk.

ImgStatusGet()

The **ImgStatusGet** function retrieves information of the image status of a specified image. The image status is a part of the image header containing information about the circumstances of how the image has been created. It can have the following syntax:
ImgStatusGet(*Destination,*All)
ImgStatusGet(*Destination,*Section,*Sectionidentifier*)
ImgStatusGet(*Destination,* Token,*Sectionidentifier,Tokenidentifier*)

Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number

Type can be one of the following:

All	The full image status is returned
Section	A specified section is returned
Token	A specified Token within a specified section is returned

Sectionidentifier,Tokenidentifier are valid Sectionidentifiers and Tokenidentifiers.

Example:
ImgStatusGet(Current,Token,Application,Date)
Response:
0,ImgStatusGet,04-07-2006
Note1: Even though the commands and parameters are generally case insensitive, Sectionidentifiers

and Tokenidentifieres have to be specified as they appear in the image status, thus specifying Application will return a valid section but application will not.
Note2: Even though the image status may contain <CR> and <LF> characters these are removed before the status is returned.

ImgStatusSet(Destination,*Token,Sectionidentifier,Tokenidentifier*)

The ImgStatusSet writes tokens to the specified sections.
Destination, Sectionidentifier and Tokenidentifier have the same meaning as described above.
This command can also write new tokens and new sections, this it can be used to add user specific information to the images.
Note: Care has to be taken if existing tokens are modified. Some of the tokens are essential and should not be modified.

ImgDataInfo(*Destination, DataType*)

This function returns information about the current image data.

Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number

Currently only **Size** can be specified as the ***DataType***.

This command returns the image size in pixels and the Bytes per pixel of a single pixel. It returns:
0,ImgDataInfo,iX,iY,iDX,iDY,BPP
where

iX	X-Offset
iY	Y-Offset
iDX	Horizontal size in pixels
iDY	Vertical size in pixels
BPP	Bytes per Pixel

Example:
ImgDataInfo(Current, Size)
Response:
0,ImgDataInfo,0,0,1024,1024,2

ImgDataGet(*Destination,Type*)

This command get image, display or profile data of the select image.

Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number

Type can be one of the following:

Data	The image raw data (1,2 or 4 BBP)

Display	The display data (always 1 BBP)
Profile	A profile is returned (4 bytes floating point values)

The image data is transferred by the optional second TCP-IP channel. If this channel is not available an error is issued.

If **Profile** is selected for ***Type*** the syntax is:
ImgDataGet(*Destination,Type,Profiletype,iX,iY,iDX,iDY*)
where ***Profiletype*** has to be one of the following:

1=Line profile
2=Horizontal profile (integrated)
3=Vertical profile(integrated)

iX,iY,iDX,iDY are the coordinates of the area where to extract the profile.

The response is:
0,ImgDataGet,iDX,iDY,BBP,Type (Data,Display)
0,ImgDataGet,NumberOfData,Type (Profile)

Example:
ImgDataGet(current,data)
Response:
0,ImgDataGet,1024,1024,2,0

ImgDataDump(Destination,Type,filename)

This command get image or display data of the select image and writes it to file (only binary data, no header). It can be used to get image or profile data alternatively to using the second TCP-IP port.

Destination can be one of the following:

Current	The currently selected image
A number from 0 to 19	The specified image number

Type can be one of the following:

Data	The image raw data (1,2 or 4 BBP)
Display	The display data (always 1 BBP)
Profile	A profile is returned (4 bytes floating point values)

The image data is transferred by the optional second TCP-IP channel. If this channel is not available an error is issued.

File can be any valid file name including files on network devices.

If **Profile** is selected for ***Type*** the syntax is:
ImgDataGet(*Destination,Type,Profiletype,iX,iY,iDX,iDY,filename*)
where ***Profiletype*** has to be one of the following:

1=Line profile
2=Horizontal profile (integrated)
3=Vertical profile(integrated)

iX,iY,iDX,iDY are the coordinates of the area where to extract the profile.

The response is:
0,ImgDataDump,iDX,iDY,BBP,Type (Data,Display)
0,ImgDataDump,NumberOfData,Type (Profile)

Example:

ImgDataDump(current,data,c:\test.dat)
Response:
0,ImgDataDump,1024,1024,2,0

ImgRingBufferGet(*Type,SeqNumber,filename*)

This command get image or profile data of the select image. This command can be used only in combination with AcqLiveMonitor(RingBuffer,***NumberOfBuffers***). As soon as AcqLiveMonitor with option RingBuffer has been started the data of every new live image is written to a ring buffer and a continuously increasing sequence number is assigned to this data. As long as the image with this sequence number is still in the buffer it can be accessed by calling **ImgRingBufferGet(*Type,SeqNumber).*** If ***SeqNumber*** is smaller then the oldest remaining live image in the sequence buffer, the oldest live image is returned together with its sequence number. If ***SeqNumber*** is higher than the most recent live image in the buffer an error is returned.

Type can be one of the following:

Data	The image raw data (1,2 or 4 BBP)
Profile	A profile is returned (4 bytes floating point values)

SeqNumber is the sequence number of the image to get
Filename (optional) File where to write to data to. Raw data is written to the file without any header.
If a file name is specified the date is written to this file (same as with ImgDataDump). If no file name is written the image data is transferred by the optional second TCP-IP channel. If this channel is not available an error is issued.

If **Profile** is selected for ***Type*** the syntax is:
ImgRingBufferGet(Profile,*Profiletype,iX,iY,iDX,iDY,seqnumber,file*)
where ***Profiletype*** has to be one of the following:

1=Line profile
2=Horizontal profile (integrated)
3=Vertical profile(integrated)

iX,iY,iDX,iDY are the coordinates of the area where to extract the profile.

The response is:
0,ImgRingBufferGet,iDX,iDY,BBP,Type,seqnumber (Data,Display)
0,ImgRingBufferGet,NumberOfData,Type,seqnumber (Profile)

Example:
ImgRingBufferGet(data,125)
Response:
0,ImgRingBufferGet,1024,1024,2,0,125

Quick profile commands

QprParamGet(*Parameter*)

This command gets the values of the quick profile options (See the meaning of these options in the manual or help file)
Parameter can be one of the following:

UseMinAsZero	Use Minimum as zero FWHM calculation
DisplayQPOutOfImage	Display the Quick profile outside of the image
QPRelativeSpace	Relative space for Quick profile
DisplayDirectionForRect	Direction for the display of rectangular ROIs
AdjustQPHeight	Adjustment criterion for the Height of the quick profile
DisplayFWHM	Display FWHM
FWHMColor	Color of FWHM number
FWHMSize	Size of FWHM number
FWHMNoOfDigits	Number of digits for FWHM number

QprParamSet(*Parameter*,*Value*)

This command sets the specified parameter of the quick profile options. Possible values for ***Parameter*** are described above**.**

QprParamInfo(*Parameter*) / QprParamInfoEx(*Parameter*)

This command gets information about the specified parameter.
QprParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **QprParamInfo.**

Example:
QprParamInfo(QPRelativeSpace)
Response:
0,QprParamInfo,20 (The relative space for the quick profile is 20%)

LUT commands

LutParamGet(*Parameter*)

This command gets the values of the Lut options (See the meaning of these options in the manual or help file)

Parameter can be one of the following:

Limits	Limits of the LUT control. Three values are returned (upper and lower limit and multiplication factor)
Cursors	Cursors of the LUT control. Two values are returned (upper and lower cursor and multiplication factor)
Color	Lut color
Inverted	Inverted
Gamma	Gamma factor
Linearity	Linearity
Overflowcolors	Overflow colors (superimposed images only)

LutParamSet(*Parameter*,*Value*)

This command sets the specified parameter of the Lut options. Possible values for ***Parameter*** are described above**.**

In case of **Cursors** two values have to be set. The parameter **Limits** cannot be set.

LutParamInfo(*Parameter*) / LutParamInfoEx(*Parameter*)

This command gets information about the specified parameter.

LutParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **LutParamInfo.**

LutSetAuto()

This command executes the AutoLut functions. Three parameters are returned (upper and lower cursor and multiplication factor).

Sequence commands

SeqParamGet(*Parameter*)

This command gets the values of the Sequence options or parameters (See the meaning of these options or parameters in the manual or help file)
Parameter can be one of the following:

From options

AutoCorrectAfterSeq	Do auto corrections after sequence
DisplayImgDuringSequence	Always display image during acquisition
PromptBeforeStart	Prompt before start
EnableStop	Enable stop
Warning	Warning on
EnableAcquireWrap	Enable wrap during acquisition
LoadHISSequence	Load HIS sequences after acquisition
PackHisFiles	Pack 10 or 12 bit image files in a HIS file
NeverLoadToRAM	Do not attempt to load a sequence to RAM
LiveStreamingBuffers	Number of Buffers for Live Streaming
WrapPlay	Wrap during play
PlayInterval	Play interval
ProfileNo	Profile number for jitter correction
CorrectionDirection	Jitter Correction direction

From Acquisition Tab

AcquisitionMode	Acquisition mode
NoOfLoops	No of Loops
AcquisitionSpeed	Acquisition speed (full speed / fixed intervals)
AcquireInterval	Acquire interval
DoAcquireWrap	Do wrap during acquisition

From Data storage Tab

AcquireImages	Store images
ROIOnly	Acquire images in ROI
StoreTo	Data storage
FirstImgToStore	File name of first image to store
DisplayDataOnly	Store display data (8 bit with LUT)
UsedHDSpaceForCheck	Amount of HD space for HD check
AcquireProfiles	Store profiles
FirstPrfToStore	File name of first profile to store

From processing Tab

AutoFixpoint	Find Fixpoint automatically
ExcludeSample	Exclude the current sample

From general sequence dialog

SampleType	Sample type
CurrentSample	Index of current sample

SeqParamSet(*Parameter*,*Value*)

This command sets the specified parameter of the Sequence options or parameters. Possible values for ***Parameter*** are described above**.**

SeqParamInfo(*Parameter*) / SeqParamInfoEx(*Parameter*)

This command gets information about the specified parameter.
SeqParamInfoEx (available from 8.2.0 pf5) returns more detailed information in case of a list parameter (Parameter type = 2) than **SeqParamInfo.**

SeqStart()

Starts a sequence acquisition with the current parameters. Please note that any sequence which eventually exist is overwritten by this command.

SeqStop()

Stops the sequence acquisition currently under progress.

SeqStatus()

Returns the current sequence status.
Response:
0,SeqStatus,idle (no sequence acquisition under progress)
0,SeqStatus,busy,*PendingAcquisition* (sequence acquisition under progress)

PendingAcquisition can be either **Sequence Acquisition, Live Streaming, Save Sequence, Load Sequence** or **No sequence related async command: *command***

SeqDelete()

Deletes the current sequence from memory.
Note: This function does not delete a sequence on the hard disk.

SeqSave(ImageType,FileName,Overwrite)

ImageType,FileName,Overwrite are same as described under ImgSave().

SeqLoad(ImageType,FileName)

ImageType,FileName are same as described under ImgLoad().
Response:
0,SeqLoad,*ImageNumber*
ImageNumber describes the image number of the sequence image.

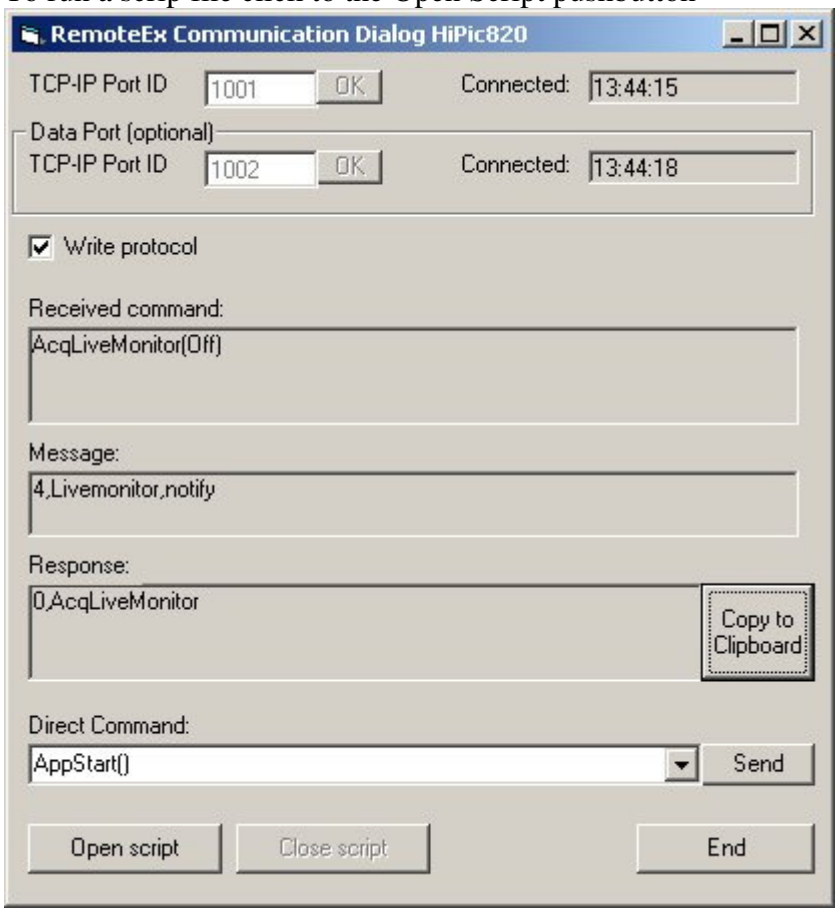
Using Script files

General

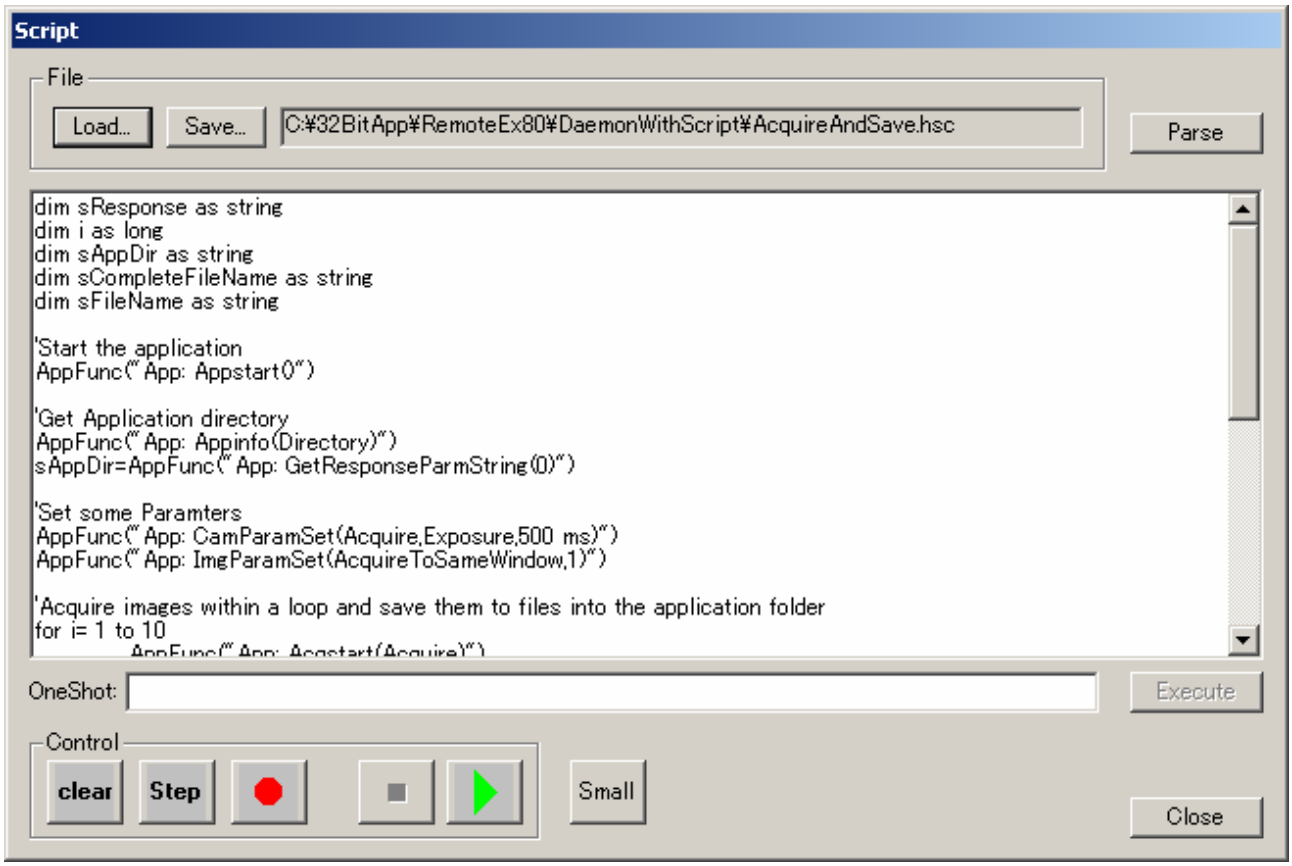
The RemoteEx program can run script files. For this purposes it uses a script engine which is provided in two DLLs (ScrEngUI.dll and ScrptEng.dll). The Syntax of this script language is described in the file ScriptConstruction40305A_US.xls. The script language can call three different types of commands:

1.) Keywords of the script language itself (like “For”, “Next”, “dim”, “CStr” and the like)
2.) Commands of the RemoteEx command set (like “AppStart(), CamParamSet()” etc.)
3.) Command provided from the RemoteEx which are provided for the Script language only (like IsEqual or Format or JoinPathAndFileName)

To run a scrip file click to the Open Script pushbutton



The ScriptFile Editor will appear:



With this Script editor you can Load and Save Script files, execute the script files either in steps or continuously and edit your scripts. While executing the script the commands are transferred to the HiPic or HPDTA and are executed. If the Application has been started visible you can observe the progress of the script as well as on the script editor, the RemoteEx and the HiPic or HPDTA windows.

Special functions provided for the Script

`Format(`*`expression,format`*`)`
This function returns a formatted number, where expression is the number and format the format specifier.

Example:
Format(1,"0000")
Return Value: "0001"

Format specifier

Symbol	Description
0	Digit placeholder; prints a trailing or a leading zero in this position, if appropriate.
#	Digit placeholder; never prints trailing or leading zeros.
.	Decimal placeholder.
,	Thousands separator.
– + $ ( ) space	Literal character; characters are displayed exactly as typed into the format string.

Examples:

Format **syntax**	**Result**
Format (8315.4, "00000.00")	08315.40
Format (8315.4, "#####.##")	8315.4
Format (8315.4, "##,##0.00")	8,315.40
Format (315.4,"$##0.00")	$315.40

You can also use named formats as follows

Named Format	**Description**
General Number	Displays number with no thousand separator.
Currency	Displays number with thousand separator, if appropriate; display two digits to the right of the decimal separator. Output is based on user's system settings.
Fixed	Displays at least one digit to the left and two digits to the right of the decimal separator.
Standard	Displays number with thousand separator, at least one digit to the left and two digits to the righseparator.
Percent	Multiplies the value by 100 with a percent sign at the end.
Scientific	Uses standard scientific notation.

See the MSDN Library documentation for more information about formatting numbers.

`StrVal(`*`sValue`*`)`
This function returns the value represented by the string sValue. Both comma and decimal point is accepted as the decimal delimiter.

`StrLeft(`*`iCount,String`*`)`
This function returns iCount characters of the left side of String.

`StrRight(`*`iCount,String`*`)`
This function returns iCount characters of the right side of String.

`StrLen(`*`String`*`)`
This function returns the length of String in characters.

`WriteToFile(`*`FilePath,iMode,sData`*`)`
This function writes the string sData to the file Filepath and adds a < CR > <LF>. The string sData can contain commas as well. The function behaves different according the value of iMode:
0=Write to file but don't overwrite if file exits
1=Write to file and overwrite if file exist
2=Append to file (create if not exist)

`JoinPathAndFileName(`*`Path,File`*`)`
This function joins a path and a file statement with eventually adding a backslash if necessary.

IsEqual(*String1, String2*)
This function performs a text based compare and the two strings. If they are equal (case insensitive) the function returns 1 if not it returns 0.

GetResponseString()
This function returns the complete string which has been returned by the previous application command.

GetResponseCountLong()
This function returns the number of parameters which has been returned by the previous application command.

GetResponseParmString(*Index*)
This function returns a single parameter (specifying the index of the parameter with the parameter ***Index***) which has been returned by the previous application command in string format.

GetResponseParmBool(*Index*)
This function returns a single parameter (specifying the index of the parameter with the parameter ***Index***) which has been returned by the previous application command in bool format.

GetResponseParmByte(*Index*)
This function returns a single parameter (specifying the index of the parameter with the parameter ***Index***) which has been returned by the previous application command in byte format.

GetResponseParmLong(*Index*)
This function returns a single parameter (specifying the index of the parameter with the parameter ***Index***) which has been returned by the previous application command in Long format.

Sample Script files

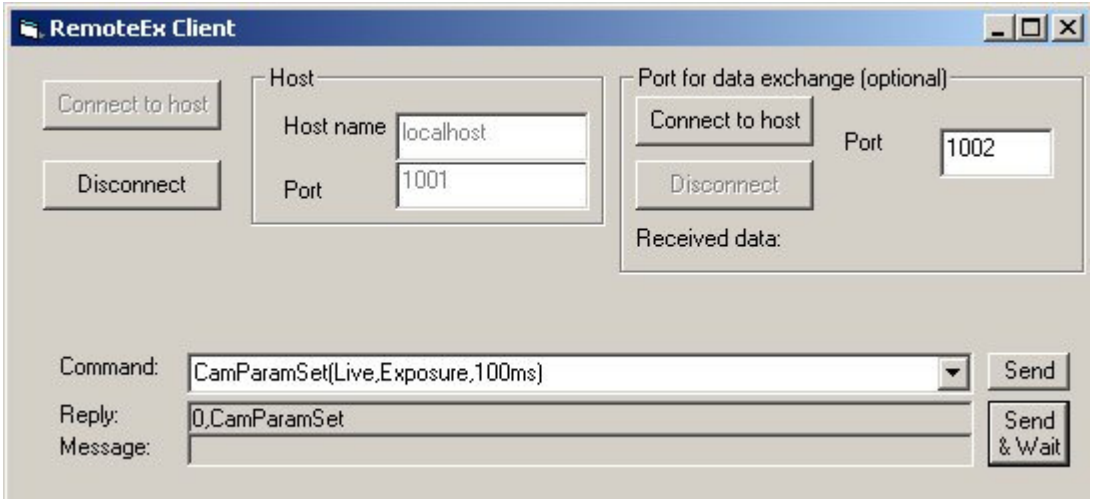
To learn how to use the Scrip language and how to use the RemoteEx as a total there are several script sample files. The samples uses the following functions:

Sample File	Topic	Used Functions
AcqParms.hsc	Set and get Acquisition parameters	Appstart, AcqParamGet, GetResponseParmString, AcqParamInfo, GetResponseParmLong, AcqParamSet
AcquireAndSave.hsc	Acquire Images and save them	Appstart, Appinfo, GetResponseParmString, CamParamSet, ImgParamSet, Acqstart, Format, JoinPathAndFileName, imgsave, Acqstatus, GetResponseParmString, IsEqual
AppInfo.hsc	Get Information about the application	AppInfo, GetResponseParmString, Appstart
Background.hsc	Execute background correction	Appstart, Appinfo, GetResponseParmString, CamParamSet, ImgParamSet, Acqstart, JoinPathAndFileName, imgsave, CorParamSet, CorDoCorrection, LutSetAuto, Acqstatus, IsEqual
CamParms.hsc	Set and get Camera parameters	Appstart, CamParamGet, GetResponseParmString, CamParamInfo, CamParamSet
ImageStatus.hsc	Gets and modifies the Image status	Appstart, Appinfo, GetResponseParmString, CamParamSet, ImgParamSet, Acqstart, ImgStatusGet, ImgStatusSet, ImgDataInfo, StrLen, StrRight, JoinPathAndFileName, WriteToFile, Acqstatus, IsEqual
Sequence.hsc	Uses Sequence acquisition	Appstart, Appinfo, GetResponseParmString, SeqParamSet, Seqstart, JoinPathAndFileName, seqsave, Seqstatus, IsEqual
StartAndStop.hsc	Starts and Stops the application	AppStart, Acqstart, AppEnd, Acqstatus, GetResponseParmString, IsEqual

RemoteExClient sample

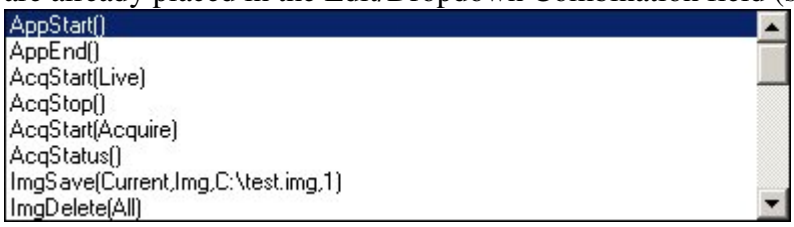
General

Basically it is the target of the RemoteEx to give the user a possibility to use the HiPic or HPD-TA from his application. Therefore it is the intention that the client program is always a code which is made by the customer under his responsibility. However, during evaluation and development it may be helpful to have a sample program which does some of the tasks which later on will be done by the customer's client program.
Therefore the program RemoteExClient.exe is provided with the HiPic or HPD-TA. It is strictly speaking not part of the RemoteEx program and we do not guarantee its proper operation.
The upper part of this dialog deals with general initialization and sending commands individually (see below screen shot), whereas the lower part provides some samples of how to transfer data to the client.



TCP-IP ports and sending commands

In the upper part of the RemoteExClient program (shown above) the user can input the host name and TCP-IP port number of the main communication TCP-IP communication and optional the TCP-IP port number for data exchange.
The edit box labeled "command" can be used to enter commands which can be sent to the RemoteEx with the "Send" or "Send & Wait" pushbutton. The "Send & Wait" waits until the correct response is returned from the RemoteEx. To make live more easy a small set of commands are already placed in the Edit/Dropdown Combination field (see following screenshot).

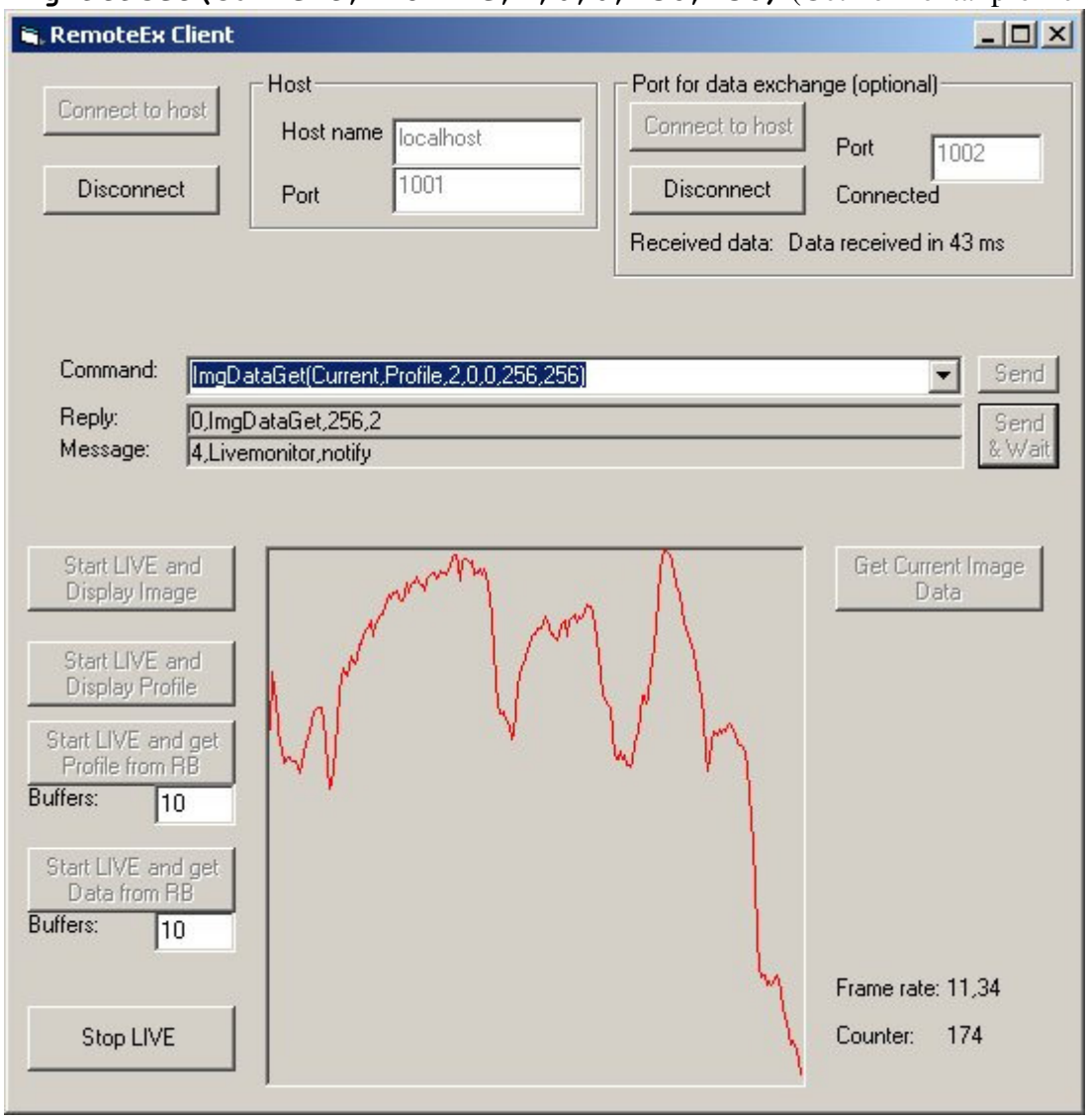


Responses and Messages are displayed below the command line.

Transferring profile data

One task which can be done is to transfer profile data to the client program. This can be done by using the pushbutton "Start LIVE and Display profile" (see screenshot).
If the second TCP-IP port is connected, data is transferred by this port. If this port is not connected data is transferred by saving it to a file in the RemoteEx program and loaded by the client program.
In principle the following RemoteEx commands are used

`AppStart(Live)` (start the live mode)
`AcqLiveMonitor(Notify)` (Start Live monitor to notify whenever a new live image appears)
`ImgDataGet(Current,Profile,2,0,0,256,256)` (Get horizontal profile data)



If the program should continuously process profile data it is recommended to get a new profile data whenever a new message from the AcqLiveMonitor(Notify) has arrived.
The sample program displays the profile data whenever a new profile has been arrived.
When no TCP-IP connection is available data save is done by the following command instead:

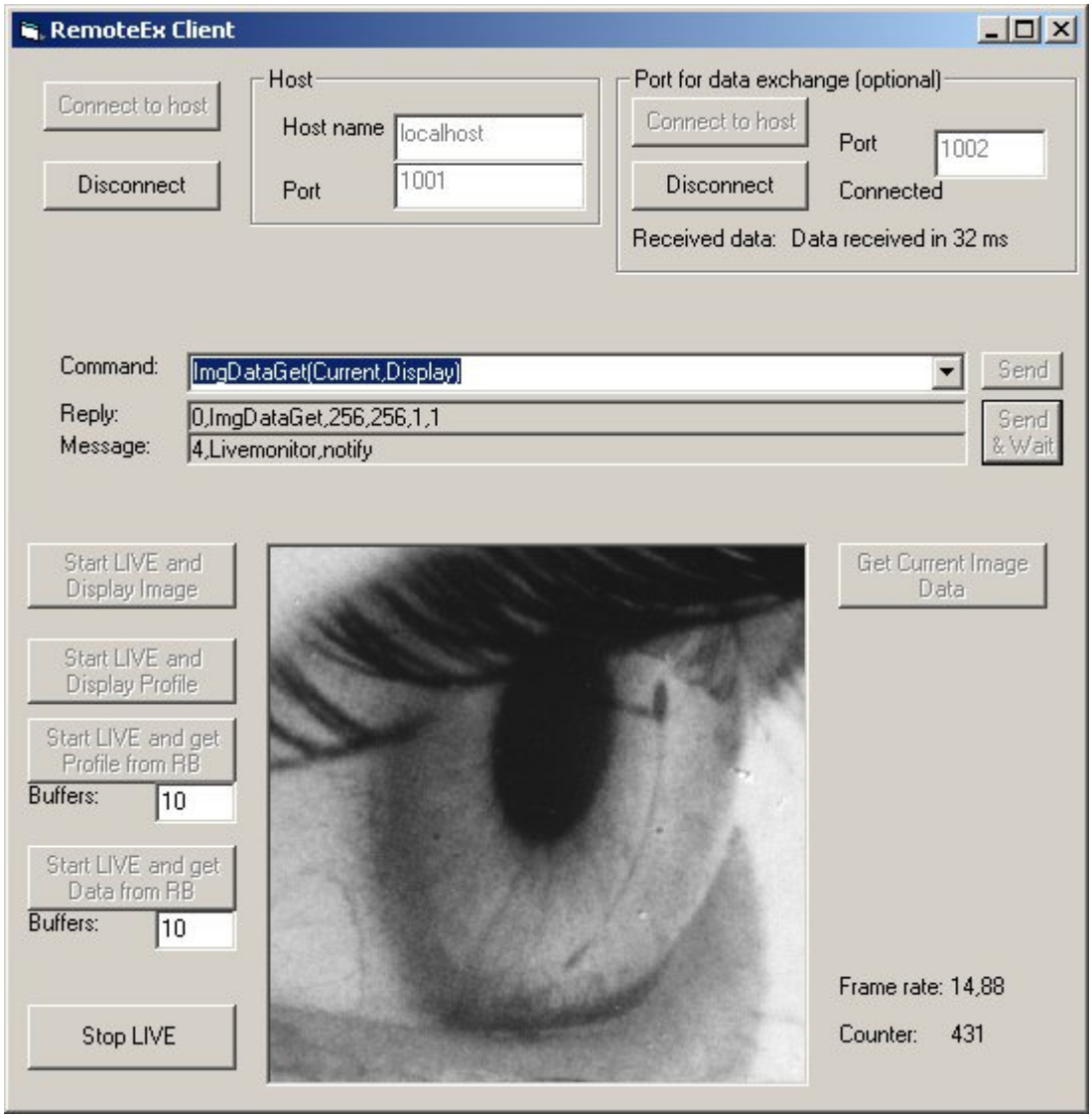
`ImgDataDump(Current,Profile,2,0,0,256,256,C:\program files\hipic\hipic820\RemoteEx_Reserved_Imagefile.dat)`

Showing LIVE display

If display data should be transferred in real time the following commands can be used:

`ImgDataGet(Current,Display)` (use TCP-IP second port) or
`ImgDataDump(current,display,C:\program files\hipic\hipic820\RemoteEx_Reserved_Imagefile.dat)` (use file save and load method)

The image display data is immediately displayed on the client dialog by the sample program.



Transferring profile or image data with ring buffer

The previous methods always get the latest available image, where it is irrelevant whether all frames are transferred or not.
If, however, it is important that data of all frames are transferred to the client a ring buffer can be setup at the RemoteEx and the data of every frame can be requested individually. Of course this works only if the speed of the network is fast enough to transfer all data.

For this purpose the Live Monitor option RingBuffer can be used. In this mode the RemoteEx application copies all data to a ring buffer of the specified number of buffers once Live mode has been started. If all buffers are filled the RemoteEx starts to write data to the first buffer. A global counter (we call it “seqnumber”) is maintained which is increased by one with every new live image. This counter is then associated with the buffer where the data is stored. This counter starts from zero if AcqLiveMonitor(RingBuffer)is called or if the acquired image size changes (Because the buffers have to be reallocated). By using seqnumber the client program has access to all images which are not yet overwritten.
Example: If the number of buffers is 10 then the images 0 to 9 are written to buffers 0 to 9. After this data write restart with buffer zero, which means that Buffer 0 is overwritten by image 10 and so on. The following diagram shows this mechanism.

SeqNr	0	1	2	3	4	5	6	7	8	9
Buffer	0	1	2	3	4	5	6	7	8	9

SeqNr	10	11	12	13	14	15	16	17	18	19
Buffer	0	1	2	3	4	5	6	7	8	9

SeqNr	20	21	22	23						
Buffer	0	1	2	3	4	5	6	7	8	9

In our Example the current situation is now as follows:
The last Acquired Image is in Buffer 3 and has the SeqNumber 23. Images 14 to 23 are still available to get. Data from Live Image Nr 18 for example can now be transferred to the client with the following command:

`imgRingBufferGet(Data,18)` or
`imgRingBufferGet(Profile,2,0,0,256,256,18)`

if a file name is specified after the seqNumber parameter the data is saved to a file like in the following examples:

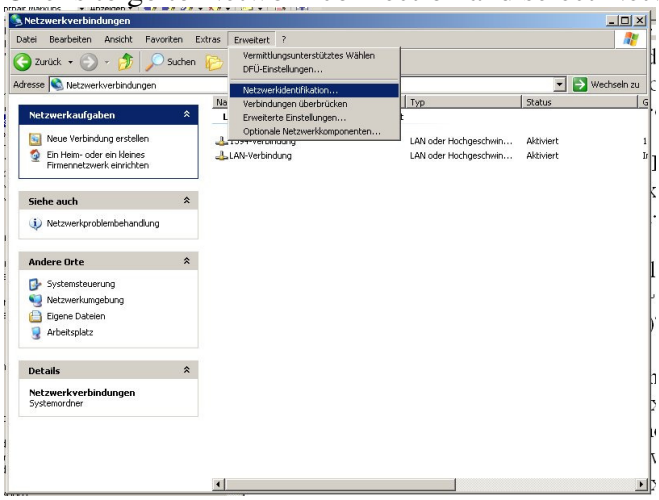
`imgRingBufferGet(Data,18, C:\program files\hipic\hipic820\RemoteEx_Reserved_Imagefile.dat)` or
`imgRingBufferGet(Profile,2,0,0,256,256,18,C:\program files\hipic\hipic820\RemoteEx_Reserved_Imagefile.dat)`

High speed data transfer

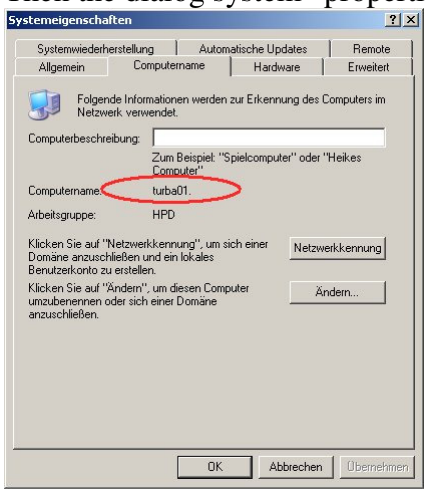
Please note that all memory transfer functions are using an ActiveX communication to transfer data from the application to the RemoteEx which is considerably slower then copying data inside the same process space. Therefore this function may not result in the desired frame rates if a high speed camera is used (like the Hamamatsu C9300). If you need to acquire data with much higher frame rate it is recommended to use the sequence functions of the main application (which can be started from the RemoteEx client) or the user function (which has direct access to the image memory).

Identifying the Host name

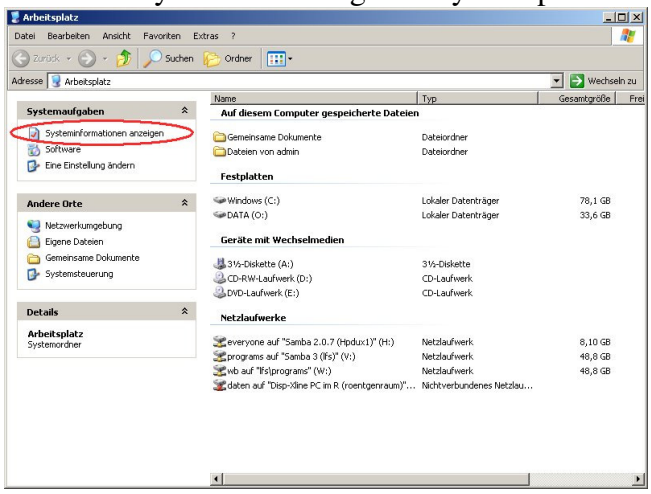
There are several ways to identify the remote computers name.
One is to go to Network connection and select Network-identification.



Then the dialog system “properties appears” and you can see the computer’s name.



Another way to do so is to go to My Computer and select show “system informations”.



To use the RemoteEx on the same computer the host name “localhost” can always be used.