HAMAMATSU
PHOTONICS DEUTSCHLAND GMBH

HiPic/HPD-TA RemoteEx Programmers Handbook

HiPic/HPD-TA Version 9.4

WT, HPD, Document Date 18.08.2016

Table of content

Introduction

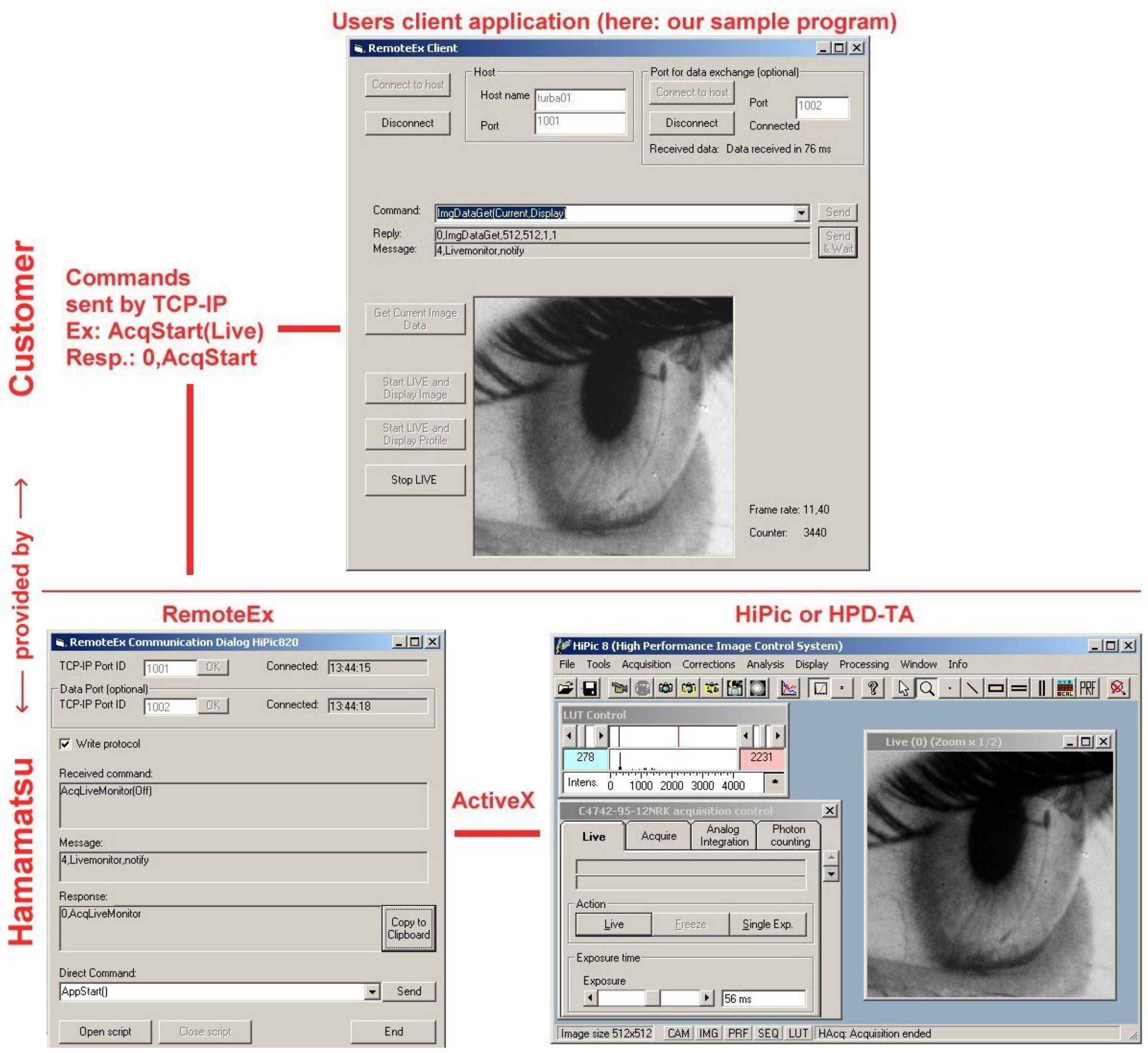
About this manual

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.



Host name: turba01
Port: 1001
Port: 1002
Received data: Data received in 76 ms
Command: ImgDataGet(Current,Display)
Reply: 0,ImgDataGet,512,512,1,1
Message: 4,Livemonitor,notify
Frame rate: 11,40
Counter: 3440

TCP-IP Port ID: 1001 Connected: 13:44:15
TCP-IP Port ID: 1002 Connected: 13:44:18
Received command: AcqLiveMonitor(Off)
Message: 4,Livemonitor,notify
Response: 0,AcqLiveMonitor
Direct Command: AppStart()

278 2231
0 1000 2000 3000 4000
Exposure: 56 ms
Image size 512x512
HAcq: Acquisition ended

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 their 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 is:

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)

Encoding
From version 9.3 the RemoteEx can interpret and send strings in three different encodings:
ANSI by using the currently selected codepage, ASCII and UTF-8. If only characters with character code from 0 to 127 are used all three encodings are identical. Using only English texts this condition is always fulfilled.

Language dependent User I/F
From version 9.3 the HiPic can have a different language dependent User I/F also called domestic language. The commands, parameter values and responses, however are independent of the domestic language. In other words: a RemoteEx program working if English language is selected will also work if another domestic language is selected. If the parameter fNoDialogs of the command AppStart() is set to true, messages are always sent in English language independently of the selected domestic language (new from version 9.3). The only case where Unicode characters can appear in the TCP-IP communication is when we transfer filenames and comments appearing in the image status. Such Unicode characters can appear in the content of text boxes and display field. Example: The command **`CorParamSet(Background,GeneralFile,C:\てすと.img)`** sets the background file to the Japanese filename “C:\てすと.img”. If the current codepage does not support Japanese characters the only way to transfer this filename correctly is to modify the encoding to UTF-8 (Be sure to interpret the strings also at the client side properly).

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 versions before 8.4.0 pf9 this file is written to the same directory as the RemoteEx executable, which is normally the HiPic or HPDTA application directory.
In version 8.4.0 pf9 or later this directory is the so called ProgDataDir, which is dependent on the Operating system. See the command AppInfo(ProgDataDir) for details. If an ini-file is specified through the command line parameter /ini=inifile, the same directory as the inifile is used to save the protocol.

```
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	(Sent from RemoteEx User I/F)
DCR:	Response from direct command	(Received through RemoteEx User I/F)
TCM:	command sent by TCP-IP	(Sent from Client program)
TCR:	Response received by TCP-IP	(Received by Client program)
DAR:	Data received on second port	

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

Asynchronous commands

Some commands are just starting an action. While the RemoteEx command returns the response properly, which indicates that the command has been successfully performed, the action still executes and may run an indeterminate time. Such commands are the RemoteEx commands "AcqStart", "SeqStart", "SeqSave" and "SeqLoad". After these commands have been issued it may take a small amount of time until the action has been really started. Therefore there are four phases in such command:

1.) Before the command has been sent or recognized by the RemoteEx
2.) The command has been recognized, but the corresponding action has not yet started
3.) The corresponding action has been started
4.) The corresponding action has been ended

There are dedicated function to inquire the status of the corresponding action like AcqStatus or SeqStatus, however these actions reports an activity only during case 3). To wait until an action has been started or until it has been ended it is therefore difficult with the AcqStatus or SeqStatus. These commands may report idle but it is not clear whether the action is in phase 2.) or 4.) And also of waiting for the start of an action it is not clear if the action is already finished or didn’t even start.
For this purpose the command “AsyncCommandStatus” returns three different status flags:
AsyncCommandPending (case 2.) or 3.))
AsyncCommandPreparing (case 2.)
AsyncCommandActive (case 3.)
So if this command is used after the response of the command has been returned there are three cases:

A) The corresponding action has not yet been started (AsyncCommandPreparing=True)
B) The corresponding action has been started (AsyncCommandActive=True)
C) The corresponding action has already been ended (AsyncCommandPending =False)

Single Threaded Application

The RemoteEx in combination with the HiPic/HPD-TA is a single threaded application. This makes it quite simple to avoid some side effects which may come up on multi-threaded applications. This does not mean that no other threads can run but they do not directly interfere with data in the scope of the main application. Especially grabber routines can open up separate threads, but they can only enter the main code if no other thread is running.
Sometimes it is an advantage to serve other events, and therefore the program calls the function DoEvents at well-defined places. For example during a running live mode there is no possibility to respond on incoming RemoteEx commands except in these DoEvent routines. It is important to know that during such DoEvents the currently running main thread is suspended and can only proceed if the functions in the DoEvents have been executed.
To allow such asynchronous commands as described in the previous chapter, Timer events are used. The Timer can only start if no other code is executed. This means that the code starting a command like AcqStart() should end before the Timer event can be fired and the real Acquisition can start.
If you execute an asynchronous command (Command A) and immediately afterwards another command (Command B) which uses a call to DoEvents the Timer Event may be fired from within the DoEvents, which means the command B never can end unless the Command A has finished. This leads to a deadlock.
Of course no RemoteEx programmer knows which commands uses DoEvents, but every command may use it.
Therefore a good way if programming is the following:

1.) Execute the asynchronous command and wait until the response has arrived ("AcqStart", "SeqStart", "SeqSave" and "SeqLoad").
2.) Wait until the asynchronous command signals AsyncCommandPreparing=False. If this is the case the timer which has started the asynchronous command has fired.
3.) Start any other command. In this situation it is without any risk.

Data transfer

Binary image and profile data can be transferred through a separate TCP-IP port by using the commands ImgDataGet or ImgDataGetSubsampling. To handle data transfer correctly it is recommended to proceed in the following order:

1.) Clear all data pending on the port for savety
2.) Send the command ImgDataGet (or ImgDataGetSubsampling)
3.) Wait until all expected bytes have been received on this port.

The time to get all data through TCP-IP may vary considerably and some parts of an image may arrive delayed. If these parts are not handled by a previous function they may arrive during a subsequent call to ImgDataGet and thus lead to a shift in the image content.
So be sure to get all data bytes before proceeding with any other function.

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 than 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 Orca Flash). 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).

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 for compatibility reasons 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,fNoDialogs,iEncoding`***`)`

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.

Note: The following is no longer applicable from version 9.0:
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 (no longer applicable from version 9.0). If this parameter is omitted the default INI file is used.

If ***`fNoDialogs`*** is specified it decides whether the application outputs dialog boxes or whether it sends messages through TCP-IP. If the application is running at a remote computer or if it is running invisible this parameter must be omitted or set to true. If the Application should behave like the standalone program – in other words if it should communicate with the user by normal message boxes - this parameter should be set to False. Default value is true (New from 9.1 pf4). If this parameter is set to true messages are always sent in English language independently of the selected domestic language (new from version 9.3).

`iEncoding` defines the encoding of the strings sent and received by the RemoteEx. There are three possibilities:

“0” or ANSI	ANSI encoding using the currently selected codepage.
“1” or ASCII	ASCII encoding (a 7 bit character set). All other characters are replaced by “?”
“2” or UTF-8 or UTF8	UTF-8 encoding. If Unicode characters should be used this is the best choice.

Default is ANSI. Please note that the first string sent to the RemoteEx is always interpreted as ANSI. Avoid language specific characters in this first command.

`AppEnd()`

This command ends the application.

`AppShowEntry()`

This command is available from version 9.3 pf6. When starting the application with AppStart() the program starts completely without showing the Entry screen. In most cases this is not necessary. However if the configured camera should be checked or the camera selection should be changed the Entry screen (this is the screen which appears when the HiPic is started manually) can be shown in a first step. Then in the next step the program can be started with AppStart(). While the entry screen is shown it is not recommended to start the application by using the OK pushbutton, but it should be started with the AppStart() RemoteEx command.

`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)
`ProgDataDir`	Program data directory (from version 8.4.0 pf9). This is the directory where the application stores important data like the INI file or default image data. This is the subdirectory Hamamatsu\HIPIC or Hamamatsu \HPDTA of the COMMON_APPDATA directory. COMMON_APPDATA is dependent on the Operating system: Windows XP: C:\documents and setting\All Users\Application Data\ Windows 7: C:\ProgramData\ Thus ProgDataDir is: Windows XP, HiPic: C:\documents and setting\All Users\Application Data\Hamamatsu\HiPic\ Windows XP, HPDTA: C:\documents and setting\All Users\Application Data\Hamamatsu\HPDTA\ Windows 7, HiPic: C:\ProgramData\Hamamatsu\HiPic\ Windows 7, HPDTA: C:\ProgramData\Hamamatsu\HPDTA\

Example:

`Appinfo(Version)`

Response:

`0,AppInfo,8.3.0 pf8`

`AsyncCommandStatus()`

This command returns information whether an asynchronous command is currently running and returns some details about this command (Available from 9.3 pf7). The response is:

`0,AsyncCommandStatus,iPending,iPreparing,iActive,sCommand`

The meaning of the parameters are:

`iPending`:	Command is pending (iPending= iPreparing or iActive)
`iPreparing`:	Command has been issued but not started
`iActive`:	Command is executed
`sCommand`:	Command name (if any)

`AppLicenceGet()`

This command returns information about implemented license keys at the application (Available from 9.3 pf7). The response is:

`0,AppLicenceGet,ApplicationKeyFound,LicenceAcquire,LicenceSave,`
`LicenceFitting,LicencePhotonCorr,LicenceTransAbs`

Note: The result of every Key is either 0 (not licence) or 1 (licence found)

Example:

`AppLicenceGet()`

Response: **`0,AppLicenceGet,1,1,1,0,0,0`**

`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 (read only)

MainParamsList()

This command returns a list of all parameters related to main window (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to main window at runtime.
The response returns:
ErrorCode,MainParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **MainParamsList()**
Response: **0,MainParamsList,2,ImageSize,Message**

MainSyncGet()

This command returns the setting of the sync parameter which is available on the HPD-TA main window. It is only available for the HPD-TA (Available from version 9.3 pf6).
The response returns:
ErrorCode,MainParamsList,DoSync,CanSync,IsVisible,Label
DoSync: 0 or 1 indication whether Sync is switched on or off.
CanSync: Indicates whether it is possible to switch on or off sync
IsVisible: The Controls to switch on or off sync are visible
Note: Actual synchronisation takes only place if all three parameters show 1
Label: The label which can be read on the toolbar
Example: **MainSyncGet()**
Response: **0,MainSyncGet,0,0,1,Sync: Off**

`MainSyncSet()`

This command allows to switch the sync parameter which is available on the HPD-TA main window. It is only available for the HPD-TA (Available from version 9.3 pf6).
The format is:
`MainSyncSet(iSwitch)`
`iSwitch:` 0 to switch sync off, 1 to switch sync on.
The response is:
`ErrorCode,MainSyncSet`
Example: **`MainSyncSet(1)`**
Response: **`0,MainSyncSet`**

`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,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.

`GenParamsList()`

This command returns a list of all parameters related to the general options (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to general options at runtime.
The response returns:
ErrorCode,GenParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`GenParamsList()`**
Response: **`0,GenParamsList,2,RestoreWindowPos,UserFunctions`**

Acquisition commands

`AcqStart(`*`AcqMode`*`)`

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

`Live`	Live mode
`SingleLive`	Live mode (single exposure) *(existing, but not documented until version 9.2 pf3)*
`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.`** In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

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.
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)
5=Display	A string which is displayed only (read only)

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. In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

`AcqParamsList()`

This command returns a list of all parameters related to acquisition (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to acquisition at runtime.
The response returns:
ErrorCode, AcqParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`AcqParamsList()`**
Response:
`0,AcqParamsList,8,DisplayInterval,32BitInAI,CCDGainForPC,32BitInPC,MoireeReduction,WriteDPCFile,DefaultDPCFile,AdditionalTimeout`

`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,LiveMonitor,notify (Notify)
4,LiveMonitor,notifytimestamp,*timestamp* (Notify timestamp)
4,LiveMonitor,ringbuffer,*Seqnumber* (RingBuffer)
4,LiveMonitor,*data* (Average,Minimum,Maximum)
4,LiveMonitor,*data0,data1,...* (Profile)

AcqLiveMonitorTSInfo()

This command (available from 8.4.0 pf9) correlates the current time with the timestamp. It outputs the current time and the time stamp. With this information the real time for any other time stamp can be calculated.

The response is:
0,AcqLiveMonitorTSInfo,17:13:46,103745454.743
Note: This functions waits until the next full second to return to provide maximum precisison.

`AcqLiveMonitorTSFormat(Format)`

This command sets the format of the time stamp. It can be one of the following values:

"Timestamp" (or any other string) this is default value. Timestamp in ms from start of the computer	
	Example: 1623448.03354
"DateTime"	Date and time in the format (fixed): yyyy/mm:dd-hh-ss
	Example 2011/06/30-17:33:04
"Unix" or "Linux"	Seconds and μseconds since 01.01.1070
	Example: 354659254:364549

`AcqAcqMonitor(Type)`

The AcqAcqMonitor command starts a mode which returns information on every new image or part image acquired in Acquire/Analog Integration or Photon counting mode (Acquisition monitoring). This function is available from 9.3 pf7. Its behavior is similar to AcqLiveMonitor, which returns information on every new live image. Type can be one of the following:

`Off`	No messages are output. This setting can be used to stop acquisition monitoring.
`EndAcq`	Whenever a complete new image is acquired a message is output.
`EndPart`	For every new part image a message is output. A part is a single image which contributes to a full image. For example in Analog Integration or Photon counting mode several images are combined to give one resulting image.
`All`	For every new image or every new part a message is output.

The response is:

`0,AcqAcqMonitor`

The messages output are:

`4,Acqmonitor,Endacq` (when the a complete image is acquired)

`4,Acqmonitor,Endpart` (when the a part image is acquired)

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`	Parameters on trhe 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
`TDIMode`	TDI Mode (Line sensor, from ver. 8.4.0)
`BinningX`	Binning X direction (Line sensor, from ver. 8.4.0)
`BinningY`	Binning Y direction (Line sensor, from ver. 8.4.0)
`AreaExposureTime`	Exposure time in area mode (Line sensor, from ver. 8.4.0)
`Magnifying`	Use maginfying geometry (Line sensor, from ver. 8.4.0)
`ObjectDistance`	Object Distance (Line sensor, from ver. 8.4.0)
`SensorDistance`	Sensor Distance (Line sensor, from ver. 8.4.0)
`ConveyerSpeed`	Conveyer speed (Line sensor, from ver. 8.4.0)
`LineSpeed`	Line speed (Line sensor, from ver. 8.4.0)
`LineFrequency`	Line frequence (Line sensor, from ver. 8.4.0)
`ExposureTime`	Exposure time in line scan mode(Line sensor, from ver. 8.4.0)
`DisplayDuringMeasurement`	Display during measurement option (Line sensor, from ver. 8.4.0)
`GainTable`	Gain table (Line sensor, from ver. 8.4.0)
`NoOfTimesToCheck`	Number of times to check (Line sensor, from ver. 8.4.0)
`MaximumBackgroundLevel`	Maximum background level (Line sensor, from ver. 8.4.0)
`MinimumSensitivityLevel`	Maximum sensitivity level (Line sensor, from ver. 8.4.0)
`Fluctuation`	Fluctuation (Line sensor, from ver. 8.4.0)
`NoOfIntegration`	Number of Integration (Line sensor, from ver. 8.4.0)
`DualEnergyCorrection`	Dual energy correction method (Line sensor, from ver. 8.4.0)
`LowEnergyValue`	Dual energy correction low energy value (Line sensor, from ver. 8.4.0)
`HighEnergyValue`	Dual energy correction high energy value (Line sensor, from ver. 8.4.0)
`NoofAreasO`	Number of Ouput areas (Line sensor, from ver. 8.4.0)
`AreaStartO1 - AreaStartO4`	Ouput area start (Line sensor, from ver. 8.4.0)
`AreaEndO1 - AreaEndO4`	Ouput area end (Line sensor, from ver. 8.4.0)
`NoofAreasC`	Number of areas for confirmation (Line sensor, from ver. 8.4.0)
`AreaStartC1 - AreaStartC4`	Area for confirmation start (Line sensor, from ver. 8.4.0)
`AreaEndC1 - AreaEndC4`	Area for confirmation end (Line sensor, from ver. 8.4.0)
`SensorType`	Sensor type (Line sensor, from ver. 8.4.0)
`Firmware`	Firmware version (Line sensor, from ver. 8.4.0)
`Option`	Option list (Line sensor, from ver. 8.4.0)
`NoOfPixels`	Number of pixels (Line sensor, from ver. 8.4.0)
`ClockFrequency`	Clock frequency (Line sensor, from ver. 8.4.0)

BitDepth	Bit depth (Line sensor, from ver. 8.4.0)
TwoPCThreshold	Use two thresholds instead of one (DCAM only, from ver. 8.4.0)
AutomaticBundleHeight	Use automatic calculation of bundle height (From ver. 8.4.0)
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)
GenericCamTrigger	Programming of the Trigger (GenericCam only), from 8.4.0 pf10
IntervalTime	Programming of the Interval Time (GenericCam only), from 8.4.0 pf10
PulseWidth	Programming of the Interval Time (GenericCam only), from 8.4.0 pf10
SerialIn	Programming of the Serial In string (GenericCam only), from 8.4.0 pf10
SerialOut	Programming of the Serial Out string (GenericCam only), from 8.4.0 pf10
EnableRS232	Enable RS232 communication (GenericCam only), from 9.3 pf8, and 9.4 pf2
RS232HexInput	HEX input for RS232 communication (GenericCam only), from 9.3 pf8, and 9.4 pf2
RS232CR	Send and receive <CR> for RS232 communication (GenericCam only), from 9.3 pf8, and 9.4 pf2
RS232LF	Send and receive <LF> for RS232 communication (GenericCam only), from 9.3 pf8, and 9.4 pf2
RS232RTS	Use RTS handshake for RS232 communication (GenericCam only), from 9.3 pf8, and 9.4 pf2
AlternateTrigger	Use alternate trigger (GenericCam only), from 9.3 pf8, and 9.4 pf2
NegativeLogic	Use negative trigger (GenericCam only), from 9.3 pf8, and 9.4 pf2
DataValid	Data valid (GenericCam only), from 9.3 pf8, and 9.4 pf2
ComPort	Com port (GenericCam only), from 9.3 pf8, and 9.4 pf2
DataBit	Data Bit (GenericCam only), from 9.3 pf8, and 9.4 pf2
XMaxArea	Max Area in X-Direction (GenericCam only), from 9.3 pf8, and 9.4 pf2
YMaxArea	Max Area in Y-Direction (GenericCam only), from 9.3 pf8, and 9.4 pf2
OutputMode	Output mode (GenericCam only), from 9.3 pf8, and 9.4 pf2
TapConfiguration	Tap configuration (GenericCam only), from 9.3 pf8, and 9.4 pf2
Mode0	Mode0 (GenericCam only), from 9.3 pf8, and 9.4 pf2
Mode1	Mode1 (GenericCam only), from 9.3 pf8, and 9.4 pf2
Mode2	Mode2 (GenericCam only), from 9.3 pf8, and 9.4 pf2
RS232Baud	Baud rate for RS232(GenericCam only), from 9.3 pf8, and 9.4 pf2
AdditionalData	Additional data (GenericCam only), from 9.3 pf8, and 9.4 pf2
CameraInfo	Camera info text (from version 9.0)

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
Threshold2	Second photon counting threshold (in case two thresholds are available) (available from version 8.4.0).
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(Location,`*`Parameter`*`) /`
`CamParamInfoEx(Location,`*`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(Setup,Timingmode)`
Response:
`0,Camparaminfo, Timingmode,Internal Timing,2`
Example2:
`CamParamInfo(Live,NrExposures)`
Response:
`0,camparaminfo,# of exposures,1,1,1,1000000000`

The response consists of the following parts separated by commas:

Meaning	Value
Errorcode	`0`
CommandName	**`camparaminfo`**
Label	**`NrExposures`**
Current value	**`1`**
Parameter Type	**`1`**
Minimum (numerical type only)	**`1`**
Maximum (numerical type only)	**`1000000000`**

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.
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)
5=Display	A string which is displayed only (read only)

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. In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

`CamParamsList(Location)`

`Location` can be one of the following:
This command returns a list of all camera parameters of the specified location (Available from version 9.3 pf6). This command can be used to build up a complete parameter list for the corresponding camera at runtime.

`Setup`	Parameters on 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

The response returns:
ErrorCode, CamParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`CamParamsList(Live)`**
Response: **`0,CamParamsList,6,Exposure,Gain,DoRTBacksub,DoRTShading, RecurNumber,RecurFilter`**

`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)

`CamSetupSendSerial()`

This command (available from version 8.4.0 pf10) sends a command to the camera if this is a possibility in the Camera Options (This is mainly intended for the GenericCam camera). The user has to write the string to send in the correct edit box and can then get the command response from the appropriate edit box.

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)

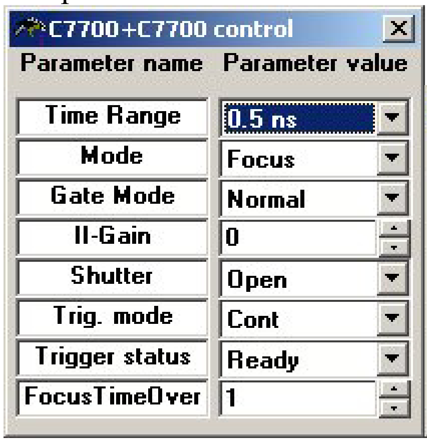
`Location` can be one of the following:

`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, the name of the device, plugin or option (from version 9.1 pf4) or a parameter from the respective options dialog (from version 9.1 pf4).

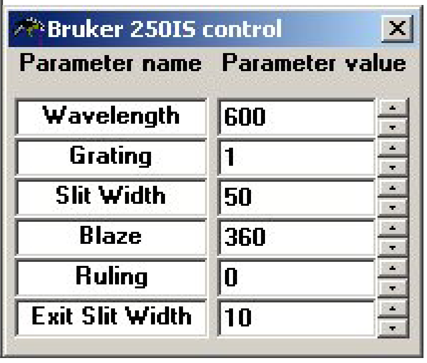
From Status control box

The parameter should be written as indicated in the Parameter name field.



C7700+C7700 control

Parameter name	Parameter value
Time Range	0.5 ns
Mode	Focus
Gate Mode	Normal
II-Gain	0
Shutter	Open
Trig. mode	Cont
Trigger status	Ready
FocusTimeOver	1



Bruker 250IS control

Parameter name	Parameter value
Wavelength	600
Grating	1
Slit Width	50
Blaze	360
Ruling	0
Exit Slit Width	10

Example:

`DevParamGet(TD,Time Range)`

Response:

`0,DevParamGet,0.5 ns`

`or`

`DevParamGet(Spec,Wavelength)`

Response:

`0,DevParamGet,600`

Device, Plugin or option name

This function also allows to get information about the device name, plugin name and option name of these devices. The following keywords are available:

DeviceName, PluginName, OptionName1, OptionName2, OptionName3, OptionName4

From Device options:
Additionally to the parameters from the status/control boxes the user can get or set also the following parameters from the Device options:

From Streak options:
AutoMCP, AutoStreakDelay, AutoStreakShutter, DoStatusRegularly, AutoActionWaitTime

From Spectrograph options:
AutoSpecShutter

From Delay options:
AutoDelayDelay

From Delay 2 Options:
AutoDelay2Delay

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(*Location*, *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

Auxiliary devices commands

All commands available from version 8.4.0

The following commands refer to auxiliary devices which can be configured in the auxiliary devices options dialog. The devices currently supported are all Hamamatsu Microfocus X-ray sources (MFX) and all gated image intensifiers (C9016, C5946, C5947, C5948 and C01880) (from version 9.3 pf4).

`AuxDevsParamGet(`***`Parameter`***`)`

This command gets values in combination with auxiliary devices.

`Parameter` can be one of the following:

From the options dialog

`MFXConnect`	Controls the Connection to the MFX
`MFXShowControl`	Controls whether to show or not the MFX control dialog
`MFXComPort`	MFX com port
`MFXBaud`	MFX Baud rate
`MFXSetupMessage`	Message shown after the MFX is connected
`MFXDirectControlIn`	Input for direct control to MFX
`MFXDirectControlOut`	Output from direct control from MFX (Form version 8.4.0 pf10 this command does no longer output the attached <CR> character)
`MFXWriteProtocol`	Controls whether or not the program writes a protocol of MFX activities (from version 9.3 pf4)
`MFXWriteSerialProtocol`	Controls whether or not the program add the serial communication to the protocol of MFX activities (from version 9.3 pf4)
`MFXSerialTimeout`	Serial command communication timeout to the MFX (from version 9.3 pf4)
`MFXProtocolFileName`	File name of the protocol of MFX activities (from version 9.3 pf4)
`IIConnect`	Controls the Connection to the Image intensifier (II) (from version 9.3 pf4)
`IIWriteSerialProtocol`	Controls whether or not the program writes a protocol of II activities (from version 9.3 pf4)
`IIBaud`	II Baud rate (from version 9.3 pf4)
`IIComPort`	II com port (from version 9.3 pf4)
`IIType`	Type of the II (from version 9.3 pf4)
`IISerial`	Serial number of the II (from version 9.3 pf4)
`IIDirectControlIn`	Input for direct control to II (from version 9.3 pf4)
`IIDirectControlOut`	Output from direct control from II (from version 9.3 pf4)

From the MFX control dialog

`MFXXRayWarning`	MFX Warning display
`MFXActVoltage`	Actual MFX voltage
`MFXSetVoltage`	Set MFX voltage
`MFXActCurrent`	Actual MFX current
`MFXSetCurrent`	Set MFX current
`MFXFocusMode`	MFX Focus mode control
`MFXStatus`	MFX Status
`MFXControlMessage`	MFX control message
`MFXPulseWidth`	MFX pulse width(from version 9.3 pf4)
`MFXPulseMode`	MFX pulse mode (from version 9.3 pf4)

From the Image Intensifier (II) control dialog

IIGateWarning	(from version 9.3 pf4)
IIGateInfo	(from version 9.3 pf4)
IIGainMonitor	II Warning display (from version 9.3 pf4)
IIMCPGain	II gain (from version 9.3 pf4)
IIWarningLevel	II warning level (from version 9.3 pf4)
IILongExposure	II long exposure (MXP gating) (from version 9.3 pf4)
IILED	II LED (from version 9.3 pf4)
IIAutoPowerOff	II auto power off (from version 9.3 pf4)
IIAPThreshold	II auto power off value (from version 9.3 pf4)
IIGateMode	II gate mode (from version 9.3 pf4)
IIPolarity	II input polarity (from version 9.3 pf4)
IIPhosophorScreenCurrent	II phosphor screen current (from version 9.3 pf4)
IIProtectionLevel	II protection level (from version 9.3 pf4)
IIMessage	II control message (from version 9.3 pf4)

From the Image Intensifier Trigger Settings (IT) control dialog

ITRepCount	II repetition count for multigate (from version 9.3 pf4)
ITGateTime	II gate time (from version 9.3 pf4)
ITGateDelay	II gate delay (from version 9.3 pf4)
ITRepInterval	II repetition interval for multigate (from version 9.3 pf4)
ITTriggerInterval	II trigger interval (from version 9.3 pf4)
ITShutterRepError	II shutter repetition error (from version 9.3 pf4)
ITTimeSettingRules	II Time setting rules (from version 9.3 pf4)
ITTriggerCycleTime	II trigger cycle time (from version 9.3 pf4)
ITToNextTrigger	II time to next trigger (from version 9.3 pf4)
ITShutterTimeSettingError	II shutter time setting error (from version 9.3 pf4)
ITTriggerDetected	II trigger detected (from version 9.3 pf4)

AuxDevsParamSet(*Parameter*,*Value*)

This command sets the specified parameter in combination with auxiliary devices. Possible values for ***Parameter*** are described above**.**

AuxDevsParamInfo(*Parameter*) / AuxDevsParamInfoEx(*Parameter*)

This command gets information about the specified parameter in combination with auxiliary devices.
AuxDevsParamInfoEx returns more detailed information in case of a list parameter (Parameter type = 2) than **AuxDevsParamInfo.**

AuxDevsParamsList()

This command returns a list of all parameters related to auxiliary devices (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to auxiliary devices at runtime.
The response returns:
ErrorCode, AuxDevsParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **AuxDevsParamsList()**
Response:
0,AuxDevsParamsList,18,MFXConnect,MFXShowControl,MFXComPort,MFXBaud,MFXSetupMessage,MFXDirectControlIn,MFXDirectControlOut,MFXWriteProtocol,MFXSerialTimeout,MFXProtocolFileName,IIConnect,IIWriteSeri

alProtocol,IIBaud,IIComPort,IIType,IISerial,IIDirectControlIn,IIDirectControlOut

`AuxDevsDoXOn()`
Switches on X-Ray radiation.

`AuxDevsDoXOff()`
Switches off X-Ray radiation.

`AuxDevsDirectControlSend()`
Sends the control command specified with the **`MFXDirectControlIn`** parameter. After sending the command the parameter **`MFXDirectControlOut`** contains the response (After sending the command you have to wait an appropriate time until the response is returned).

`AuxDevsDoIIOn`
Switches on II gated. (from version 9.3 pf4)

`AuxDevsDoIIOff`
Switches off II gated. (from version 9.3 pf4)

`AuxDevsDoIIGetStatus`
Gets the status from the II (from version 9.3 pf4)

`AuxDevsDoIIResetProtection`
Resets the protection circuit if triggered (from version 9.3 pf4)

`AuxDevsDoIITriggerSettings`
Shows the II trigger settings (from version 9.3 pf4)

`AuxDevsIIDirectControlSend`
Sends the control command specified with the **`IIDirectControlIn`** parameter. After sending the command the parameter **`IIDirectControlOut`** contains the response. (After sending the command you have to wait an appropriate time until the response is returned). (from version 9.3 pf4)

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
`SubtractWithOpenShutter`	Take Background with open shutter (From 9.1 pf4). If the CCD camera doesn't have a shutter this option is labeled "Don't prompt the user during background subtraction"
`BackFilesForAcqModes`	Individual background files for every acquisition mode
`GeneralFile`	Correction file for all acquisition modes (available from version 8.4.0)
`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`	Source for real-time background subtraction This feature has been deleted from version 8.1, but it is (again) new from 9.1 pf4
`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(Location,`*`Parameter`*`) /`
`CorParamInfoEx(Location,`*`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.`** In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

`CorParamsList(`*`Location`*`)`

This command returns a list of all camera parameters of the specified location (Available from version 9.3 pf7). This command can be used to build up a complete parameter list for the corresponding camera at runtime.

`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

The response returns:
ErrorCode, CorParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`CorParamsList(Background)`**
Response:

`0,CorParamsList,12,BackgroundSource,SubtractWithOpenShutter,BackFilesForAcqModes,GeneralFile,LiveFile,AcquireFile,AIFile,DefaultRTBSFile,Constant,ClipZero,AutoBacksub,RTBSSource`

`CorDoCorrection(`*`Destination,Type,sCorrFile`*`)`

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
`Background`	Background subtraction (same as **`Backsub`**)
`Shading`	Shading correction
`Curvature`	Curvature correction
`BacksubShading`	Background subtraction + Shading correction
`BacksubCurvature`	Background subtraction + Curvature correction
`BacksubShadingCurvature`	Background subtraction + Shading correction + Curvature correction
`DefectCorrect`	Defect pixel correction using file ***`sCorrFile`*** (Available from version 8.4.0)

sCorrFile
Defect pixel correction file used for defect pixel correction. (Available from version 8.4.0)

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.

Processing commands

All comands available from version 8.4.0

The following commands refer to the User function, which is a possibility to call DLL functions during acquisition.

`ProcUsfParamGet(`*`Parameter`*`)`

This command gets the values of the User function dialog
`Parameter` can be one of the following:

`FunctionIndex`	Function index
`Parameter1`	Parameter 1
`Parameter2`	Parameter 2
`Parameter3`	Parameter 3
`Parameter4`	Parameter 4
`Cycle`	Cycle number
`Index`	Index number
`ReturnedString`	Returned string

`ProcUsfParamSet(`*`Parameter,Value`*`)`

This command sets the specified parameter of the User function dialog. Possible values for ***`Parameter`*** are described above**.**

`ProcUsfParamInfo(`*`Parameter`*`) / ProcUsfParamInfoEx(`*`Parameter`*`)`

This command gets information about the specified parameter.
`ProcUsfParamInfoEx` returns more detailed information in case of a list parameter (Parameter type = 2) than **`ProcUsfParamInfo.`** In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

`ProcUsfParamsList()`

This command returns a list of all parameters related to User Function (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to User Function at runtime.
The response returns:
ErrorCode, ProcUsfParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`ProcUsfParamsList()`**
Response:
`0,ProcUsfParamsList,8,FunctionIndex,Parameter1,Parameter2,Paramete r3,Parameter4,Cycle,Index,ReturnedString`

`ProcUsfDoUserFunction()`

Executes the user function.

The following functions refers to the arithmetic functions in the processing menu.

`ProcArithParamGet(`*`Parameter`*`)`

This command gets the values of the Arithmetic dialog
`Parameter` can be one of the following:

`Operation`	Opertation to perform
`Constant`	Constant to add

`Type`	Type of arithmetic
`SecondImage`	Source of second image
`ImageInMemory`	Number of used image in memory
`KeepCameraDataType`	Clip to camera data type
`File`	Filename of second file

`ProcArithParamSet(`*`Parameter,Value`*`)`

This command sets the specified parameter of the Arithmetic dialog. Possible values for ***`Parameter`*** are described above**.**

`ProcArithParamInfo(`*`Parameter`*`) / ProcArithParamInfoEx(`*`Parameter`*`)`

This command gets information about the specified parameter.
`ProcArithParamInfoEx` returns more detailed information in case of a list parameter (Parameter type = 2) than **`ProcArithParamInfo.`** In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

`ProcArithParamsList()`

This command returns a list of all parameters related to arithmetic processing (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to arithmetic processing at runtime.
The response returns:
ErrorCode, ProcArithParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`ProcArithParamsList()`**
Response:
`0,ProcArithParamsList,4,Operation,Constant,Type,KeepCameraDataType`

`ProcArithDoProceed()`

Executes the Arithmetic function.

`ProcArithDoInvertImage()`

Executes the Do Invert image function.

`ProcArithDoFlipRotate(`*`Type,Angle`*`)`

Execute flip rotate.
`Type` can be one of the following:
"Flip Horizontal", "Flip Hor", "Horizontal", "Hor"
"Flip Vertical", "Flip ver", "Vertical", "Ver"
"Rotate", "Rot"

`Angle` can be one of the following:
"90 degree", "90 deg", "90"
"180 degree", "180 deg", "180"
"270 degree", "270 deg", "270"

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`**. In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

`DefPixParamsList()`

This command returns a list of all parameters related to defect pixel detection (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to defect pixel detection at runtime.
The response returns:
ErrorCode,DefPixParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`DefPixParamsList()`**
Response:

```
0,DefPixParamsList,12,Method,ImageHotPixel,AverageHotPixel,StandDe
vHotPixel,ThresholdHotPixels,ThresholdHotLines,ImageDeadPixel,Aver
ageDeadPixel,StandDevDeadPixel,ThresholdDeadPixels,ThresholdDeadLi
nes,LineColumnsPercentage
```

`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
`IntensityRuler`	Display intensity rulers (Bird view only) (from 9.4 pf0)
`BirdViewLineThickness`	Line thickness for Bird view display (from 9.4 pf0)
`BirdViewSmoothing`	Smooting for Bird view display (from 9.4 pf0)
`BirdViewScaling`	Intensity scaling for Bird view display (from 9.4 pf0)
`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`**. In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

Example:
`ImgParamInfo(Calibrated)`
Response:
`0,ImgParamInfo,1`   (Calibrated was set to true)

`ImgParamsList()`

This command returns a list of all parameters related to images (Available from version 9.3 pf7).
This command can be used to build up a complete parameter list related to images at runtime.
The response returns:
ErrorCode,ImgParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`ImgParamsList()`**
Response:

```
0,ImgParamsList,11,AcquireToSameWindow,DefaultZoomFactor,WarnWhenU
nsaved,Calibrated,LowerLUTIsZero,AutoLUT,AutoLUTInLive,AutoLUTInRO
I,HorizontalRuler,VerticalRuler,FixedITEXHeader
```

`ImgScalingInfo(ScalingDirection)`

This command returns information about the image scaling (Available from version 9.3 pf7).
ScalingDirection can be one of the following:

`H, Hor, Horizontal or X`	Horizontal Scaling
`V, Ver, Vertical or Y`	Vertical Scaling

The response returns:
Returnvalue,ImgScalingInfo,Type,factor,file
Where

Returnvalue:	0 if succeded, <>0 otherwise
Type:	1=linear scaling, 2=table scaling
Factor:	Scaling factor in case of linear scaling
File:	File name of scaling file or location of scaling data in the file in case of table scaling (Use ImgDataGet(Destination,ScalingTable) to get the scaling table in case of table scaling)

Example: **`ImgScalingInfo(current,y)`**
Response:
```
0,ImgScalingInfo,2,1,No
unit,D:\Testdata\RemoteExClient\SclY_2K.scl
```

`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
`ASCIICAL`	ASCII file with calibration
`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.

`ImgIndexGet()`

Gets the Index of the current image. (available from version 8.4.0). This is necessary for all commands which needs the ***destination*** parameter.

`ImgIndexSet(`*`Index`*`)`

Selects the image with the specified ***Index*** (available from version 8.4.0). This makes the image with the index ***Index*** the current image.

`ImgDefaultDirGet()`

Gets the default directory for the common dialog for File load/save operations (available from version 8.4.0).

`ImgDefaultDirSet(`*`DefaultDirectory`*`)`

Sets the default directory for the common dialog for File load/save operations (available from version 8.4.0).

`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 gets 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)
`ScalingTable`	A profile indicating the scaling values in the case the image has table scaling (4 bytes floating point values), New since version 9.3 pf7

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

If **`Data or Display`** is selected for ***`Type`*** the syntax is:
`ImgDataGet(`*`Destination,Type)`*

If **`Profile`** is selected for ***`Type`*** the syntax is:
`ImgDataGet(`*`Destination`*`,Profile`*`,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.

If **`ScalingTable`** is selected for ***`Type`*** the syntax is:
`ImgDataGet(`*`Destination`*`,ScalingTable`*`,iDirection`*`)`
where ***`iDirection`*** has to be one of the following:

`H, Hor, Horizontal or X`	Horizontal Scaling
`V, Ver, Vertical or Y`	Vertical Scaling

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

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

`ImgDataDump(`*`Destination,Type,Filename`*`)`

This command gets 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)

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

If **`Data or Display`** is selected for ***Type*** the syntax is:
ImgDataDump(*Destination,Type,Filename*)

If **`Profile`** is selected for ***Type*** the syntax is:
ImgDataDump(*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
Example2:
ImgDataDump(current,profile,2,0,0,1024,1024,c:\test.dat)
Response:
0,ImgDataDump,1024,2

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.
Note: The data is transferred by the second TCP-IP port. If this is not opened an error will be issued.

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,timestamp` (Data,Display)
`0,ImgRingBufferGet,NumberOfData,Type,seqnumber,timestamp` (Profile)

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

`ImgAnalyze`***`(Destination,Type,RoiType,`****`iX,iY,iDX,iDY`*****`)`***

This command is used to get analysis information from an image within an ROI (available from version 9.1).

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:

mean	The average (mean) value of the intensity in the specified ROI
sd	The standard deviation of the intensity in the specified ROI
count	The sum of all pixels in the specified ROI
Min	The Minimum value in the specified ROI (available from version 9.3 pf5)
Max	The Maximum value in the specified ROI (available from version 9.3 pf5)
MaxPos	The Maximum value in the specified ROI and it's position (available from version 9.3 pf5)

RoiType can be one of the following:

0	Rectangle ROI.
1	Circular and elliptical type (if the ROI is square it will be circular, otherwise elliptical)

iX,iY,iDX,iDY are the coordinates of the area where to extract the profile.
The response is for ROI types **`mean, sd, count, Min, Max:`**
`0, ImgAnalyze,value,NumberOfPixels`
and for ROI type **`MaxPos:`**
`0, ImgAnalyze,value,NumberOfPixels,XPos,YPos`

Example:
`ImgAnalyze(current,mean,0,0,0,1024,1024)`
Response:
`0, ImgAnalyze,200.93645234,1048576`

`ImgRoiGet`***`(Destination,iRoi)`***

This command is used to get an ROI of the specified image (available from version 9.1).

`Destination` can be one of the following:

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

`iRoi` can be one of the following:

`Selected`	The currently selected ROI. If no ROI is selected this will be ROI0
`A number from 0 to 9`	The specified ROI number

`iRoiType can be one of the following`

`0`	No ROI specified
`1`	Point ROI
`3`	Line ROI
`4`	Rectangle ROI

`iX,iY,iDX,iDY` are the coordinates of the ROI.
The response is:
`0,ImgRoiGet,iRoiType,iX,iY,iDX,iDY`

Example:
`ImgRoiGet(current,0)`
Response:
`0,ImgRoiGet,4,100,100,200,200`

`ImgRoiSet`***`(Destination,iRoi,iRoiType,iX,iY,iDX,iDY)`***
This command is used to set an ROI of the specified image (available from version 9.1).

`Destination` can be one of the following:

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

`iRoi` can be one of the following:

`Selected`	The currently selected ROI. If no ROI is selected Roi 0 is used.
`A number from 0 to 9`	The specified ROI number

`iRoiType can be one of the following`

`0`	No ROI, ROI will be deleted
`1`	Point ROI
`3`	Line ROI
`4`	Rectangle ROI
`5`	Rectangle ROI with full horizontal width. iX and iDX must be same as for the full image. Such ROI will display a horizontal profile if the PRF butten is pressed.
`6`	Rectangle ROI with full vertical width. iY and iDY must be same as for the full image. Such ROI will display a vertical profile if the PRF button is pressed.

The response is:
`0,ImgRoiSet`

Example:
`ImgRoiSet(current,0,4,100,100,200,200)`
Response:
`0,ImgRoiSet`

ImgRoiSelectedRoiGet*(Destination)*

This command is used to get the selected ROI of the specified image (available from version 9.1).

Destination can be one of the following:

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

iRoi can be one of the following:

A number from 0 to 9	The specified ROI number

If no ROI exist 0 is returned.

The response is:
0,ImgRoiSelectedRoiGet,iRoi

Example:
ImgRoiSelectedRoiGet(current)
Response:
0,ImgRoiSelectedRoiGet,0

ImgRoiSelectedRoiSet*(Destination,iRoi)*

This command is used to select an ROI of the specified image (available from version 9.1).

Destination can be one of the following:

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

iRoi can be one of the following:

A number from 0 to 9	The specified ROI number

If the ROI iRoi does not exist nothing happens.

The response is:
0,ImgRoiSelectedRoiSet

Example:
ImgRoiSelectedRoiSet(current,0)
Response:
0,ImgRoiSelectedRoiGet

ImgOverlayGet*(Destination)*

This command is used to get the line coordinates of the overlay of the specified image (available from version 9.1 pf4).

Destination can be one of the following:

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

Color is specified in the format:
color = red + green * 256 + blue * 256 * 256

The response is:
**0,ImgOverlayGet,NrLines,StartX0,StartY0,EndX0,EndY0,Color0,
StartX1,StartY1,EndX1,EndY1,Color1,...**

Example:
`ImgOverlayGet(current)`

Response:
`0,ImgOverlayGet,0` (no overlay drawn)

`ImgOverlaySet`*`(Destination,iNrLines,`*
`StartX0,StartY0,EndX0,EndY0,Color0,`
`StartX1,StartY1,EndX1,EndY1,Color1,etc.)`

This command is used to set the line coordinates of the overlay of the specified image (available from version 9.1 pf4).

`Destination` can be one of the following:

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

The color should be specified in the format:
color = red + green * 256 + blue * 256 * 256

The response is:
`0,ImgOverlaySet`

Example:
`ImgOverlaySet(current,2,200,200,300,400,0,300,400,400,400,255)`
Response:
`0,ImgOverlayGet,0`

`ImgSelectedPixelsGet`*`(Destination,iType,iRoiType,iX,iY,iDX,iDY,`*
`iThreshold,iMaxPixels)`

This command is used to get a list of pixels which follows the given rule(available from version 9.1 pf1).

`Destination` can be one of the following:

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

i**`Type`** can be one of the following:
`AboveThreshold`
`BelowThreshold`

`iROIType:`
0 (Rectangular)
1 (Circle/Elliptical)

`iX,iY,iDX,iDY`
Coordinates of the ROI

`iThreshold`
Threshold value for classification

`iMaxPixels`

Output is limited to this number of pixels
Note: This number is internally limited to 10000

The response is:
`0,ImgSelectedPixelsGet,iNrPixels,iX0,iY0,iX1,iY1,...`

Example:
`ImgSelectedPixelsGet(Current,AboveThreshold,0,100,100,200,200,300,1000)`
Response:
`0,ImgSelectedPixelsGet,3,501,603,501,604,502,604`

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
`DoGaussFit`	Do Gauss fitting to determine the 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.`** In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

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

`QprParamsList()`

This command returns a list of all parameters related to the quick profile (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to quick profile at runtime.
The response returns:
ErrorCode,QprParamsList,NumberOfParameters,Parameter1,…, ParameterN

Example: **`QprParamsList()`**
Response:

```
0,QprParamsList,6,UseMinAsZero,DisplayQPOutOfImage,QPRelativeSpace
,DisplayDirectionForRect,AdjustQPHeight,DisplayFWHM
```

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.`**

`LutParamsList()`

This command returns a list of all parameters related to the LUT (Available from version 9.3 pf7).
This command can be used to build up a complete parameter list related to the LUT at runtime.
The response returns:
ErrorCode,LutParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **`LutParamsList()`**
Response:

```
0,LutParamsList,5,Color,Inverted,Gamma,Linearity,Overflowcolors
```

`LutSetAuto()`

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

```
0,LutSetAuto,174,1601,1
```

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
`NumberOfSamples`	Number of samples (Images or profiles) (available from version 8.4.0)

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**. In case of a numeric parameter (Parameter type = 1) it additionally returns the step width (available from 9.2.0 pf5).

SeqParamsList()

This command returns a list of all parameters related to sequence mode (Available from version 9.3 pf7). This command can be used to build up a complete parameter list related to sequence mode at runtime.
The response returns:
ErrorCode,SeqParamsList,NumberOfParameters,Parameter1,…, ParameterN
Example: **SeqParamsList()**
Response:
0,SeqParamsList,20,AcquisitionMode,NoOfLoops,AcquisitionSpeed,SampleType,CurrentSample,NumberOfSamples,AcquireImages,ROIOnly,StoreTo,AcquireProfiles,AutoCorrectAfterSeq,DisplayImgDuringSequence,PromptBeforeStart,EnableStop,Warning,LoadHISSequence,PackHisFiles,NeverLoadToRAM,WrapPlay,PlayInterval

SeqSeqMonitor(Type)

The SeqSeqMonitor command starts a mode which returns information on every new image or part image acquired in Sequence mode (Sequence monitoring). This function is available from 9.3 pf7. Its behavior is similar to AcqLiveMonitor or AcqAcqMonitor, which returns information on every new live or acquisition image. Type can be one of the following:

Off	No messages are output. This setting can be used to stop acquisition monitoring.
EndAcq	Whenever a complete new image is acquired in sequence mode a message is output.
EndPart	For every new part image in sequence mode a message is output. A part is a single image which contributes to a full image. For example in Analog Integration or Photon counting mode several images are combined to give one resulting image.
All	For every new image or every new part a message is output.

The response is:
0,SeqSeqMonitor

The messages output are:
4,Seqmonitor,Endacq (when the a complete image is acquired)
4,Seqmonitor,Endpart (when the a part image is acquired)

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().
Additionally there are the following types for image types:

`HIS`	HIS sequence (Hamamatsu image sequence)
`DISPLAY2HIS`	HIS sequence (Hamamatsu image sequence) containing only display data (8 bit)

`SeqLoad(ImageType,FileName)`

`ImageType,FileName` are same as described under ImgLoad().
Additionally there are is following type for image types:

`HIS`	HIS sequence (Hamamatsu image sequence)

Response:
`0,SeqLoad,`*`ImageNumber`*
`ImageNumber` describes the image number of the sequence image.

SeqCopyToSeparateImg()

Copies the currently selected image of a sequence to a separate image. (available from version 8.4.0).

SeqImgIndexGet ()

Gets the image index of the sequence. (available from version 8.4.0). This is needed for image functions like **CorrDoCorrection** where we have to specify the ***`Destination`*** parameter.

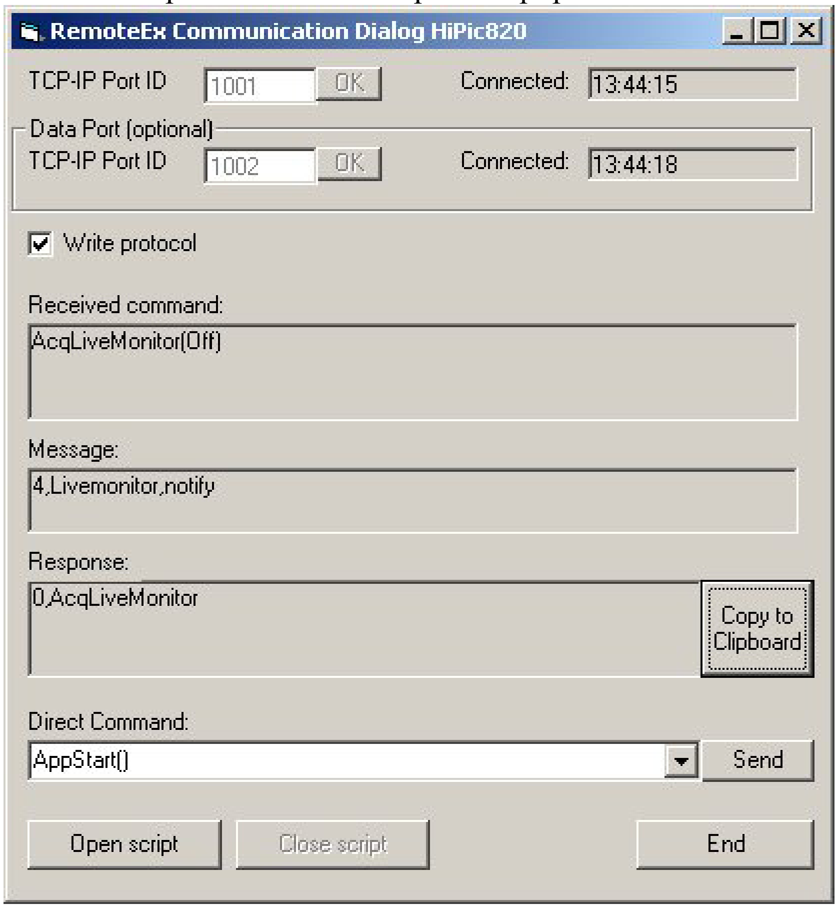
Using Script files

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


RemoteEx Communication Dialog HiPic820

TCP-IP Port ID: 1001 — Connected: 13:44:15

Data Port (optional)
TCP-IP Port ID: 1002 — Connected: 13:44:18

☑ Write protocol

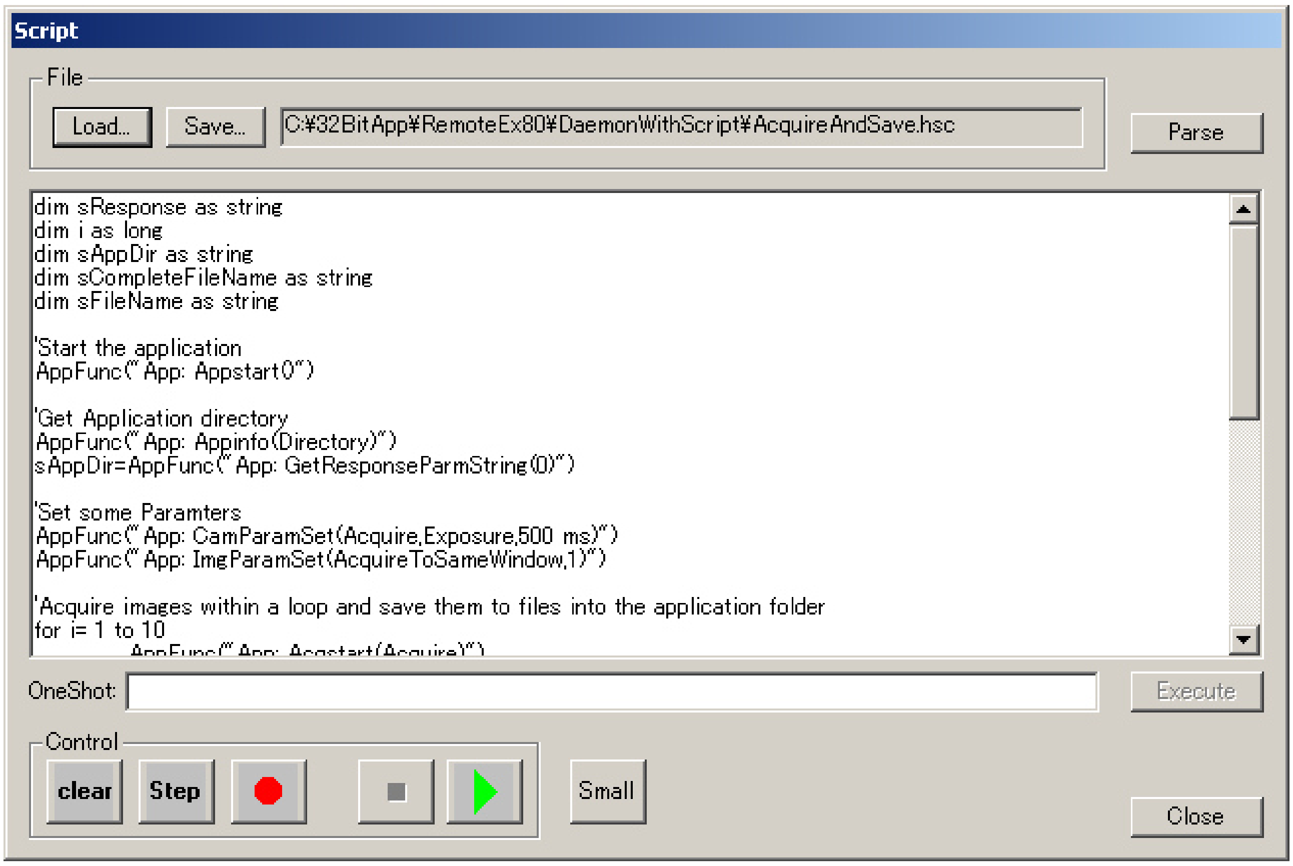
Received command:
```
AcqLiveMonitor(Off)
```

Message:
```
4,Livemonitor,notify
```

Response:
```
0,AcqLiveMonitor
```

Direct Command: AppStart()

The ScriptFile Editor will appear:



```
dim sResponse as string
dim i as long
dim sAppDir as string
dim sCompleteFileName as string
dim sFileName as string

'Start the application
AppFunc("App: Appstart()")

'Get Application directory
AppFunc("App: Appinfo(Directory)")
sAppDir=AppFunc("App: GetResponseParmString(0)")

'Set some Paramters
AppFunc("App: CamParamSet(Acquire,Exposure,500 ms)")
AppFunc("App: ImgParamSet(AcquireToSameWindow,1)")

'Acquire images within a loop and save them to files into the application folder
for i= 1 to 10
        AppFunc("App: Acqstart(Acquire)")
```

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)
The function returns the following values:
0=No Error
1=No valid file name
2=File exist

`CreateDirectory(`*`sDirectory`*`)`
This functions creates a directory (available from version 8.4.0). The parent directory has to exist already otherwise an error is issued. 0 is returned if the directory could be created, 1 if the directory already exist, 2 if the directory could not be created.

`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.

`ComDigIoOpen(iComPort,iDTR,iRTS)`
This function opens a serial port and initialized the output parameters DTR and RTS to the values iDTR and iRTS. It returns the handle iPortID (new from version 9.1 pf4).

`ComDigIoClose(iPortID)`
This function closes the communication port. It returns a success flag (new from version 9.1 pf4).

`ComDigIoSetDTR(iPortID,iDTR)`
This function sets the output value DTR to iDTR. It returns a success flag (new from version 9.1 pf4).

`ComDigIoSetRTS(iPortID,iRTS)`
This function sets the output value RTS to iRTS. It returns a success flag (new from version 9.1 pf4).

`ComDigIoGetDSR(iPortID)`

This function gets the input value of DSR (new from version 9.1 pf4).

`ComDigIoGetCTS(iPortID)`
This function gets the input value of CTS (new from version 9.1 pf4).

`ComDigIoGetDCD(iPortID)`
This function gets the input value of DCD (new from version 9.1 pf4).

`ComDigIoGetRI(iPortRI)`
This function gets the input value of RI (new from version 9.1 pf4).

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

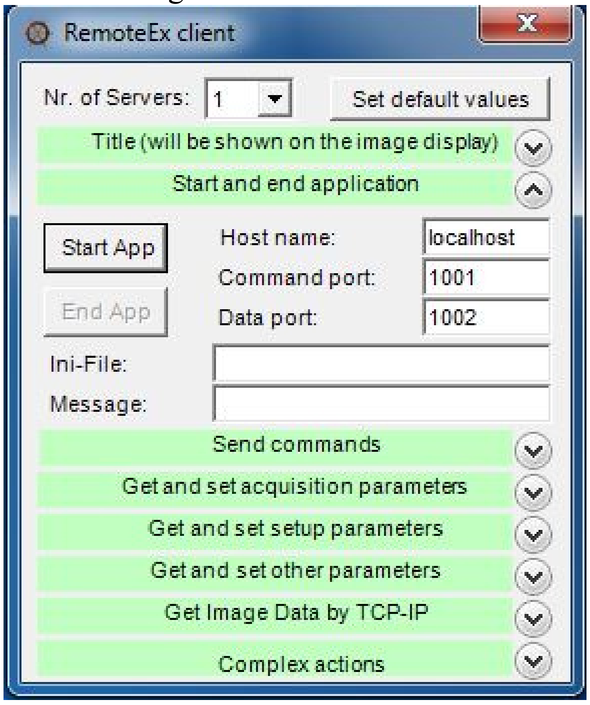
RemoteExClient sample

General

The RemoteEx is a programming interface to control the HiPic or HPD-TA from other applications, which allows a customer to use the HiPic or HPD-TA functionality from his application.
To show what can be done with this interface a RemoteEx Client program is provided. Taking into account the complete set of RemoteEx functions it can be a good guide to what possibly could be done with the RemoteEx interface.
Within the scope of its functions this client program can also be used for productive purposes and it may also be extended according to customer's requests. The RemoteEx client program covers many functions and allows access to many parameter, however it is mainly targeted to data acquisition. A unique feature is that this RemoteEx client can access more than one sever with one program (currently up to 9 servers can be accessed).

Startup and first use

As every other RemoteEx client it requires the RemoteEx application to run. It is recommended that the main application is not running when starting the RemoteEx Client program.
When starting the RemoteEx client the first time it will look like this:



RemoteEx client

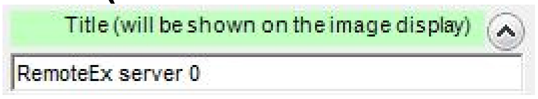
Nr. of Servers: 1

Host name: localhost
Command port: 1001
Data port: 1002
Ini-File:
Message:

Set default values
Title (will be shown on the image display)
Start and end application
Start App
End App
Send commands
Get and set acquisition parameters
Get and set setup parameters
Get and set other parameters
Get Image Data by TCP-IP
Complex actions

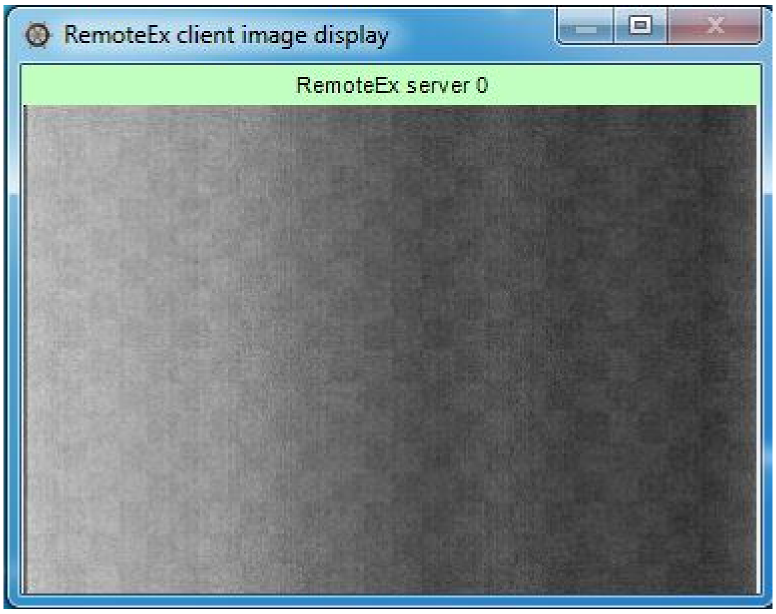
To start the main application it is enough to click to the "Start App" pushbutton.
After this the RemoteEx Client has connected to the RemoteEx and has started the main application.
To access different function the RemoteEx client has several section which are headed by a headline with light green background color.
To access the controls within this section the user can click on the arrow on the right side or just click on the headline. When doing this all controls in the section become visible and can be operated. If not needed another click to the arrow or the headline closes the section and only the headline is visible. This status and all other parameter are kept permanent when closing and opening the RemoteEx client again. The following chapters show what can be done in the individual sections.

Title (will be shown in the image display)



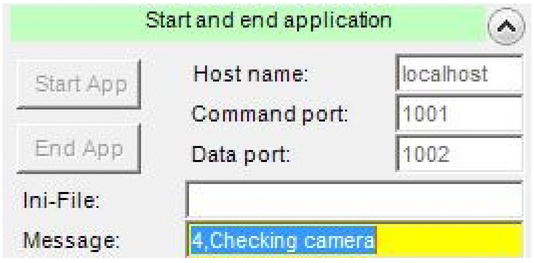
RemoteEx server 0

The string typed in here will be shown as a title line in the image display. This is especially useful if several servers are connected.



Start and end application

The controls in this sections lead to connect to the RemoteEx server.

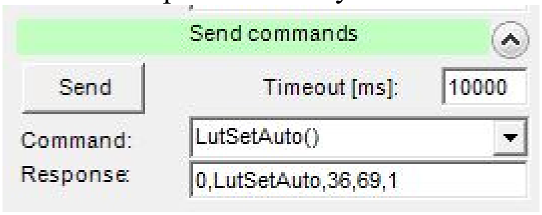


Host name: localhost
Command port: 1001
Data port: 1002
Ini-File:
Message: 4,Checking camera

Before starting the application the Command port and Data port and the Host name has to be specified. If the RemoteEx is running on the same computer “localhost” can be specified. Make sure that the values for Command port and Data port matches the values specified on the RemoteEx dialog.
Clicking to “Start App” starts the application, clicking to “End App” ends it. Messages output during the start process are outputted at the edit box labelled with “Message:”
When the edit box labeled “Ini-File:” is empty the default INI file is used. If the user wants to use another ini file he can specify another file here. Please note that this is a file on the Remote Computer. Please make sure to specify a valid file name. Specifying a different file name can be an advantage to make sure that all actions and parameter settings do not influence the standalone operation of the application.

Send commands

As a first step the user may want to send commands to the RemoteEx.

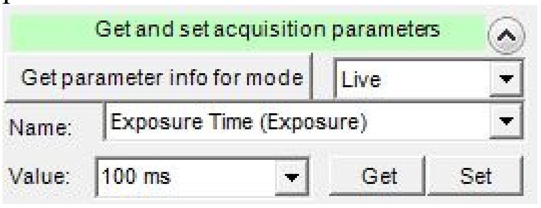


Timeout [ms]: 10000
Command: LutSetAuto()
Response: 0,LutSetAuto,36,69,1

This can be useful to check the effect of commands a user may want to use in his own RemoteEx client or if a user may want to execute a specific command even during productive use of the RemoteEx client. The command to execute should be entered in the edit box labelled “Command:”. It is executed when pressing the pushbutton “Send”. The response will be shown in the Edit box labelled “Response:”. The RemoteEx client waits the time specified in the edit box labelled “Timeout [ms]” before it times out if no response is coming.

Get and set acquisition parameters

The section “Get and set acquisition parameters” can be used to inquire and set acquisition parameters.


Get and set acquisition parameters
Get parameter info for mode
Get
Set

Live

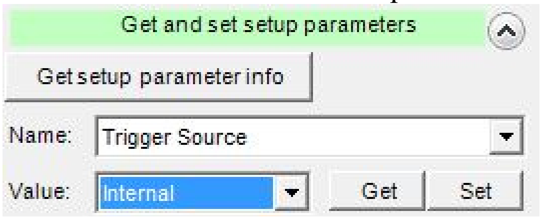
Name: Exposure Time (Exposure)

Value: 100 ms

To get a list of all available acquisition parameters from the specified acquisition mode select the proper acquisition mode and click to “Get parameter info for mode”. Then select a parameter from the list. After this the current value will appear in the field Value. If you want to set a new value enter the value into the field value and click “Set”. To confirm the current value you can click to the Get button and the RemoteEx reads the current value. The name field contains the current label on the control and the abstract name, under which the parameter is controlled.
The list of parameters contains all parameters which are currently visible in the application. Sometimes the visibility of individual parameters change. In such case it is necessary to click to “Get parameter info for mode” again to control parameters previously invisible. When changing the Acquisition mode the parameter list is updated automatically.

Get and set setup parameters

This section can be used to inquire and set the parameters from the camera options dialog.



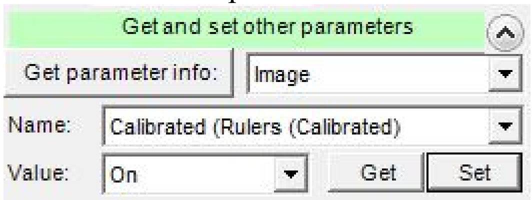
Name: Trigger Source

Value: Internal

To get a list of all parameters from the camera options dialog click to “Get setup parameter info”. With the Get and Set pushbuttons the user can get the current value of set a new value of the parameter.
The list of parameters contains all parameters which are currently visible in the application. Sometimes the visibility of individual parameters change. In such case it is necessary to click to “Get setup parameter info” again to control parameters previously invisible.

Get and set other parameters

With this section parameters from various origins can be controlled.



Image

Name: Calibrated (Rulers (Calibrated)

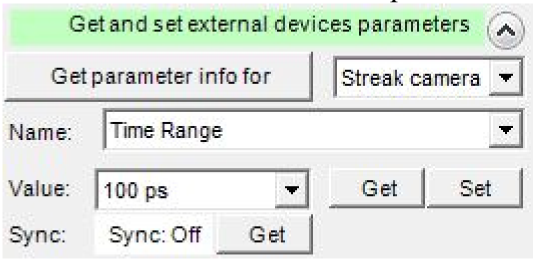
Value: On

To get the list of parameters of the specified origin select the origin and then click to “Get parameter info:”. As the next step select the parameter and get or set the current value.
Within this section parameters of the following origins can be selected:

Acquisition	Parameters from the acquisition options
Auxiliary devices	Parameters from the auxiliary devices options and from the MFX control and II control and II trigger dialogs
Background correction	Parameters from the background options
Curvature correction	Parameters from the curvature correction options
Defect pixel correction	Parameters from the defect pixel options
Shading correction	Parameters from the shading correction options
Defect pixel	Parameters from defect pixel detection dialog
General	Parameters from the general options
Image	Parameters from the image options
Look up table	Parameters from the LUT options
Main	Parameters from the main dialog
Arithmetic processing	Parameters from the arithmetic processing dialog
User function	Parameters from the user function dialog
Quick profile	Parameters from the quick profile options
Sequence	Parameters from the sequence dialog and the sequence options

Get and set external devices parameters

This section can be used to inquire and set parameters from the a specified external device.

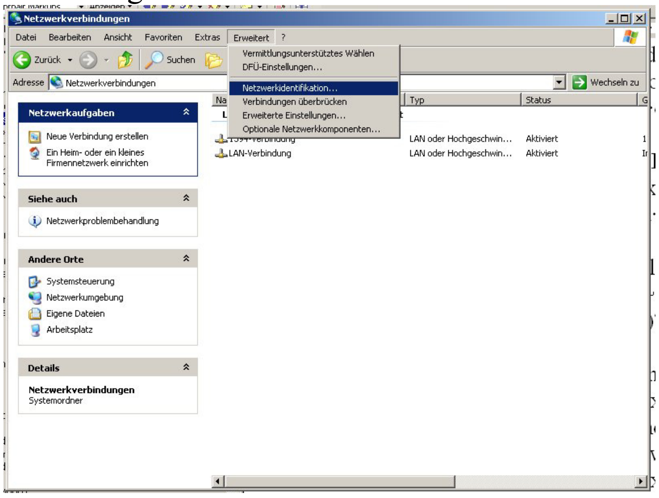


Streak camera
Name: Time Range
Value: 100 ps
Sync: Sync: Off

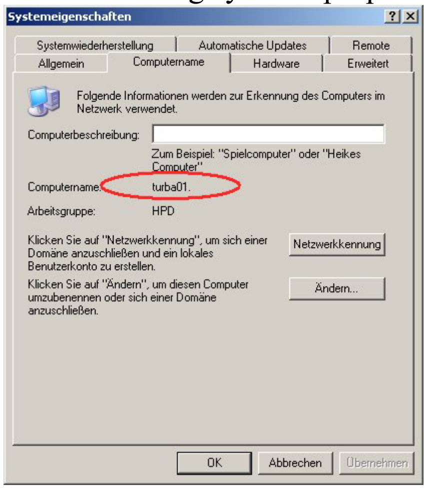
To get a list of all available parameters from a specified external device select the proper device aand click to “Get parameter info for”. Then select a parameter from the list. After this the current value will appear in the field Value. If you want to set a new value enter the value into the field value and click “Set”. To confirm the current value you can click to the Get button and the RemoteEx reads the current value.
When changing the external device the parameter list is updated automatically.

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.


Netzwerkverbindungen
Datei Bearbeiten Ansicht Favoriten Extras Erweitert ?
Zurück Suchen
Adresse Netzwerkverbindungen
Wechseln zu
Vermittlungsunterstütztes Wählen
DFÜ-Einstellungen...
Netzwerkidentifikation...
Verbindungen überbrücken
Erweiterte Einstellungen...
Optionale Netzwerkkomponenten...
Netzwerkaufgaben
Neue Verbindung erstellen
Ein Heim- oder ein kleines Firmennetzwerk einrichten
Siehe auch
Netzwerkproblembehandlung
Andere Orte
Systemsteuerung
Netzwerkumgebung
Eigene Dateien
Arbeitsplatz
Details
Netzwerkverbindungen
Systemordner
Typ
Status
LAN-Verbindung
LAN oder Hochgeschwin...
Aktiviert

Then the dialog system "properties appears" and you can see the computer's name.



Folgende Informationen werden zur Erkennung des Computers im Netzwerk verwendet.

Computerbeschreibung:

Zum Beispiel: "Spielcomputer" oder "Heikes Computer"

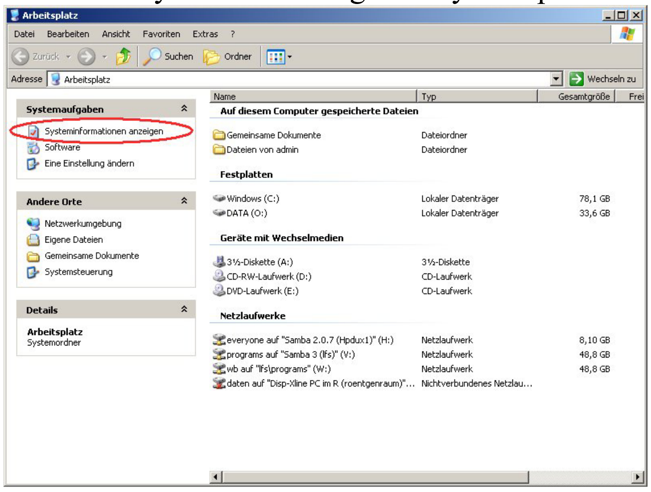
Computername: turba01.

Arbeitsgruppe: HPD

Klicken Sie auf "Netzwerkkennung", um sich einer Domäne anzuschließen und ein lokales Benutzerkonto zu erstellen.

Klicken Sie auf "Ändern", um diesen Computer umzubenennen oder sich einer Domäne anzuschließen.

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



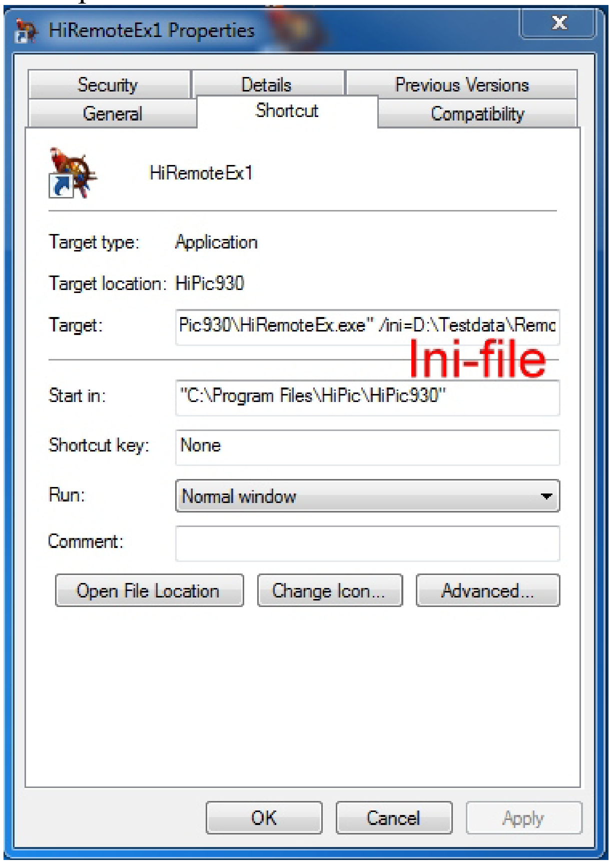
Name	Typ	Gesamtgröße
Auf diesem Computer gespeicherte Dateien		
Gemeinsame Dokumente	Dateiordner	
Dateien von admin	Dateiordner	
Festplatten		
Windows (C:)	Lokaler Datenträger	78,1 GB
DATA (O:)	Lokaler Datenträger	33,6 GB
Geräte mit Wechselmedien		
3½-Diskette (A:)	3½-Diskette	
CD-RW-Laufwerk (D:)	CD-Laufwerk	
DVD-Laufwerk (E:)	CD-Laufwerk	
Netzlaufwerke		
everyone auf "Samba 2.0.7 (Hpdux1)" (H:)	Netzlaufwerk	8,10 GB
programs auf "Samba 3 (lfs)" (V:)	Netzlaufwerk	48,8 GB
wb auf "lfs\programs" (W:)	Netzlaufwerk	48,8 GB
daten auf "Disp-Xline PC im R (roentgenraum)"...	Nichtverbundenes Netzlau...	

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

Specifying a different ini-file name

The default storage of the permanent settings for the RemoteEx program are in the ini-file RemoteEx.ini with in the directory ProgDataDir (See the command AppInfo(ProgDataDir) for details).
A command parameter can be used to specify a different ini file name (available since 9.3 pf6). This

is useful if several instances of RemoteEx should be used with different parameters. If a protocol file is written this protocol will be written in the same directory as the ini file.
To specify a different ini-file name use the token /ini=filename in the command parameter of a link.
Example:


HiRemoteEx1 Properties

HiRemoteEx1

Target type: Application

Target location: HiPic930

Target: Pic930\HiRemoteEx.exe" /ini=D:\Testdata\Remo

Start in: "C:\Program Files\HiPic\HiPic930"

Shortcut key: None

Run: Normal window

Comment: