
# Analyser TCL Library

## User's Guide



Byte Paradigm – info@byteparadigm.com

# Table of Content

1 Introduction ..... 4
2 TCL Interpreter ..... 5
2.1 Starting a TCL Session ..... 5
2.2 Getting Information on TCL Procedures ..... 7
3 Analyser Library - AnaTclLib ..... 8
3.1 Quick Reference Table ..... 8
3.2 Procedures Detailed Description ..... 10
3.3 Reference Sequences and Scripts ..... 22

# Table of Tables

Table 1: Quick reference table of Analyser procedures (by functionality) ..... 8

# Table of Pictures

Figure 1: Tcl session start-up example ..... 5
Figure 2: TCL console shortcut in 8PI Control Panel program group ..... 6
Figure 3: Tcl console at startup ..... 6
Figure 4: Program group with TCL script examples ..... 22

# References

[]

# History

| Version | Date | Description |
|---|---|---|
| 1.00 | 01-Nov-2005 | Initial revision |
| 1.01 | 15-Feb-2006 | Updated to comply with software version 1.03 |
| 1.02 | 22-May-06 | Some corrections |
| 1.03 | 09-Nov-2006 | Update for software version 1.04 |
| 1.04 | 29-Aug-2007 | Added trigger positioning functions / description |
| 1.05 | 13-Nov-2007 | Update for GP Series Introduction |
| 1.06 | 16-Feb-2010 | Review for release 1.08f. |
| 1.07 | 20-Jan-2012 | Added IO voltage selection |

# 1 Introduction

The objective of this document describe all the TCL procedures available in the Analyser library provided to control the Analyser operating mode of the GP Series device. Each procedure functionality and parameters are described in detail. Scripts examples, that use some of these procedures, are also provided to help the users build their own test environments.

A section is also dedicated to the TCL interpreter provided with the *8PI Control Panel* application. The way to start it and to use it is briefly described.

# 2 TCL Interpreter

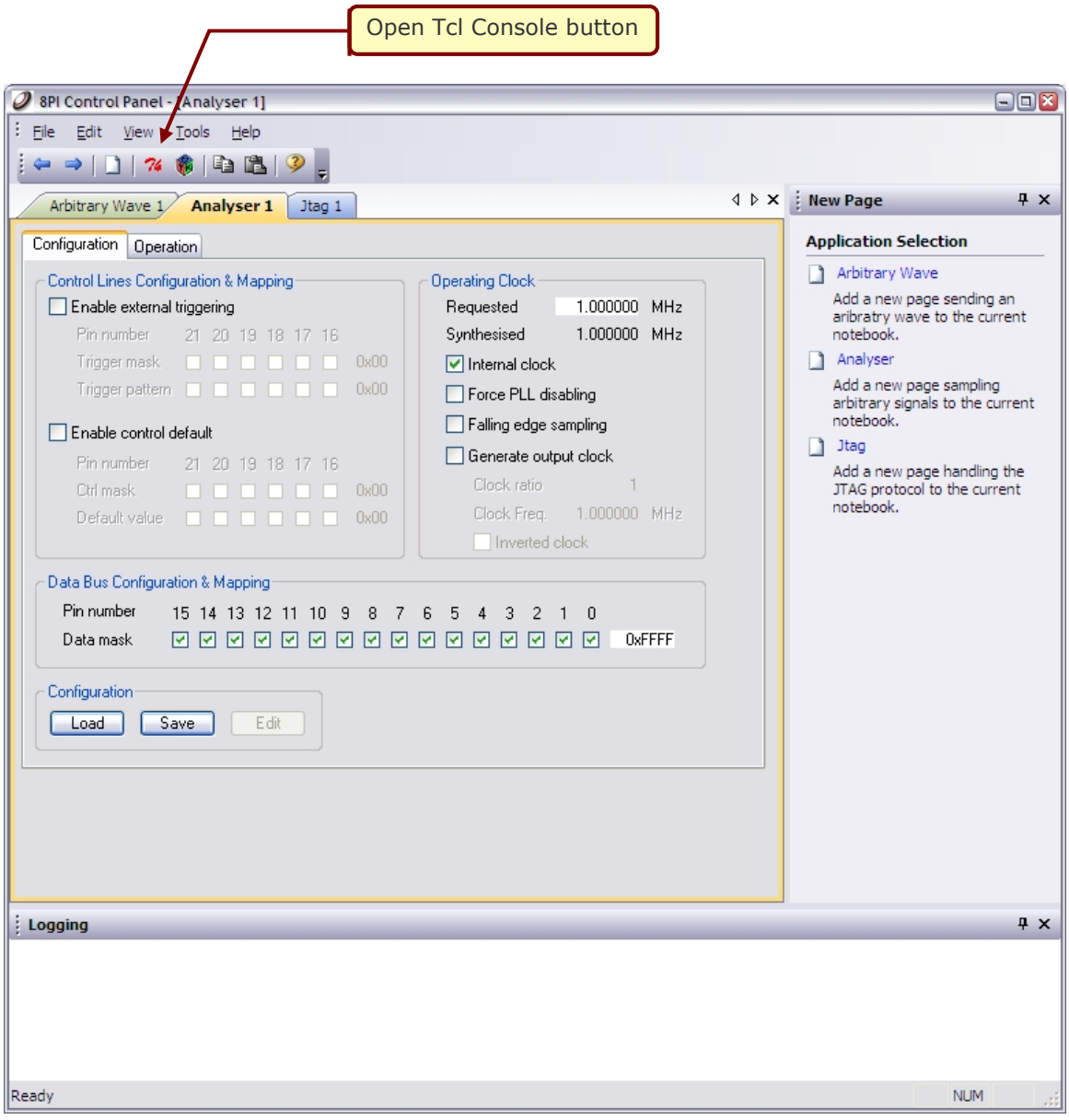
## 2.1 Starting a TCL Session

**To start a TCL session from the 8PI Control Panel GUI:**

1. From the 8PI Control Panel GUI, access the desired operating mode sheet.
2. Click on the 'Open Tcl Console' button (Figure 1).

This opens the Tcl console, loads the Tcl libraries relative to the chosen operating mode and initialises the Tcl session.

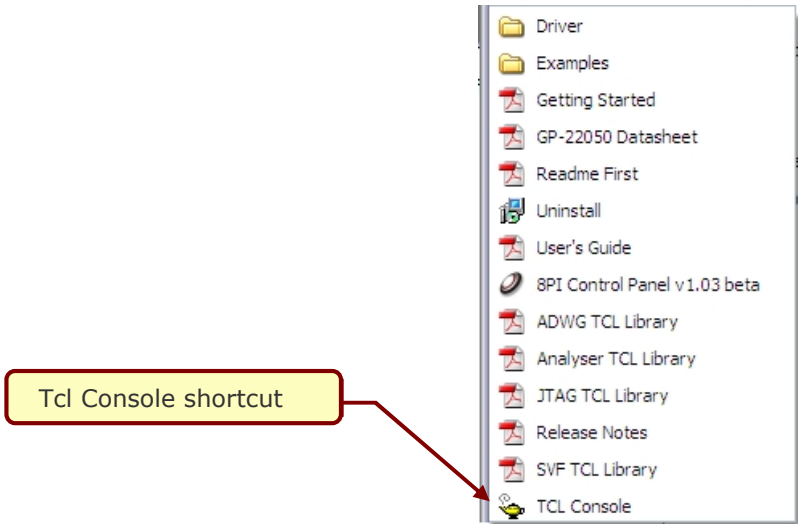
**Figure 1: Tcl session start-up example**



**To start a stand-alone TCL session (without running GUI):**

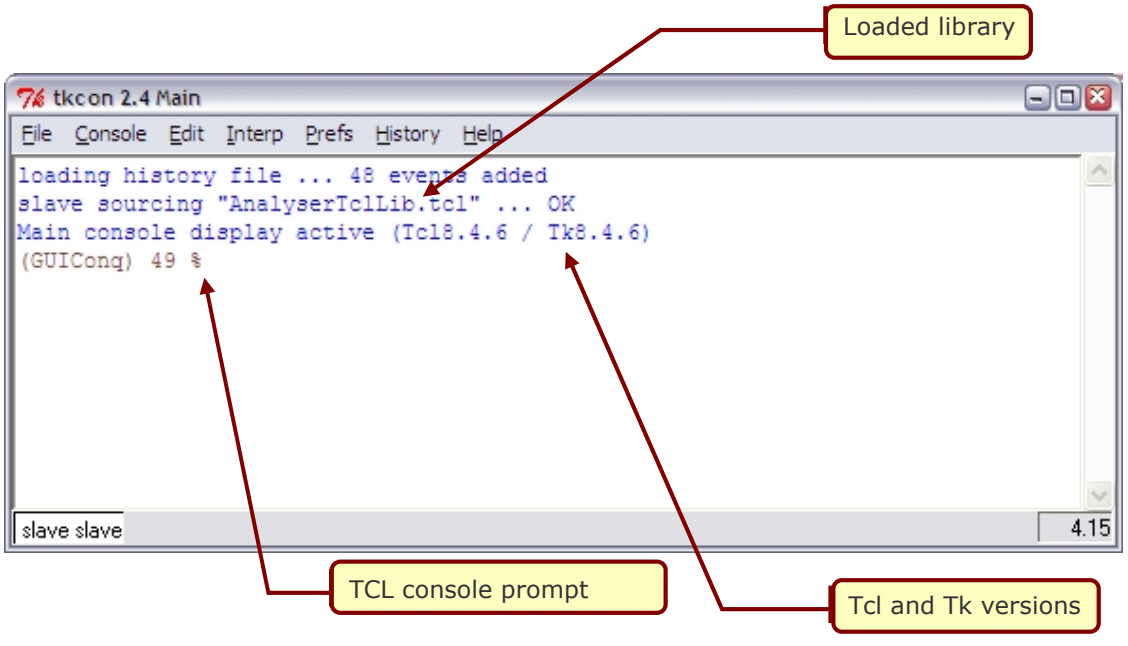
1. From the 'Byte Paradigm > 8PI Control Panel' program group, click on the 'TCL Console' shortcut. This starts the Wish84 interpreter with the tkcon console.
2. In the Tcl console, type: `% source AnalyserTclLib.tcl`
   This initialises your TCL session in Analyser operating mode.

**Figure 2: TCL console shortcut in 8PI Control Panel program group**




By default, the 8PI Control Panel software environment uses the WISH interpreter with the TKCON console (interactive mode) (Figure 3). For more information about the TKCON console, please check the following links: http://tkcon.sourceforge.net/ - http://wiki.tcl.tk/1878.
Please note that TCL is case sensitive.

**Figure 3: Tcl console at startup**




## 2.2 Getting Information on TCL Procedures

Tcl provides built-in commands for getting information about the elements loaded in memory during a Tcl session. We simply describe a few of them for those unfamiliar to the Tcl language.

***To list the libraries loaded in the TCL environment:***

```
% info loaded
```

***To list all the procedures loaded in the TCL environment:***

```
% info procs
```

***To list the arguments of a given procedure:***

```
% info args <procedure name>
```

***To list the body of a given procedure:***

```
% info body <procedure name>
```

To learn more about the TCL/TK language, numerous man pages, tutorials and references can be found at the following location: http://www.tcl.tk/doc/ .

# 3 Analyser Library - AnaTclLib

This library contains all the procedures available to control the GP Series device when operating in analyser mode. Using these procedures in scripts or command lines allows the user to capture samples from the system connector of the GP Series device.

## 3.1 Quick Reference Table

Table 1 gives a list of all the procedures available for the Analyser mode. They are grouped by type and functionality.

**Table 1: Quick reference table of Analyser procedures (by functionality)**

| Procedures | Description |
|---|---|
| **Clock Configuration** | |
| `SetReqClock {Freq}` | Defines the operating clock frequency (in Hz). |
| `GetReqClock {}` | Returns the requested operating clock frequency (in Hz). |
| `GetSynthClock {}` | Returns the synthesised operating clock frequency (in Hz). It can differ from the requested frequency due to the limited accuracy of the frequency divider. |
| `GetSynthOutClock {}` | Returns the achieved frequency (in Hz) of the output clock. This value depends on the operating clock frequency and the output clock ratio. |
| `SetOutputClock {Output {Display 1}}` | Enables or disables the generation of an output clock |
| `GetOutputClock {}` | Gets the status of the output clock generation (enabled or disabled) |
| `UseIntClock {}` | Configures the device to operate with the internal clock as reference. |
| `UseExtClock {}` | Configures the device to operate with an externally supplied clock as reference. |
| `GetClockSource {{Display 0}}` | Returns the selected clock source configured in the device. |
| `SetClockEdge {Pos {Display 1}}` | Defines the phase relation between the output clock and the device internal clock. |
| `GetClockEdge {}` | Returns the phase relation between the device internal clock and the generated output clock. |
| `SetClockSampling {RisingEdge {Display 0}}` | Defines the sampling clock edge used to sample incoming data |
| `GetClockSampling {{Display 0}}` | Returns the clock edge currently used to sample data. |
| `SetOutClockRatio {Ratio}` | Defines the integer output clock ratio with respect to the operating clock. |
| `GetOutClockRatio {}` | Returns the current value of the output clock ratio |
| `SetClockDisablePLL {Disable {Display 1}}` | Forces disabling the PLL or let the device enable it automatically when possible. |
| `GetClockDisablePLL {{Display 0}}` | Gets the operating mode of the PLL (disabled/automatic) |
| **Triggering Configuration** | |
| `SetInternalTrigger {Internal {Display 1}}` | Defines the triggering mode (internal or external). |
| `GetInternalTrigger {{Display 0}}` | Returns the current triggering mode. |
| `SetEdgeTrigger {Enable {Display 1}}` | Selects the triggering mode (edge or level) |
| `GetEdgeTrigger {{Display 0}}` | Returns the triggering mode (edge or level) |
| `SetTriggerPos {Sample}` | Defines the position of the trigger in the run – that is the number of samples before the trigger. |

| Procedures | Description |
|---|---|
| GetTriggerPos {} | Returns the programmed trigger position. |
| SetCtrlTrigMask {Mask {Display 1}} | Defines the triggering mask to select the control lines to use as trigger inputs. |
| GetCtrlTrigMask {} | Returns the triggering mask. |
| SetCtrlTrigPattern {Pattern {Display 1}} | Defines the pattern to detect on the trigger inputs to generate the trigger event. |
| GetCtrlTrigPattern {} | Returns the triggering pattern. |
| **Control Lines Configuration** | |
| EnDefaultCtrl {{Display 0}} | Enables applying default static levels on the control lines. |
| DisDefaultCtrl {} | Disables applying a forced default level on the control lines. |
| GetDefaultCtrlEn {{Display 0}} | Returns the status of the default level feature. |
| SetCtrlMaskIn {MaskIn {Display 1}} | The control mask selects the control lines on which the default static levels have to be applied. |
| GetCtrlMaskIn {} | Returns the control mask value. |
| SetCtrlDefaultVal {Val {Display 1}} | Defines the default level to apply on the control lines. |
| GetCtrlDefaultVal {} | Returns the current default level applied on the control lines. |
| **Data Lines Configuration** | |
| SetDataMaskIn {MaskIn {Display 1}} | Selects the data lines enabled as inputs. |
| GetDataMaskIn {{Display 1}} | Returns the data mask value. |
| **Operations** | |
| Unsupervised {Enable {Display 1}} | Enables or disables the unsupervised operating mode. |
| AnalyserRun {NrSamples {Display 1} {PopUp 0}} | Starts collecting data with the GP Series device and store them in the host memory. |
| GetCapturedData {StartIndex NrData} | Display data collected with the Analyser and stored in the host memory. |
| **File Management** | |
| SetExportFileType {ExportFileType {Display 0}} | Defines the 'AutoSave' option file type. |
| GetExportFileType {} | Returns the defined 'AutoSave' option file type. |
| SetExportAutoSave {AutoSave {Display 0}} | Enables / disables the 'AutoSave' option. |
| GetExportAutoSave {} | Returns the 'AutoSave' option status (enabled/disabled). |
| SetExportFileName {FileName} | Defines the 'AutoSave' option file name. |
| GetExportFileName {} | Returns the 'AutoSave' option file name. |
| ExportRawBinFile {FileName} | Exports the data collected with the Analyser to a raw binary file. |
| ExportRawTextFile {FileName} | Exports the data collected with the Analyser to a raw text file. |
| ExportVCDFile {FileName} | Exports the data collected with the Analyser to a VCD file. |
| MakeConfFileTemplate {FileName} | Creates a configuration file template. |
| ReadConfFile {FileName {Display 0} {PopUp 0}} | Reads the device controls, configuration and data header from a file. |
| WriteConfFile {FileName} | Writes the current configuration to a file. |
| **Miscellaneous procedures** | |
| SelectIOVoltage {IOVoltage} | Selects the user interface voltage. |
| GetLastError {} | Returns the last error detected during the data transfer. |
| ResetCfg {} | Reset the device to a default configuration. |
| InitVarsAna {} | Initialises local variables to a default value. |
| AnalyserConfig {} | Display the Analyser session configuration. |

| Procedures | Description |
|---|---|
| `Ver {}` | Displays the software version. |
| `IsDeviceReady {}` | Checks if the device is properly connected to the host PC. |
| `Terminate {}` | Closes the Analyser TCL session. |

## 3.2 Procedures Detailed Description

This section gives a detailed description of each procedure available to control the GP Series device analyser mode. The procedures are listed in alphabetic order.

**AnalyserConfig {}**

| | |
|---|---|
| ***parameters :*** | ***none*** |
| *returns:* | |
| *description:* | Displays the Analyser session configuration. |
| *conditions:* | None |
| *see also:* | |

**AnalyserRun {NrSamples {Display 1} {PopUp 0}}**

| | | |
|---|---|---|
| ***parameters :*** | ***NrSamples:*** | ***integer value that specifies the number of samples to be collected.*** |
| | ***Display:*** | ***Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| | ***PopUp:*** | ***Optional parameter. Enables pop up windows when set to 1. By default set to 0.*** |
| *returns:* | 0 | if sampling was successful |
| | -1 | if sampling failed. |
| *description:* | Directs the sampling of data from the GP Series device system connector. The specified number of samples are collected from the GP Series device system connector and transferred to the host PC memory. If the *AutoSave* option is selected, this procedure also automatically saves the collected samples, with the chosen predefined options. | |
| *conditions:* | None | |
| *see also:* | *SetDataMaskIn{}, ExportRawTextFile{}, ExportRawBinFile{}, ExportVCDFile{}, SetExportAutoSave{}, SetExportFileType{}, SetExportFileName{}* | |

**DisDefaultCtrl {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | |
| *description:* | Disables applying a forced default level on the control lines |
| *conditions:* | |
| *see also:* | *EnDefaultCtrl{}, EnCtrlseq{}, DisCtrlseq{}* |

**EnDefaultCtrl {{Display 0}}**

| | | |
|---|---|---|
| ***parameters :*** | ***Display:*** | ***Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 0.*** |
| *returns:* | | |
| *description:* | Enables applying default static levels on the control lines. When this | |

feature is enabled, static levels can be set on the selected control lines. Valid for *File* or *Static* modes.

*conditions:* To enable the default level feature for the control lines, the control sequence feature has to be enabled. Using *EnDefaultCtrl{}* automatically calls *EnCtrlSeq{}* if needed. If *Display* is set to 1, a message is displayed to report the control sequence enabling.

*see also:* *DisDefaultCtrl{}, EnCtrlseq{}, DisCtrlseq{}, SetDefaultVal{}*

**ExportRawBinFile {FileName}**

***parameters:*** ***FileName:*** ***specifies the procedure output file name (including path).***

*returns:* 0 when the file is saved.

*description:* Saves the last sampling run samples to a file, formatted as raw binary data.

*conditions:*

*see also:* *AnalyserRun{}*

**ExportRawTextFile {FileName}**

***parameters:*** ***FileName:*** ***specifies the procedure output file name (including path).***

*returns:* 0 when the file is saved.

*description:* Saves the last sampling run samples to a file, formatted as raw text data.

*conditions:*

*see also:* *AnalyserRun{}*

**ExportVCDFile {FileName}**

***parameters:*** ***FileName:*** ***specifies the procedure output file name (including path).***

*returns:* 0 when the file is saved.

*description:* Saves the last sampling run samples to a file, formatted as value change dump (VCD) vector information.

*conditions:*

*see also:* *AnalyserRun{}*

**GetCapturedData {StartIndex NrData}**

***parameters:*** ***StartIndex:*** ***integer value specifying the rank of the first sampled data to be returned to the TCL console.***

***NrData:*** ***integer value specifying the number of samples to be returned to the TCL console.***

*returns:* A TCL list object holding the requested samples, displayed as hexadecimal numbers (4 digits – 16 bits).

*description:* Displays the values of the last set of samples collected by using *AnalyserRun{}* procedure. The data are returned as a set of space separated values (TCL list).

*conditions:*

*see also:* *AnalyserRun{}*

**GetClockDisablePLL {{Display 0}}**

***parameters:*** ***Display:*** ***Optional parameter. Displays additional information messages when set to 1. By default set to 0.***

*returns:* 1 when the internal device PLL is disabled
0 when the automatic PLL control mode is enabled

*description:* Returns the control mode of the device internal PLL.

*conditions:*

*see also:* *SetClockDisablePLL{}*

**GetClockEdge {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | 1 Output data lines are generated in phase with the rising edge of the output clock.<br>0 Output data lines are generated in phase with the falling edge of the output clock |
| *description:* | Returns the phase setting between the data transition and the output clock. |
| *conditions:* | |
| *see also:* | *SetClockEdge{}* |

**GetClockSampling {{Display 0}}**

| | |
|---|---|
| ***parameters :*** | ***Display: Optional parameter. Displays additional information messages when set to 1. By default set to 0.*** |
| *returns:* | 1 if the clock rising edge is used to sample data.<br>0 if the clock falling edge is used to sample data. |
| *description:* | Returns the clock edge that is used to sample the incoming data. |
| *conditions:* | |
| *see also:* | *SetClockSampling{}* |

**GetClockSource {{Display 0}}**

| | |
|---|---|
| ***parameters :*** | ***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 0.*** |
| *returns:* | 1 Internal reference clock is used<br>0 External reference clock is used<br>When *Display* is set to 1, and additional text message is returned depending on the selected clock source.<br>*Source: INTERNAL clock*<br>*Source: EXTERNAL clock* |
| *description:* | Returns the reference clock source used to generate the device operating clock. |
| *conditions:* | |
| *see also:* | *UseIntClock{}, UseExtClock{}* |

**GetCtrlDefaultVal {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A hexadecimal value prefixed with 0x |
| *description:* | Returns the current default level pattern defined for the 6 control lines. The LSB of the value corresponds to control line 0 and the MSB corresponds to control line 5. |
| *conditions:* | |
| *see also:* | *SetCtrlDefaultVal{}, EnDefaultCtrl{}* |

**GetCtrlMaskIn {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A hexadecimal value prefixed with 0x |
| *description:* | Returns a mask value representing the control lines enabled as input. The |

LSB of the value corresponds to control line 0 and the MSB corresponds to control line 5. When a bit is set to 1, the corresponding control line is configured as an input. When a bit is set to 0, the corresponding pin is ignored.

*conditions:*
*see also:* *SetCtrlMaskIn{}, SetCtrlTrigMask{}*

**GetCtrlTrigMask {}**

***parameters:*** ***None***
*returns:* A hexadecimal value prefixed with 0x
*description:* Returns the mask applied on the control lines to detect the external trigger pattern.
*conditions:* *SetCtrlTrigMask{}, SetCtrltrigPattern{}, SetInternalTrigger{}*

**GetCtrlTrigPattern {}**

***parameters:*** ***None***
*returns:* A hexadecimal value prefixed with 0x
*description:* Returns the pattern to detect on the selected control lines to generate a trigger event to start applying data samples on the GP Series device system connector.
*conditions:*
*see also:* *SetCtrlTrigPattern{}, SetCtrlTrigMask{}, SetInternalTrigger{}*

**GetDataMaskIn {}**

***parameters:*** ***None.***
*returns:* A hexadecimal value prefixed with 0x
*description:* Returns a mask value representing the data lines enabled as input. The LSB of the value corresponds to data line 0 and the MSB corresponds to data line 15. When a bit is set to 1, the corresponding data line is configured as an output. When a bit is set to 0, the corresponding pin is ignored.
*conditions:*
*see also:* *SetDataMaskIn{}*

**GetDefaultCtrlEn {{Display 0}}**

***parameters:*** ***Display: Optional parameter. Displays additional information messages when set to 1. By default set to 0***
*returns:* 1 when the default level feature is enabled
0 when the default level feature is disabled
*description:* Returns the status of the default level feature for the control lines.
*conditions:*
*see also:* *EnDefaultCtrl{}, EnCtrlSeq{}*

**GetEdgeTrigger {{Display 0}}**

***parameters:*** ***Display: Optional parameter. Displays additional information messages when set to 1. By default set to 0.***
*returns:*
*description:* Returns the current triggering mode: 1 for 'edge triggering mode'; 0 for 'level triggering mode'.

*see also:*

**GetExportAutoSave {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | 1 when the *AutoSave* option is enabled<br>0 when the *AutoSave* option is disabled |
| *description:* | The *AutoSave* option allows to automatically save the data collected with *AnalyserRun{}* command to a file specified with the *SetExportFileName{}* and *SetExportFileType{}* procedures. *GetExportAutoSave{}* returns the enable/disable status of this option |
| *conditions:* | |
| *see also:* | *SetExportAutoSave{}, AnalyserRun{}, SetExportFileName{}, SetExportFileType{}* |

**GetExportFileName {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A string with name of the export file name used with the *AutoSave* option |
| *description:* | The *AutoSave* option allows to automatically save the data collected with *AnalyserRun{}* command to a file specified with the *SetExportFileName{}* and *SetExportFileType{}* procedures. *GetExportFileName{}* returns the file name set with the command *SetExportFileName{}.* |
| *conditions:* | |
| *see also:* | *SetExportFileName{}, SetExportAutoSave{}, AnalyserRun{}, SetExportFileType{}* |

**GetExportFileType {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A string with type of the export file used with the *AutoSave* option. Possible values are *"TEXT", "BIN" and "VCD"* |
| *description:* | The *AutoSave* option allows to automatically save the data collected with *AnalyserRun{}* command to a file specified with the *SetExportFileName{}* and *SetExportFileType{}* procedures. *GetExportFileType{}* returns the file type set with the command *SetExportFileType{}.* |
| *conditions:* | |
| *see also:* | *SetExportFileType{}, SetExportAutoSave{}, AnalyserRun{}, SetExportFileName{}* |

**GetInternalTrigger {{Display 0}}**

| | |
|---|---|
| ***parameters :*** | ***Display: Optional parameter. Displays additional information messages when set to 1. By default set to 0*** |
| *returns:* | 1 when internal triggering mode is enabled<br>0 when external triggering mode is enabled |
| *description:* | Returns the triggering mode in use. |
| *conditions:* | |
| *see also:* | *SetInternalTrigger{}, SetExternalTrigger{}* |

**GetLastError {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |

| | |
|---|---|
| *returns:* | A decimal value |
| *description:* | Returns the last error code returned by the device.<br>1 : underflow error: the system available bandwidth is not sufficient. Please reduce the used clock frequency.<br>Any other value different from 0: system error. Please report error to Byte Paradigm. |
| *conditions:* | This procedure can be used in unsupervised mode, after having used AnalyserRun, to check if the last set of data was sent without error. |
| *see also:* | *Unsupervised{}, AnalyserRun{}* |

**GetOutClockRatio {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A decimal value |
| *description:* | Returns the integer ratio used to generate the output clock. |
| *conditions:* | |
| *see also:* | *SetOutClockRatio{}* |

**GetOutputClock {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | 1 when the output clock is enabled<br>0 when the output clock is disabled |
| *description:* | Returns status of the output clock. |
| *conditions:* | |
| *see also:* | *SetOutputClock{}, SetOutClockRatio{}* |

**GetReqClock {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A decimal value |
| *description:* | Returns the requested frequency for the device operating clock (unit = Hz) |
| *conditions:* | |
| *see also:* | *GetSynthClock{}, SetReqClock{}* |

**GetSynthClock {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A decimal value |
| *description:* | Returns the achieved frequency for the system operating clock. As the operating clock is generated using an integer ratio to divide a reference clock, not all frequencies can be generated. This procedure returns the achieved frequency value. This value can differ from the requested clock frequency. If the requested frequency can not be achieved, the device uses the first achievable frequency smaller than the requested one. |
| *conditions:* | |
| *see also:* | *SetReqClock{}, GetReqClock{}* |

**GetSynthOutClock {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | A decimal value |
| *description:* | Returns the achieved frequency of the output clock (unit = Hz). |

*see also:* *SetOutClockRatio{}*

**GetTriggerPos {}**

***parameters:*** ***none.***
*returns:* A integer representing the trigger position.
*description:* Returns the trigger position previously programmed. The trigger position is defined with the sample index at which it is positioned.

**InitVarsAna {}**

***parameters:*** ***None***
*returns:*
*description:* Initialises *Analyser mode* parameters
*conditions:*

**IsDeviceReady {}**

***parameters:*** ***None***
*returns:* 1 if device is connected.
*description:* Checks if the device is properly connected to the host PC.
*see also:*

**MakeConfFileTemplate {FileName}**

***parameters:*** ***FileName: specifies the procedure output file name (including path).***
*returns:* 0 when file generation is complete
*description:* Creates a standard template to be used as configuration file with the device.
*conditions:*
*see also:*

**ReadConfFile {FileName {Display 0} {PopUp 0}}**

***parameters:*** ***FileName: specifies the procedure input file name (including path)***
***Display: Optional parameter. Displays additional information messages when set to 1. By default set to 0***
***PopUp: Optional parameter. Enables pop up windows when set to 1. By default set to 0.***
*returns:* 0 if read succesful; another value if read failed.
*description:* Reads the device configuration, the control sequence configuration and the data header from the specified file. Does not expect any set of data defined in the input file.
*conditions:*
*see also:*

**ResetCfg {}**

***parameters:*** ***None***
*returns:*
*description:* Resets the device to a default state.
*conditions:*

**SetClockDisablePLL {Disable {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Disable: valid values: 0 or 1. Specifies whether the PLL must be disabled (1) or left enabled when needed (0).***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *Returns:* | |
| *Description:* | Forces the disabling of the device internal PLL used for the connector clock generation. For the generation of the connector clock, a PLL embedded in the device can be used. When the *Disable* parameter is set to 0, the PLL will be automatically enabled if possible. When *Disable* is set to 1, the PLL is not used. |
| *Conditions:* | |

**SetClockEdge {Pos {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Pos: valid values: 0 or 1. Specifies the output clock polarity. 1 for rising edge; 0 for inverted clock.***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | Defines the phase relation between the output clock and the device internal clock. When the clock edge is set to 1, the output clock is in phase with the device internal clock; when set to 0, there is a 180° phase shift between the clocks. |
| *conditions:* | |
| *see also:* | *GetClockEdge{}* |

**SetClockSampling {RisingEdge {Display 0}}**

| | |
|---|---|
| ***parameters :*** | ***RisingEdge: valid values: 0 or 1. Specifies the reference clock edge on which the input data is sampled. 1 for rising edge; 0 for falling edge.***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 0.*** |
| *returns:* | |
| *description:* | Defines the phase relation between the output clock and the data transition. |
| *conditions:* | |
| *see also:* | *GetClockSampling{}* |

**SetCtrlDefaultVal {Val {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Val: hexadecimal (decimal) input value ranging from 0x00 (0) to 0x3F (63).***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | The static pattern is applied on the control lines while sampling data. The level remains constant until a new control pattern is defined. If a new pattern is defined, it will only be applied on the control lines with the next sampled data. |
| *conditions:* | |

**SetCtrlMaskIn {MaskIn {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***MaskIn: hexadecimal (decimal) input value ranging from 0x00 (0) to 0x3F (63).***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | The control mask selects the device control lines on which the default static levels have to be applied. When a mask bit is set to 0, the corresponding control line is masked. The mask is given as a hexadecimal value; each of its bits corresponds to one of the six control lines, MSB to LSB. |
| *conditions:* | |

**SetCtrlTrigMask {Mask {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Mask: hexadecimal (decimal) input value ranging from 0x00 (0) to 0x3F (63).***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | When the external triggering is used, the trigger mask selects the control lines to be used as trigger inputs. When a mask bit is set to 0, the corresponding control line is masked for triggering. The mask is given as a hexadecimal value; each of its bits corresponds to one of the six control lines, MSB to LSB |
| *conditions:* | |

**SetCtrlTrigPattern {Pattern {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Pattern: hexadecimal (decimal) input value ranging from 0x00 (0) to 0x3F (63).***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | Defines the pattern to detect on the trigger inputs to generate the trigger event. |
| *conditions:* | |

**SetDataMaskIn {MaskIn {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***MaskIn: hexadecimal (decimal) input value ranging from 0x0000 (0) to 0xFFFF (65535).***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | Applies a mask onto the data input lines to enable / disable them. The mask MSB and LSB respectively correspond to data line 15 and data line 0. A mask bit set to 1 enables the data line; a mask bit set to 0 disables it. |
| *conditions:* | |

**SetEdgeTrigger {Enable {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Enable: Valid values: 0 or 1 – 0 for 'level trigger', 1 for 'edge trigger'***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By*** |

***default set to 0.***

*returns:*

*description:* Selects the triggering mode: in edge triggering mode (Enable = 1), the Analyser run will be triggered upon detection of a transition to the programmed trigger pattern. In level triggering mode (Enable = 0), the Analyser run is triggered upon detection of the programmed trigger pattern.

*see also:*

**SetExportAutoSave {AutoSave {Display 0}}**

***parameters:*** ***AutoSave: valid values: 0 or 1. Specifies if the AutoSave option must be enabled or disabled. 1 for enable; 0 for disable.***
***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 0.***

*returns:*
-1 when the operation did not succeed
0 when the operation succeeded

*description:* The *AutoSave* option allows to automatically save the data collected with *AnalyserRun{}* command to a file specified with the *SetExportFileName{}* and *SetExportFileType{}* procedures. *SetExportAutoSave{}* enables/disables it.

*conditions:*

*see also:* *GetExportAutoSave{}, AnalyserRun{}, SetExportFileName{}, SetExportFileType{}*

**SetExportFileName {FileName}**

***parameters:*** ***FileName: specifies the file name to be used with the AutoSave option.***

*returns:*

*description:* The *AutoSave* option allows to automatically save the data collected with *AnalyserRun{}* command to a file specified with the *SetExportFileName{}* and *SetExportFileType{}* procedures. *SetExportFileName{}* specifies the file name (including path).

*conditions:*

*see also:* *GetExportFileName{}, AnalyserRun{}, SetExportFileName{}, SetExportFileType{}*

**SetExportFileType {FileType {Display 0}}**

***parameters:*** ***FileType: specifies the file type to be used with the AutoSave option. Valid values are TEXT, BIN and VCD***
***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 0.***

*returns:*

*description:* The *AutoSave* option allows to automatically save the data collected with *AnalyserRun{}* command to a file specified with the *SetExportFileName{}* and *SetExportFileType{}* procedures. *SetExportFileType{}* specifies the file type where the autosaved data have to be stored. Only *TEXT (raw text file), BIN (raw bin file)* and *VCD (VCD file)* can be specified.

*conditions:*

*see also:* *GetExportFileName{}, AnalyserRun{}, SetExportFileName{}, SetExportFileType{}, ExportRawTextFile{}, ExportRawBinFile{}, ExportVCDFile{}*

**SetInternalTrigger {Internal {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Internal: valid values: 0 or 1.***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | Defines the triggering mode. When *Internal* is set to 1, the internal triggering mode is selected; when *Internal* is set to 0, the external triggering mode is selected. |
| *conditions:* | |

**SelectIOVoltage {IOVoltage}**

| | |
|---|---|
| ***parameters :*** | ***IOVoltage: integer value representing the IO voltage, the IO voltage can be internally generated of user applied. The voltage level is defined in millivolts.*** |
| *returns:* | |
| *description:* | This function only takes the following predefined values: 3300, 2500, 1800, 1500 and 1200. The nearest value must be selected when the user applies a different external voltage level. For example, set IOVoltage to 2500 when 2.7V is applied.<br>The default value is 3300. |
| *conditions:* | |

**SetOutClockRatio {Ratio}**

| | |
|---|---|
| ***parameters :*** | ***Ratio: decimal value ranging from 1 to 65535.*** |
| *returns:* | |
| *description:* | Defines the output clock ratio with respects to the operating clock: Frequency(Output Clock) = Frequency(Operating Clock) / *Ratio*. |
| *conditions:* | |

**SetOutputClock {Output {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Output: valid values: 0 or 1.***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | Enables or disables the generation of an output clock. *Output* set to 1 enables the generation of the output clock; *Output* set to 0 disables it. |
| *conditions:* | |
| *see also:* | SetOutClockRatio{}, SetHoleClock{}, SetClockEdge{} |

**SetReqClock {Freq}**

| | |
|---|---|
| ***parameters :*** | ***Freq: integer value representing the requested frequency in Hz. Range: from 800 (800 Hz) to 50000000 (50 MHz) for GP-22050/Xpress and 100000000 (100MHz) for the GP-24100.*** |
| *returns:* | |
| *description:* | Defines the requested operating clock frequency. The device will actually set the operating clock frequency to the closest frequency available. |
| *conditions:* | |
| *see also:* | GetReqClock{}, GetSynthClock{} |

**SetTriggerPos {Sample}**

| | |
|---|---|
| ***parameters :*** | ***Sample: integer value representing the index of the sample in the run where the trigger should be positioned.*** |
| *returns:* | An integer error code: 0 is successful; another value if unsuccessful. |
| *description:* | Use this function to position the trigger after a given number of samples in the total run. Once the sampling run is over, the corresponding number of samples before the trigger is displayed, together with the rest of the run *after* the trigger. |

**Terminate {}**

| | |
|---|---|
| ***parameters :*** | ***None*** |
| *returns:* | |
| *description:* | Closes the Analyser TCL session. This function unloads from memory the class instance used to interact with the GP Series device in Analyser mode, and hence unconnects the Analyser session from the 'Smart Router'. |
| *conditions:* | |

**Unsupervised {Enable {Display 1}}**

| | |
|---|---|
| ***parameters :*** | ***Enable: valid values: 0 or 1.***<br>***Display: Optional parameter. Displays additional information messages when set to 1. Display disabled when set to 0. By default set to 1.*** |
| *returns:* | |
| *description:* | Enables or disables the unsupervised operating mode. In this mode, the traffic part reserved to the control between the host and the device the device is reduced to a minimum. As a consequence, more bandwidth is available for the data. |
| *conditions:* | |

**UseExtClock {}**

| | |
|---|---|
| ***parameters :*** | ***None.*** |
| *returns:* | |
| *description:* | Configures the device to operate with an externally supplied operating clock as reference. |
| *conditions:* | |

**UseIntClock {}**

| | |
|---|---|
| ***parameters :*** | ***None.*** |
| *returns:* | |
| *description:* | Configures the device to operate with the internal operating clock as reference. |
| *conditions:* | |

**Ver {}**

| | |
|---|---|
| ***parameters :*** | ***None.*** |

| | |
|---|---|
| *returns:* | A decimal value representing the software version. |
| *description:* | Returns the software version as a decimal value. Example: software version 1.04 ➔ returns 0x104 converted as a decimal value: 260. |
| *conditions:* | |

**WriteConfFile {FileName}**

| | |
|---|---|
| ***parameters :*** | ***FileName: specifies the procedure output file name (including path).*** |
| *returns:* | 0 when write successful. |
| *description:* | Writes the device configuration to a file. |
| *conditions:* | |

## 3.3 Reference Sequences and Scripts

Examples of the TCL scripts are provided with the TCL library. They can be accessed in the *Examples* group located under the *8PI Control Panel* program group created in your start menu during the installation of the application and driver on your computer.

**Figure 4: Program group with TCL script examples**