

8PI Control Panel

User's Guide

Revision 1.16 – 23 December 2011

FCC Compliance

This device complies with Part 15 of the FCC Rules. Operation is subject to the following conditions:

(1) This device may not cause harmful interference, and

(2) This device must accept any interference received, including interference that may cause undesired operation

Caution

Any changes or modifications of the device, not expressly approved by Byte Paradigm sprl, that could modify the FCC compliance of the device, could void the user's authority to operate the equipment.

Information to the user

Class A digital device, pursuant to Part 15 of the FCC Rules.

These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Table of Content

Table of Content 3
Table of Tables 4
Table of Pictures 4
References 7
History 8
1 About this guide 9
1.1 Guide Contents 9
1.2 Additional Documentation 9
1.3 Additional Support Resources 9
1.4 Typographical Conventions 9
2 Software Installation 10
2.1 System Requirements 10
2.2 Installation Wizard 10
3 8PI Control Panel 15
3.1 Starting the application 15
3.2 Smart Router 16
3.3 Hardware revisions 16
3.4 Multi-device and external voltage support 18
4 Arbitrary Digital Waveform Generator 20
4.1 Features 20
4.2 ADWG Application Page 21
4.3 Device Configuration 21
4.3.1 Control Lines Configuration & Mapping 21
4.3.2 Operating Clock 24
4.3.3 Data Bus Configuration & Mapping 26
4.3.4 Configuration 26
4.4 Static Mode Operation 27
4.4.1 Features 27
4.4.2 Interface Description 27
4.5 File Mode Operation 29
4.5.1 Features 29
4.5.2 Interface Description 30
4.6 File Format 33
4.6.1 Configuration & data file 33
4.6.2 Data sample file 39
5 State Analyser (logic analyser) 41
5.1 Features 41
5.2 Analyser Application Page 41
5.3 Device Configuration 41
5.3.1 Control Lines Configuration & Mapping 41
5.3.2 Operating Clock 44
5.3.3 Data Bus Configuration & Mapping 45
5.3.4 Configuration 45
5.4 Real Time Monitor Operation 46
5.4.1 Features 46
5.4.2 Interface Description 46
5.5 File Mode Operation 48
5.5.1 Features 48
5.5.2 Interface Description 49
5.5.3 Export File Format 52
6 JTAG Protocol 54
6.1 Features 54
6.2 JTAG Interface Signals 54

6.3 JTAG Low Level Protocol 55
6.4 SVF File Support 58
7 SPI 61
7.1 Features 61
7.2 SPI interface signals 62
7.3 SPI Application Page 65
7.4 Device Configuration in SPI mode of operation 65
7.4.1 SPI Configuration 65
7.4.2 SPI Mode Summary 66
7.5 SPI Master 67
7.5.1 Overview of the SPI Master protocols 67
7.5.2 Interface description 70
7.6 SPI Analyser 75
7.6.1 Overview of the SPI Analyser features 75
7.6.2 Interface description 76
7.7 File format 79
7.7.1 Configuration file 79
7.7.2 SPI Master script / log file 81
7.7.3 SPI Analyser export file 82
8 I²C 84
8.1 Features 84
8.2 I²C interface signals 84
8.3 I2C Application Page 87
8.4 Device configuration in I2C mode of operation 87
8.5 I2C Master 89
8.6 I2C Analyser 92
8.7 I2C Configuration / Data file format 93
8.8 Additional remarks about the generation / decoding of the I²C protocol 94
9 Appendix A: Devices Connectors 96
9.1 Check hardware revision 96
9.2 Connecting the GP or Xpress device 98
9.2.1 GP / Xpress device Connectors 98
10 Appendix B: TCL Console 101
10.1 Starting a TCL session from the 8PI Control Panel GUI 101
10.2 Starting a stand-alone TCL session 103
11 Appendix C: Wave Viewer 105
12 Appendix E – Overview of SPI accesses, options and parameters 107

Table of Tables

Table 1: List of typographical conventions 9
Table 2: Description of the 8PI Control Panel group 14
Table 3: Directory structure description 14
Table 4 : ADWG maximum data depth per run 21
Table 5 : Analyser maximum data depth per run 48
Table 6: JTAG interface pin mapping on the system connector 55
Table 7: SPI modes overview 67
Table 8 : SPI mode of operation data length / mode / operation 81
Table 9 : I2C Command Area description 90
Table 10 : SPI parameters 109

Table of Pictures

Figure 1: Installation wizard welcome window 10

Figure 2: Destination directory selection 11
Figure 3: Start menu selection 11
Figure 4: Installation summary 12
Figure 5: Installation progress bar 12
Figure 6: Installation completion message 13
Figure 7: Program Group for the 8PI Control Panel 13
Figure 8: 8PI Control Panel Overview 15
Figure 9: Smart Router window 16
Figure 10: Arbitrary wave configuration tab 22
Figure 11: Static operating mode for the arbitrary wave application 27
Figure 12: File mode operation for arbitrary wave application 30
Figure 13: Example of configuration & data file 37
Figure 14: Example of raw binary text file 39
Figure 15: Example of raw hexadecimal text file 39
Figure 16: Example of binary file 40
Figure 17: Analyser wave configuration tab 42
Figure 18: Static operating mode for the arbitrary wave application 46
Figure 19: File mode operation for state analyser application 49
Figure 20: Example of Raw Text file 52
Figure 21: Example of Raw Binary file 52
Figure 22: Pin mapping of the GP Series device system connector 54
Figure 23: JTAG low level control 56
Figure 24: Example of SVF file (read device IDCODE) 59
Figure 25: JTAG SVF file support 59
Figure 26: View of the GP-22050 top side 62
Figure 27: Pin mapping of SPI signals on the GP Series system connector (hardware revision 01) 62
Figure 28: Pin mapping of SPI signals on the GP Series system connector (hardware revision 02 and higher) 63
Figure 29: Pin mapping of SPI signals on the SPI Xpress connector (hardware revision 01) 63
Figure 30: Pin mapping of SPI signals on the SPI Xpress connector (hardware revision 02 and higher) 64
Figure 31: SPI configuration tab 65
Figure 32: 4-wires SPI protocol overview 68
Figure 33: 3-wires SPI protocol overview 69
Figure 34: SPI Master tab (SPI4 selected) 70
Figure 35: SPI Master tab (SPI3 selected) 73
Figure 36: SPI Analyser tab (SPI4 selected) 76
Figure 37: Example of SPI Configuration file 80
Figure 38: Example of Raw Data file 82
Figure 39: Example of Raw SPI file 83
Figure 40: Pin mapping of I²C signals on the GP Series system connector (hardware revision 01) 84
Figure 41: Pin mapping of I²C signals on the GP Series system connector (hardware revision 02 and higher) 85
Figure 42: Pin mapping of the I²C Xpress device system connector (hardware revision 01) 85
Figure 43: Pin mapping of the I²C Xpress device system connector (hardware revision 02 and higher) 86
Figure 44 : I2C configuration tab 87
Figure 45 : I2C Master tab 89
Figure 46 : I2C Master command area 89
Figure 47 : I2C Properties window 91
Figure 48 : I2C Analyser tab 92
Figure 49 : I²C analyser with error in transfer 95
Figure 50: View of the GP Series device top side 96
Figure 51: GP-22050 Top View – hardware revision starting with 01 98
Figure 52: GP-22050 Top View – hardware revision starting with 02 or 03 99
Figure 53: Tcl session start-up example 101
Figure 54: Opening a TCL console from the tools menu 102

Figure 55: TCL console shortcut in 8PI Control Panel program group 103
Figure 56: Tcl console at startup..... 104
Figure 57: Wave viewer menu 105
Figure 58: Wave viewer button..... 105
Figure 59: GTKWave viewer window..... 106
Figure 60 : Typical SPI access 107
Figure 61 : Master SCLK and SS# generation options..... 107
Figure 62 : 3 wires protocols overview & clock options 108

References

[1] GP-22050 device data sheet (ds_GP22050.pdf)
This document contains all the technical characteristics of the GP Series device.

[2] TCL scripting for ADWG application (ug_ADWGTclLib.pdf)
This document explain how to control the GP22050 using the Tcl scripting interface. It also contains a complete list of all available procedures to operate in ADWG mode.

[3] C/C++ library for ADWG application (ug__ADWGCLib.pdf)
This document describes the different functions available in the C/C++ library provided with the ADWG application.

[4] TCL scripting for Analyser application (ug_AnalyserTclLib.pdf)
This document explain how to control the GP22050 using the Tcl scripting interface. It also contains a complete list of all available procedures to operate in Analyser mode.

[5] C/C++ library for Analyser application (ug_AnalyserCLib.pdf)
This document describes the different functions available in the C/C++ library provided with the Analyser application.

[6] TCL scripting for JTAG application (ug_JTAGTclLib.pdf)
This document explain how to control the GP22050 using the Tcl scripting interface. It also contains a complete list of all available procedures to operate in JTAG mode.

[7] C/C++ library for JTAG application (ug_JTAGCLib.pdf)
This document describes the different functions available in the C/C++ library provided with the JTAG application.

[8] TCL scripting for SVF application (ug_SVFTclLib.pdf)
This document explain how to control the GP22050 using the Tcl scripting interface. It also contains a complete list of all available procedures to operate in SVF mode.

[9] C/C++ library for SVF application (ug_SVFCLib.pdf)
This document describes the different functions available in the C/C++ library provided with the SVF application.

[10] SPI Xpress device data sheet (ds_SPIXpress.pdf)

[11] Wave Generator Xpress data sheet (ds_WaveGeneratorXpress.pdf)

[12] C/C++ library for SPI application (ug_SPICLib.pdf)

[13] TCL scripting for SPI application (ug_SPITclLib.pdf)

[14] I2C Xpress device data sheet (ds_I2CXpress.pdf)

[15] C/C++ library for I2C application (ug_I2CCLib.pdf)

[16] TCL Scripting for I2C application (ug_I2CTclLib.pdf)

History

Version	Date	Description
1.00	30-Sep-2005	Initial revision
1.01	02-Nov-2005	Analyser and JTAG protocol description included in the document
1.02	16-Feb-2006	FCC compliance note added Stand-alone TCL console
1.03	9-Nov-2006	Update for software release 1.04
1.04	5-Mar-2007	Update with trigger auto rearm functions
1.05	31-May-2007	Added edge triggering description
1.06	13-Aug-2007	Some corrections
1.07	29-Aug-2007	Added trigger positioning for Analyser mode
1.08	13-Nov-2007	Update for GP series introduction
1.09	18-Mar-2008	Reviewed document to include all modes of operation / for Xpress series introduction
1.10	10-Oct-2008	Added section for I2C mode of operation
1.11	08-Dec-2008	Added information for downloading and installing potentially missing Visual C++ libraries
1.12	17-Feb-2010	Complete review/update
1.13	20-June-2010	Updated max length / details for SPI mode of operation
1.14	08-July-2010	Added details about max SPI parameters
1.15	26-Nov-2010	Added description for multi-device use
1.16	23-Dec-2011	Update for device voltage selection

1 About this guide

1.1 Guide Contents

This document describes the usage of the 8PI Control Panel application. Up-to-date documentation of the 8PI Control Panel host software is available on the Byte Paradigm web site at: http://www.byteparadigm.com.

1.2 Additional Documentation

1.3 Additional Support Resources

For additional support, submit your request to support@byteparadigm.com or refer to Byte Paradigm support page at: http://www.byteparadigm.com (click on 'Support' in the top menu).

1.4 Typographical Conventions

Unless explicitly notified, the following rules are used in the document.

Table 1: List of typographical conventions

The bus index are always defined and represented with the most significant bit placed on the most left position. The most right bit represents the least significant bit.
A paragraph marked with the "information" sign contains an important remark or advice for the use of the software and/or device.
A value starting with prefix "0x" represent an hexadecimal value (e.g.: 0x123A).

2 Software Installation

2.1 System Requirements

To be able to correctly install and use the application on your computer, your system must fulfil the following minimum requirements:

- PC running Windows 2000, Windows XP or Windows Vista
- 60 MB of free disk space
- USB 2.0 or 1.1 host controller

ⓘ *You must have administrator rights on your computer to ensure correct installation of the application*

ⓘ *The application and device drivers are not available for Mac and Linux. For further information concerning the support of these operating systems, refer to the Byte Paradigm web site.*

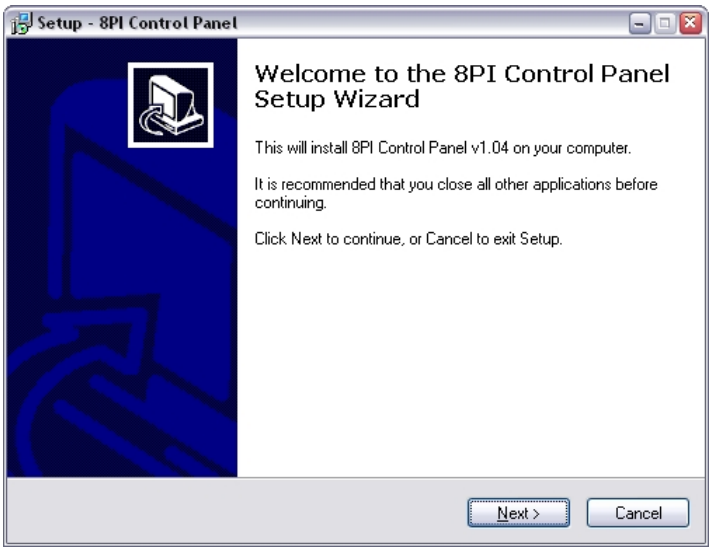
2.2 Installation Wizard

ⓘ *When updating the 8PI Control Panel application, it is recommended to uninstall the previous version before proceeding with the upgrade.*

ⓘ *At any time during the installation process you can click the Back button to return to the previous installation window.*

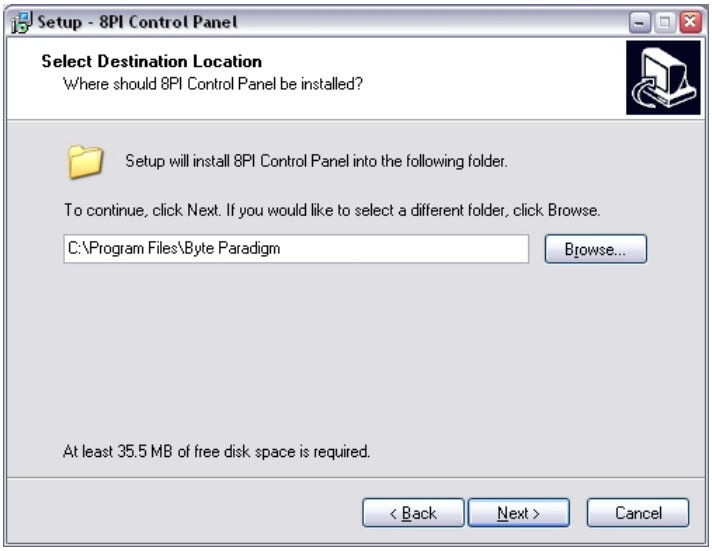
To install the program, run the *Setup.exe* file from the CD-ROM delivered with the purchased Byte Paradigm kit. If you downloaded the *8P Control Panel* from the Internet, unpack the downloaded archive to a directory and run the *Setup.exe* file from this directory. The welcome window opens, as depicted on Figure 1. Click on the *Next* button to proceed to the next installation step.

Figure 1: Installation wizard welcome window



You are then requested to choose the target installation directory for the 8PI Control Panel application. The wizard proposes to create a new directory into the default *Program Files* directory, as shown on Figure 2. If you do not wish to install the 8PI Control Panel environment in the proposed directory, type the desired path in the upper field or navigate to another directory using the *Browse* button.

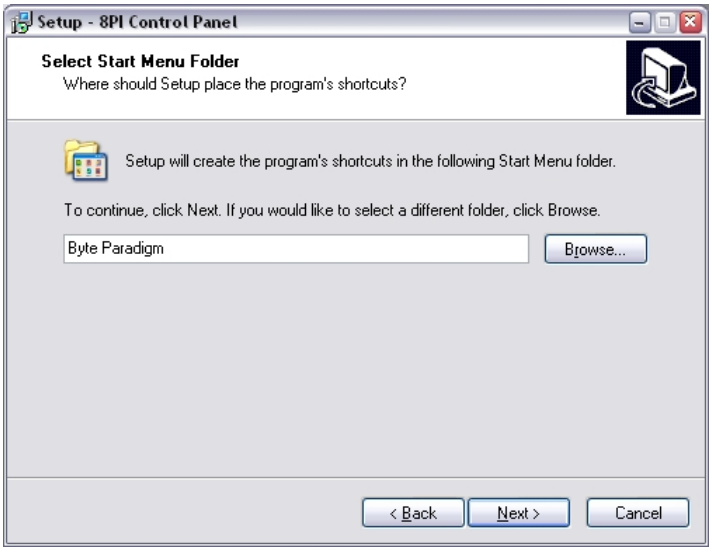
Figure 2: Destination directory selection



It is recommended to install the 8PI Control Panel into its own directory to avoid mixing files from different applications. Therefore, check that the directory you have entered in the upper filed does not already exist.
Once the directory is selected, click on the *Next* button to proceed to the next installation window.

In this final window, you are asked to enter the *Start Menu* folder name that will hold the program's shortcuts. This shortcut will appear in your *Start -> Programs* menu. Then click on the *Next* button to get the installation summary.

Figure 3: Start menu selection



A summary of the installation information is now displayed, as shown at Figure 4. Click on the *Install* button to start installing the *8P Control Panel* software environment according to the parameters shown in the window. If you want to change any parameter, click on the *Back* button as many times as necessary. During the installation, a progress bar is displayed, as well as information about the files being installed (refer to Figure 5).

Figure 4: Installation summary

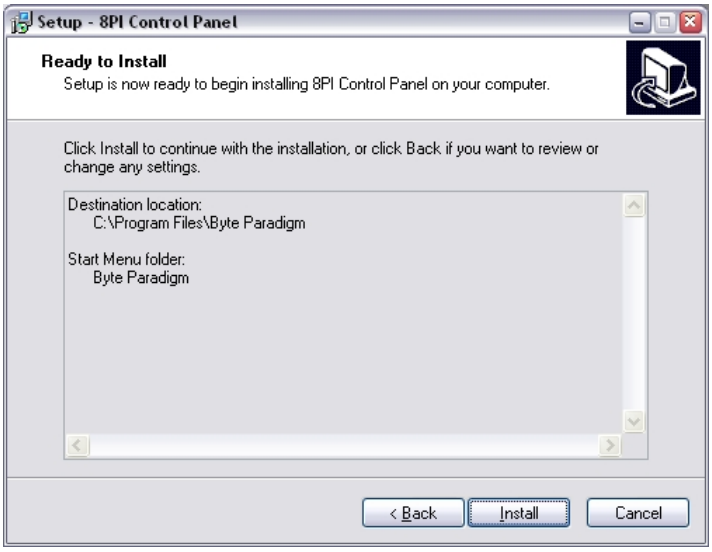
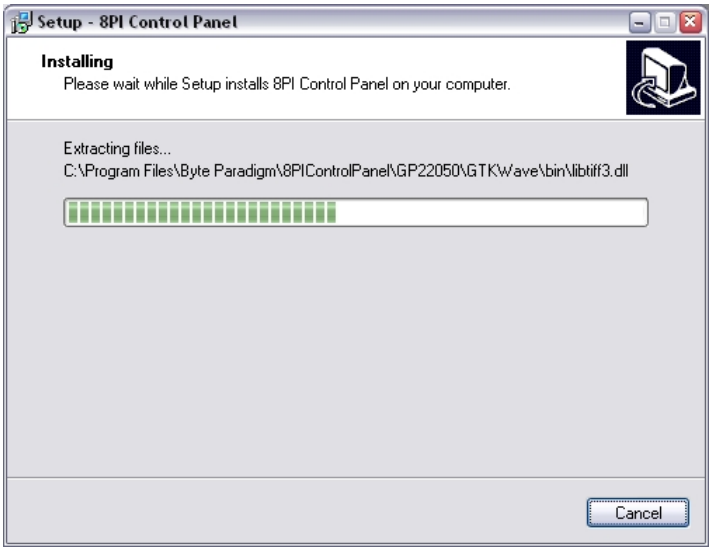
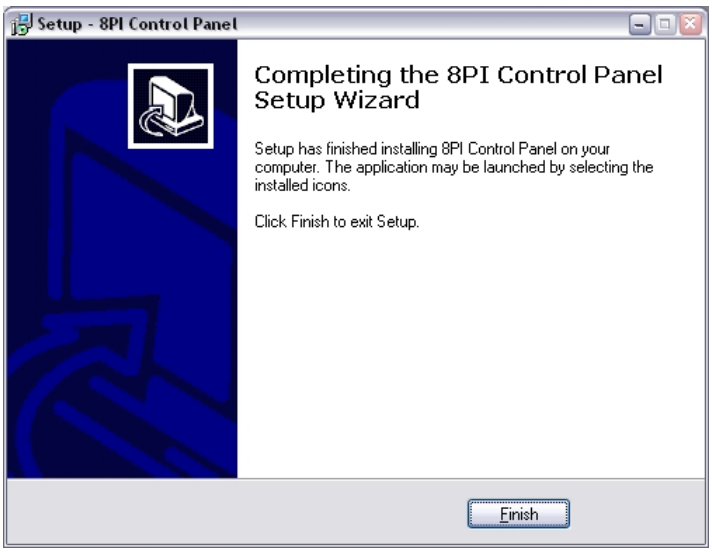


Figure 5: Installation progress bar



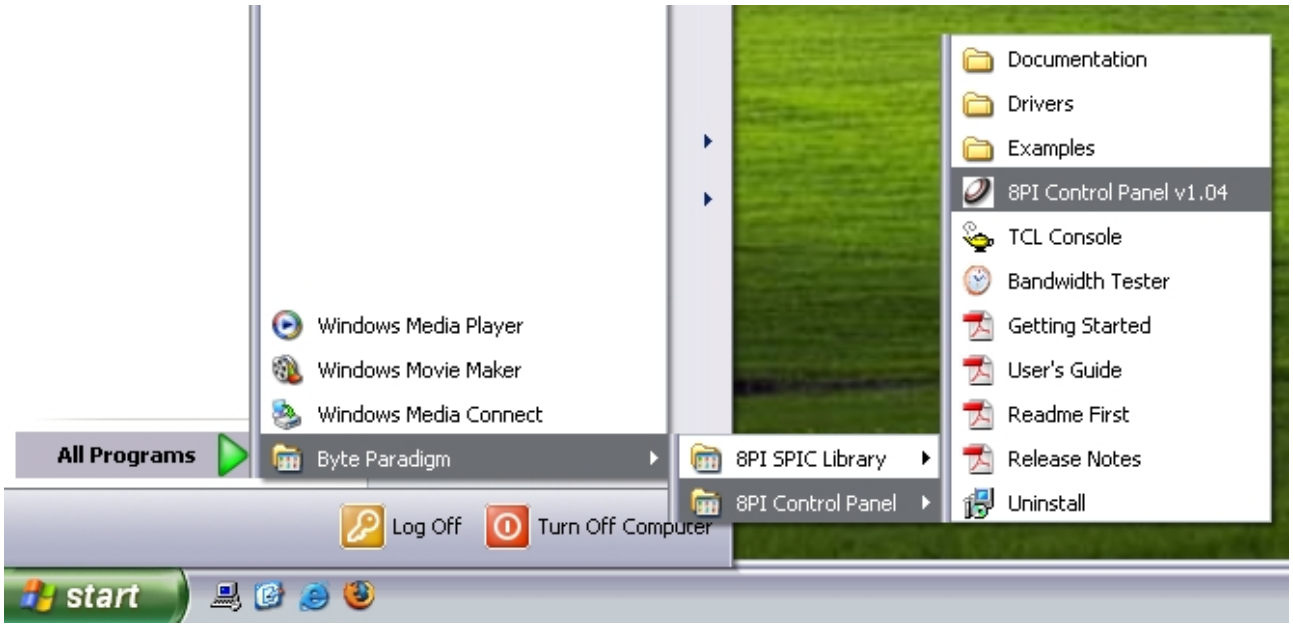
When the installation is successfully completed, a completion message is displayed, as shown in Figure 6. Click on the *Finish* button to close the window and finish the installation process.

Figure 6: Installation completion message



In your start menu, the program group "8PI Control Panel" is now available (refer to Figure 7). The content of this group is described in Table 2.

Figure 7: Program Group for the 8PI Control Panel



At 8PI Control Panel start, if you receive the following message: ***"cannot execute the specified program",*** *please go to the following link: http://www.microsoft.com/downloads/details.aspx?FamilyId=32BC1BEE-A3F9-4C13-9C99-220B62A191EE&displaylang=en – or look for 'Microsoft Visual C++ 2005 redistributable package (x86)' on the web. This link gives instruction for installing missing libraries on your PC.*

Table 2: Description of the 8PI Control Panel group

Item	Description
[Documentation]	Directory containing all the documentation, guides and application notes delivered with the application package.
[Driver]	Directory containing the driver files needed to correctly install the GP SERIES device. These files will be requested the first time the device is connected to the computer.
[Examples]	Directory containing some examples of TCL scripts, configuration/data files and output files
8PI Control Panel	Shortcut to launch the *8PI Control Panel* application
TCL Console	Shortcut to open a TCL console to control the GP Series device
Bandwidth Tester	Shortcut to launch the *USB Bandwidth Tester* tool for the GP Series device
Readme First	Document to read before to connect the device to a computer and to use it.
Getting Started	Guide to help installing and using the GP Series device
User's Guide	Complete documentation describing the different applications available in the *8PI Control Panel* for the GP Series device.
Release Notes	List of updates and changes included since last release.
Uninstall	Shortcut to the uninstall application to remove the application from the computer.

When the installation is completed, the 8PI Control Panel directory structure has been created on your computer. Table 3 gives a brief description of the content of each directory.

Table 3: Directory structure description

Directory	Contents
[8PI Control Panel]	Root directory containing the *8PI Control Panel application.* It also contains all DLL and others related files.
[Doc]	Directory containing all the PDF documentation provided with the software package.
[Drivers]	Driver files for the GP Series device.
[Examples]	Directory containing reference files and examples: TCL scripts for the different applications available in the software package and example of input and output files handled by the *8PI Control Panel*.
[GTKWave]	Directory containing the GTKWave viewer application
[include]	Directory containing the different C/C++ header files needed to integrate the supplied C/C++ library in a user application.
[lib_vc80]	Directory containing the different C/C++ library files needed to integrate the supplied C/C++ library in a user application.
[tcl]	Application files and libraries required for the TCL console

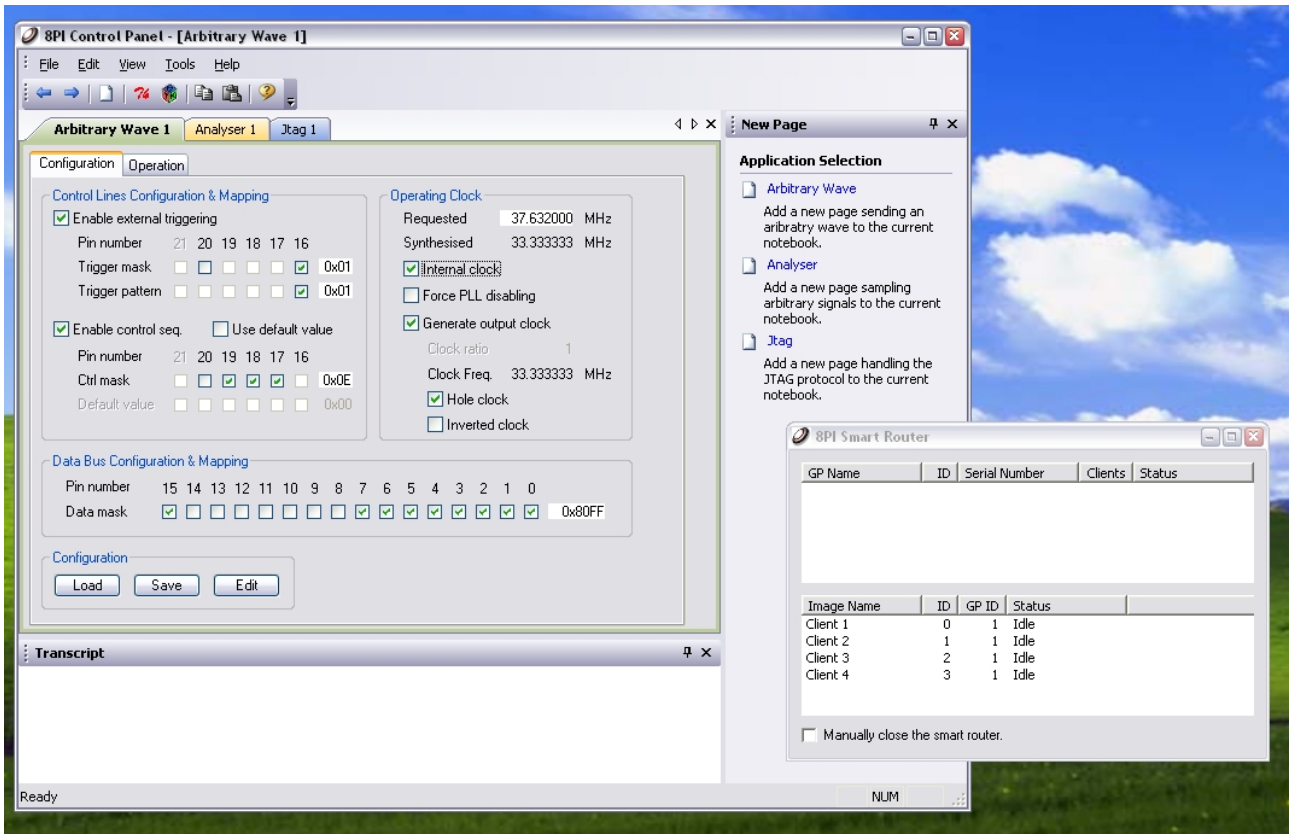
3 8PI Control Panel

3.1 Starting the application

The application is started from the *8PI Control Panel* application group created in the program menu during the installation. The application is started by clicking the *8PI Control Panel v1.xx* shortcut, where xx indicating the software release number (for example v1.04, refer to Figure 7).

When the *Control Panel* is started, the *Smart Router* is launched automatically, if it is not already running. The *Smart Router* is started each time an application tries to access the GP Series or Xpress Series device. Refer to section below for more details on the *Smart Router* features.

Figure 8: 8PI Control Panel Overview



The control panel is an environment in which different applications controlling the GP Series or Xpress Series device can be started. The interface is composed of 3 main areas.

- The *New Page* pane allows creating a new page session for each of the available modes of operation. The list of modes of operation depends on the installed licenses.
- The *Transcript* window is used to display log and status messages
- The *Application Tabs*. Each tab is used to control a GP Series or Xpress Series device mode of operation. Several tabs can be opened simultaneously, but only the active tab actually controls the device. Each time the user switches between tabs, the device is automatically set up to operate in the defined mode. Xpress Series devices only function in one single mode of operation.

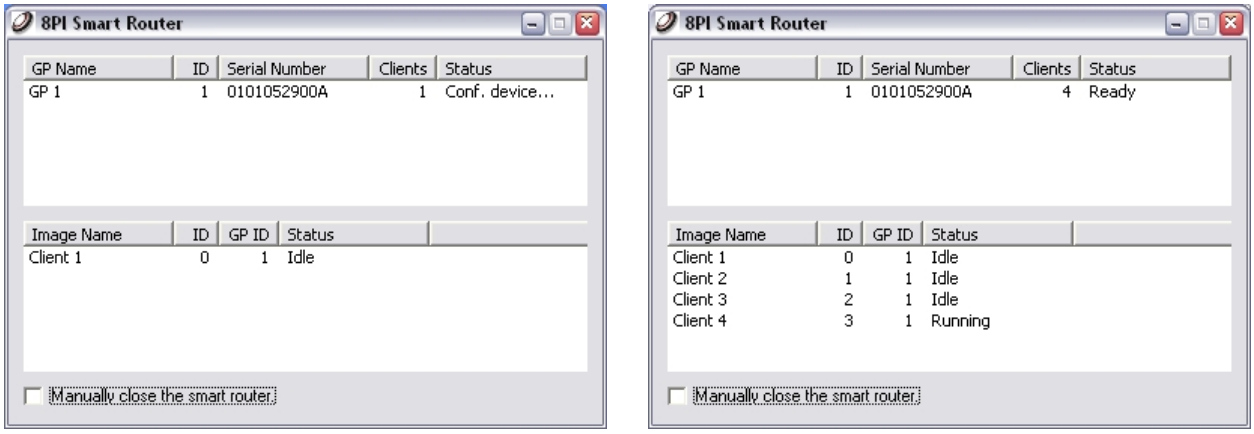
Version the 8PI Control Panel operates only in single device mode. This means that it is not possible to connect more than one GP Series device to the same PC simultaneously.

3.2 Smart Router

Using the graphical *8PI Control Panel* interface is not the only way to control the GP Series or Xpress Series devices. Jobs can be executed from a TCL console or a user-developed application using the provided C/C++ libraries. Multiple modes of operation sessions can be started simultaneously, but only one application can access the device at a time. The *Smart Router* is an arbiter controlling the access to the connected device. When an application requests an access to the device, the *Smart Router* automatically configures the connected device into the mode suitable for that application before granting access. This is performed automatically and transparently for the user, and is performed only if parameters differ from one application to the other.
The router cannot be stopped as long as a control application is running. When no more application is running, it is automatically closed, unless you choose to close it manually by selecting the corresponding check box.

By default, the *Smart Router* agent is minimised when running and a tray icon is displayed.
The router window can be opened by clicking on the tray icon. It is composed of an upper window containing device information and status. The lower window contains the status of the active clients able to access the device.

Figure 9: Smart Router window



3.3 Hardware revisions

8PI Control Panel is used with all Byte Paradigm's Xpress and GP series devices.
The device model name defines the set of functionalities of the 8PI Control Panel software that can be used. GP and Xpress devices also exist out of various hardware revisions.

To know the hardware revision of your device, check the serial number located on a label on the back of the device. The 2 first digit define the hardware revision.

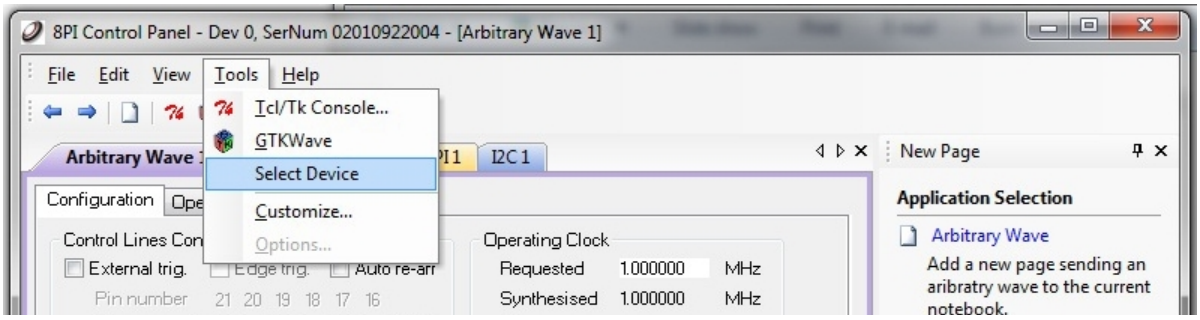
According to the hardware revision, the following elements vary:

Serial number 2 first digits	***In/out clocks on user interface connector***	***External power source selection***
01	**ClkOut** pin is located on pin labelled D21 / C5 **ClkIn** pin is located on pin labelled D20/C4	Using external power source for I/Os requires properly **positioning** the power selection jumper AND applying the external voltage onto the VEXT pins of the user interface connector
02	**ClkOut** pin is dedicated to output clock and is labelled CkOut (GP / Wave Generator Xpress) or SCLKo (SPI Xpress) **ClkIn** pin is dedicated to input clock and is labelled CkIn	Using external power source for I/Os requires its application on the side power source connector after having **removed** the jumper.
03		

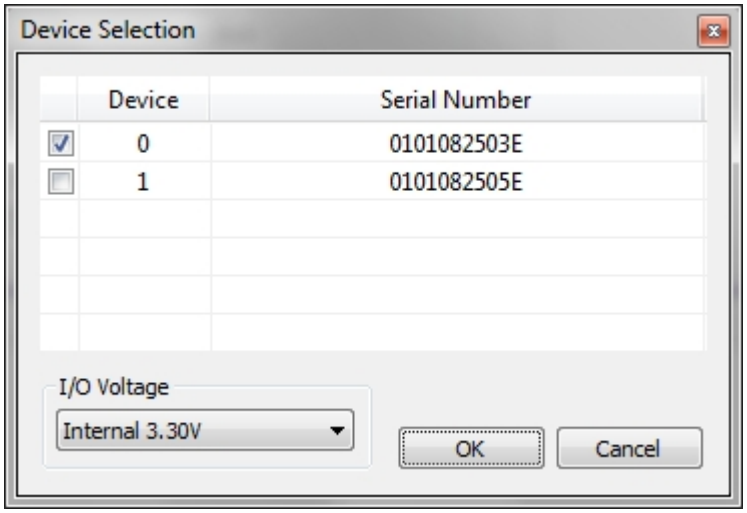
For more information, please refer to the devices data sheet or the 'Getting Started' document provided with your device or in this document's Appendix A.

3.4 Multi-device and external voltage support

From 8PI Control Panel version 2.00, it is possible to control more than one device from 8PI Control Panel. To do so from the GUI, please follow this procedure:



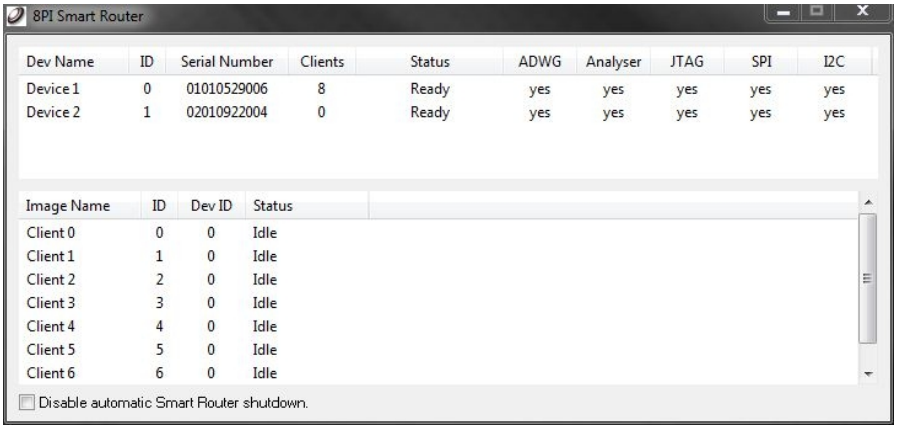
From the 8PI Control Panel main window, click on **Tools> Select Device.**
A window pops up with the list of the devices connected to your PC.

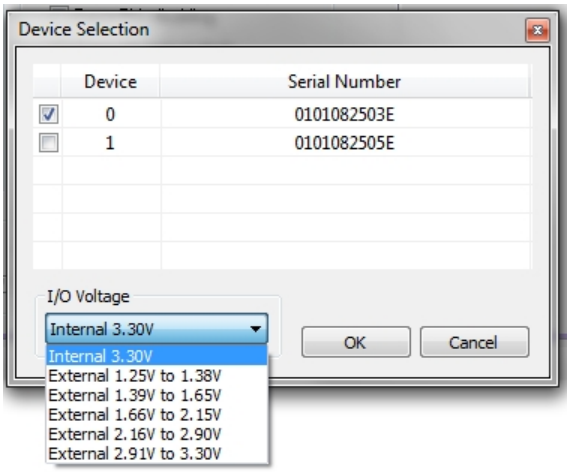


Click on the tick box in the left column corresponding to the device that you wish to select. Click on **Ok.**

! You need to have the licenses of all connected devices properly installed to be able to use them !

Here is an example of how the Smart Router Window looks like when multiple devices are connected:





From 8PI Control Panel version 2.03 and higher, the Device Selection window also features a drop-down list used to specify the voltage range used for the device I/Os. This setting optimizes the device's I/O buffer characteristics for the selected I/O voltage. Available choices are:

Internal 3.30 V	This is the default choice. The I/Os will use the device internal I/O voltage supply with 3.3V signaling.
External 1.25V to 1.38V	If you are using an external voltage supply for the device's I/Os, select the corresponding I/O voltage range.
External 1.39V to 1.65V	
External 1.66V to 2.15V	
External 2.16V to 2.90V	
External 2.91V to 3.30V	

Remark:

Selecting the I/O voltage in this drop-down list DOES NOT automatically switch the actual value of the I/O voltage !

When using a setting different from the default internal 3.3V, you MUST supply the voltage on the external voltage connector (see your device data sheet).

4 Arbitrary Digital Waveform Generator

Applies to: GP Series devices and Wave Generator Xpress device.

4.1 Features

This application is delivered to use the GP Series device as a pattern generator and to control the Wave Generator Xpress device. The graphical interface provides a fast and easy way to configure and control the device. In this particular unidirectional mode, the GP Series device is configured to drive signals to the target user's system and it is not possible to capture or sample incoming signals. Wave Generator Xpress only functions in this mode of operation.

The following list describes the main features that can be controlled through the graphical interface in the ADWG mode.

- 16 bits output data lines available to generate up to 100 MByte / 100 MSample of arbitrary waveform
- 6 bits control lines available to generate periodic patterns
- Up to 6 bits to detect external trigger pattern
- Internal or external reference clock operation
- Programmable operating frequency up to 50MHz
- Loop mode to generate repetitive patterns
- Output clock generation with a programmable ratio
- Clock continuity, phase and polarity control
- Configuration save/restore
- Static operating mode to apply static levels on selected pins
- File based operating mode reading the configuration and data waveform from text and/or binary files. This mode is an easy to use mechanism to apply simulation results or sampled data to the system connector of the GP Series device

These features are also accessible through the provided C/C++ DLL and TCL library. Refer to [3] and [2] for more details on how to use the C/C++ library and the Tcl scripting facilities.

From the *8PI Control Panel* interface, the pattern generator can be used in static or in a dynamic mode. The *Static* mode is used to apply constant level on the device connector. The user defines a static pattern and applies it on the connector. The pattern remains stable until a new pattern is sent or the device configuration is changed. This operating mode can be used for low level and basic debug purposes or to emulate switches.
The *Dynamic* mode is referenced as the *File* mode. In this mode, the patterns to apply on the different pins of the device connector are stored in a file. This file contains the complete device configuration and samples values. The application reads the file and applies the user defined sequence to 16 data lines. According to the used lines, the source file is limited to 100 MByte and 100 MSample (refer to Table 4). This mode can be used to apply test patterns to the target system or to emulate unidirectional protocol.
The *File* mode also gives the possibility to use the 6 *Control* lines of the device. All or some of them can be configured as inputs to detect a trigger pattern before generating the requested pattern. This gives the possibility to synchronise the pattern generation with external events coming from the target system. If not used as trigger inputs, two control lines can be used as clock pins: one to generate a programmable output clock, and the other one to receive an external reference clock to generate the pattern on the data lines. The control lines can also be used to generate a repetitive pattern each time a new sample is sent out. The periodic pattern is stored once in the device and sent out each time a new sample is applied on the device connector. This can be used, for example, to control the chip select and write enable pins of a microprocessor interface.

Table 4 : ADWG maximum data depth per run

Number of pins used on the connector	*Maximum data depth per run (Samples)*	*Maximum data depth per run (Byte)*
1 to 8	100 MSample	100 MByte
9 to 16	50 MSample	100 MByte

4.2 ADWG Application Page

The ADWG mode of operation page can be opened using the *File* > *New* application menu and then by selecting *Arbitrary Wave* in the dialog. Figure 10 gives an overview of the ADWG page. The page can also be opened using the *New Page* pane. If the pane is not visible go to menu *View* > *New Page Pane*. When opening an ADWG configuration file using the *File* > *Open* menu, a new page is automatically created.

4.3 Device Configuration

4.3.1 Control Lines Configuration & Mapping

The system connector of the GP Series device gives access to 16 data lines and 6 control lines (refer to [1] for more details on the system connector and to section "Appendix A: Devices connector mapping"). This group of settings is used to configure the operating mode of the 6 control lines.

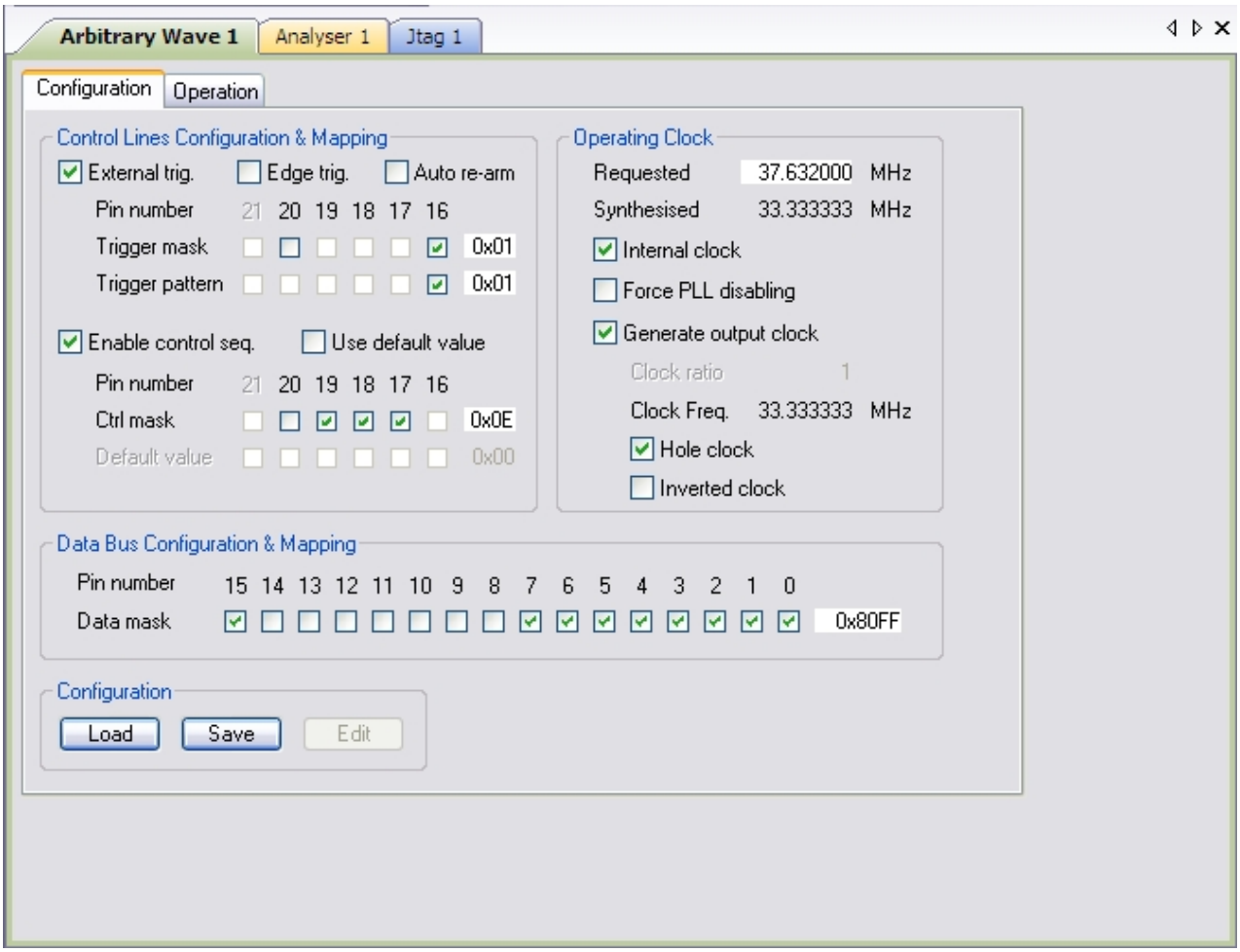
Even if the ADWG is an output only application, the control lines can be configured as input or output. When operating in output mode, they can be used to:

- Apply a static default 0 or 1 logic level
- Generate a periodic waveform sequence
- Generate a clock signal related to the system clock

When operating as input signals, they can be used:

- As external triggering lines to start a job
- As input for an external reference clock

Figure 10: Arbitrary wave configuration tab



EXTERNAL TRIG.

☑ When checked, it enables the use of an external trigger to start the ADWG sequence. At least one control line has to be selected to operate as an input and be used as trigger event. Several lines can be used to define the trigger event. When the defined pattern on the selected control lines is detected, the trigger event is generated and the ADWG sequence is started. There is a latency of 3 clock cycles between the trigger pattern detection and data transfer start.

☐ When not checked, no external trigger event is used and the ADWG sequence is started as soon as requested by the user.

EDGE TRIG.

☑ When checked, it selects the 'edge triggering mode'. In this mode, the ADWG run is triggered when the selected trigger lines transit to the programmed trigger pattern.

☐ When not checked, the 'level triggering mode' is selected. In this mode, the ADWG run is triggered when the programmed trigger pattern is present on the selected trigger lines.

AUTO RE-ARM

☑ When checked, it enables the trigger auto-rearm feature. In this mode, the ADWG sequence is repeated upon each arrival of the programmed external trigger.

☐ When not checked, the trigger is not rearmed after each run.

TRIGGER MASK

☑ When selected, the corresponding control line is selected to be used as input for the external triggering. The control line can not be used to drive a default logic level or to generate a periodic waveform. It remains operating as an input during the complete ADWG sequence.

☐ When not selected, the pin is not reserved for external triggering and is operating as an output.

TRIGGER PATTERN

The *trigger pattern* can only be defined if at least one control line is selected as input in the external *trigger mask*. The *trigger pattern* defines the logic levels that must be detected on the input control lines to generate the trigger event. The level applied on the control lines not selected for the trigger is ignored while detecting the trigger pattern.

☑ When selected, logic 1 is associated to the control line.

☐ When not selected, logic 0 is associated to the control line.

ⓘ *Instead of clicking on the different check box of the trigger mask and pattern, the expected values can be directly entered in hexadecimal format with prefix 0x in the corresponding field.*

ENABLE CONTROL SEQUENCE

When configured as output, the control lines can be used to drive static levels or a periodic sequence. If the periodic sequence is enabled (and *Use default value* option disabled), each time a data is sent out the sequence on the control lines is generated. For example, if a sequence of 8 clock cycles is defined for the control lines, each time a data is sent out, the data value will remain stable for 8 clock cycles while the control sequence is generated. This feature can be used full to generate write enable or chip select like signals.

☑ When checked, it enables the used of the control lines to apply static logic levels or periodic waveform. At least one control line has to be selected in the *control mask*.

☐ When unchecked, all control lines not used for the *trigger mask* are set in Hi-Z.

ⓘ *The periodic control sequence is only available for the file mode. The pattern to repeat is defined through the configuration file only. The length of the control sequence can be defined between 1 and 250 cycles.*

USE DEFAULT VALUE

☑ When the *enable control sequence* is checked, the *Default Value* flag forces the use of static logic level on the output control lines. The logic level of each output control line is defined using the *default value* pattern. The default levels are applied on the control lines with the first data sent out on the system connector.

☐ When not checked, a user defined periodic waveform is applied on the output control lines instead of the default levels. Refer to section "4.5 File Mode Operation" for more details on the definition of the periodic waveform.

CONTROL MASK

☑ When selected, the corresponding control line is configured as an output and can be used to drive the static level or a periodic waveform depending on the *Use default value* parameter. Only the control lines not used for the external trigger can be used. They remain configured as an output during the complete ADWG job.

☐ When not selected, the corresponding control line is set in Hi-Z if not used for the external *trigger mask*.

DEFAULT VALUE

The *default value* can only be defined if the *enable control sequence* and *use default* value are activated, and if at least one control line is selected as output in the *control mask*. The *default value* defines the static logic levels that must be applied on the selected output lines.

☑ A default logic 1 is defined when checked

☐ A default logic 0 is defined when unchecked

ⓘ *Instead of clicking on the different check box of the control mask and default value, the expected values can be directly entered in hexadecimal format with prefix 0x in the corresponding field.*

4.3.2 Operating Clock

The operating clock can be internally generated or provided by the user as an external reference clock. When the clock is internally generated, it is derived from a 200 MHz internal reference clock. The operating clock is then generated using a divider. Only integer ratio can be programmed and so not all frequencies can be achieved exactly. The accuracy of the clock divider is 4ns. The achieved frequency is displayed under the requested one.
When the external clock reference is used, the operating clock is equal to the external clock. No divider is used in this case. In external clock mode, the requested frequency field must be filled with the frequency of that clock. This is mandatory to ensure a correct configuration of the device PLL (refer to *Force PLL disabling*).

REQUESTED OPERATING CLOCK

This field defines the frequency of the operating clock used in the device. The frequency must be defined in a range between the maximum clock frequency of your device (e.g: 50 MHz for Wave Generator Xpress and GP-22050, 100 MHz for GP-24100 device) and 800Hz. The value entered in this field must be a decimal value having a maximum of 6 decimal digits and representing the frequency in MHz.

INTERNAL CLOCK

☑ When checked, the 200 MHz internal reference clock is used to generate the operating clock. By default, the internal reference clock is selected.

☐ When not checked, an external reference clock must be provided by the user. For hardware revision 01, the clock has to be applied on pin D20 of the system connector. When the external clock is enabled, control line D20 is reserved as input for that clock. This control line can not be used anymore for external triggering nor for applying static or periodic levels. For hardware revisions 02 and higher, there is a dedicated CLKIn input pin on the device connector. Please refer to the device data sheet for the connector pin mapping.

ⓘ *If an external clock is used, it must be stable, continuous and applied to the device before starting to use it.*

FORCE PLL DISABLING

Under defined circumstances, the GP Series device can use internal PLL to reduce clock-to-output delay of the data and control lines with respect to the operating clock. The operating clock frequency must be in the range 'Max clock' (100 MHz for GP-24100 device, 50 MHz for Xpress and GP-22050 devices) to 18 MHz to be able to use the PLL. The PLL can be used for both internal and external clock.

The user has the possibility to disable the PLL or to let the application turning it ON or OFF automatically in function of the requested operating frequency.

☑ The PLL is disabled and never used when checked.

☐ The PLL is automatically enabled in function of the operating frequency when unchecked.

ⓘ *The external clock must be stable, continuous and applied to the device before starting to use it.*

ⓘ *If the PLL is used, refer to [1] to get the timing and jitter specifications of the external clock.*

GENERATE OUTPUT CLOCK

☑ When checked, an internally generated clock can be applied on output pin D21 of the system connector. For device hardware revision 01, connector pin D21 is reserved for this clock. The control line can neither be used for external triggering nor for applying default or other periodic logic levels. For device hardware revisions 02 and higher, there is a dedicated output clock pin marked ClkOut – please refer to the device data sheet for the connector pin mapping.

☐ When not checked, no clock is sent out. In case a device hardware revision 01 is used, pin D21 and the corresponding control line can then be used for external trigger or driving logic level. If the device hardware revision is 02 or higher, using or not using this output clock does not affect the control lines availability.

CLOCK RATIO

If the output clock is enabled, then its frequency is determined using an integer ratio that can be defined between 1 and 65535. When set to 1, the output clock is the image of the operating clock (internal or external in function of the selected operating mode). If the ratio differs from 1, the integer ratio value is used to divide the operating clock.

The ratio value must be defined as a decimal number.

The achieved frequency of the output clock is displayed under the requested ratio.

HOLE CLOCK

☑ When checked, the output clock is generated as a "hole" clock. Clock edges are only sent out when a data is applied on the output. When no data is sent out, the clock signals remains low. This can be useful when the data stream is not continuous.

☐ When unchecked, the output clock is continuous and independent of the data stream.

ⓘ *The hole clock feature can only be used when the clock ratio is equal to 1.*

INVERTED CLOCK

☑ When checked, the output clock is generated on the falling edge of the operating clock and is then in opposite phase with the data. The data are sent out in phase with the falling edge of the output clock.

☐ When unchecked, the output clock is generated on the rising edge of the operating clock and is then in phase with the data.

4.3.3 Data Bus Configuration & Mapping

The 16 pins (D0 to D15) of the system connector are reserved for user data. In the ADWG application, these lines can either be used as output, either be disabled and set in Hi-Z state. The user just has to click on the check box associated to a data output pin to enable it. An hexadecimal value with the prefix "0x" can also be directly entered in the field next to the check box.

4.3.4 Configuration

The user configuration can be saved to a file. This gives the possibility to retrieve a configuration by loading a saved configuration file. The configuration is saved in a text file. Refer to section "4.5 File Mode Operation" for more details concerning the file format.

LOAD

Load a configuration from a file.

SAVE

Save the current configuration to a file

EDIT

Open the loaded configuration file in a text editor to let the user modify it. When the text editor is closed, the user is prompt to reload the configuration.

4.4 Static Mode Operation

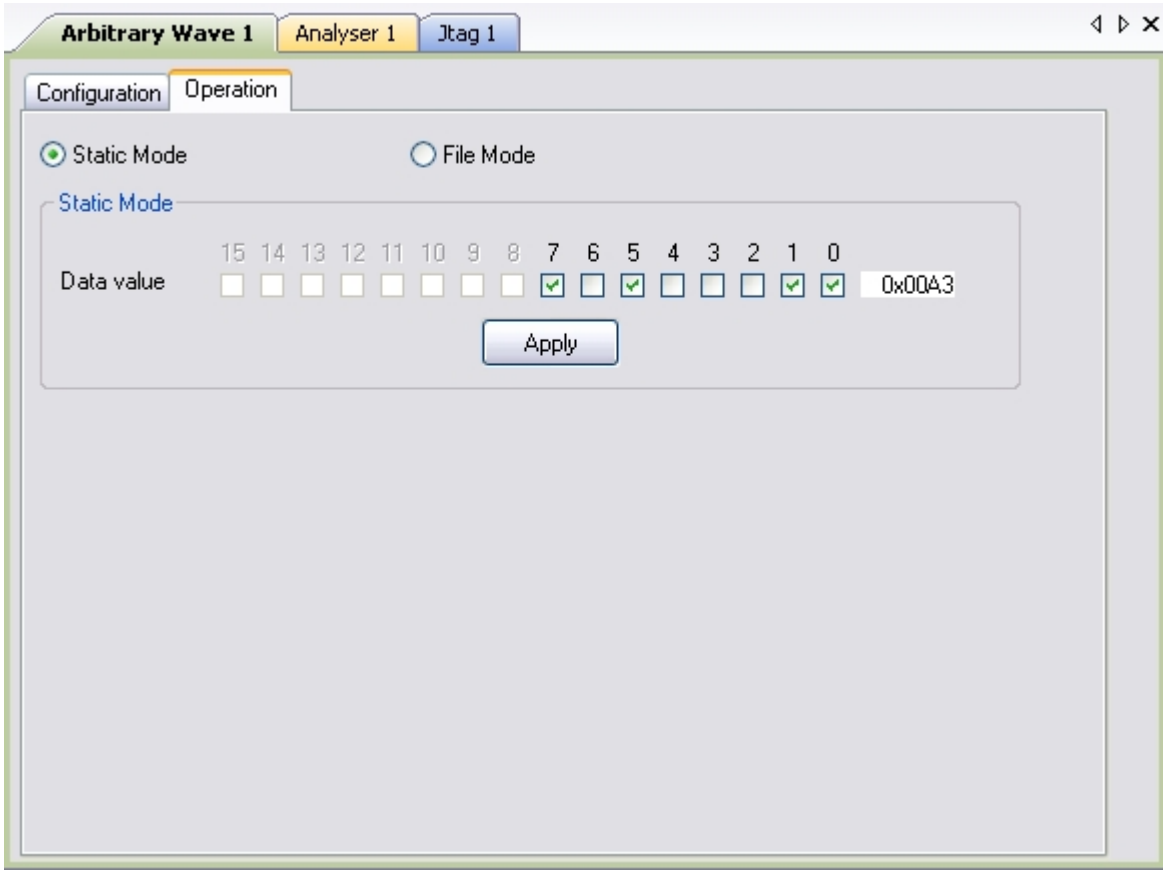
4.4.1 Features

The static operation mode gives the possibility to define constant logic levels on the 16 data and 6 control lines. The user defines the pattern and applies it to the system connector. The pattern remains stable until a new pattern is defined and applied by the user.
The main features of this operating mode are:

- 16 output data lines available
- 6 output control lines available. One can be reserved if the output clock is generated.
- A continuous output clock can be generated with a ratio between 1 and 65535 with respect to the defined operating clock.
- In hole clock operating mode, a single clock edge is generated when the new pattern is applied on the system connector. In that case, the ratio must be equal to 1.
- When working with a ratio equal to 1, the output clock rising edge can be generated in phase with the output data or half a period of the operating clock after the output data. When the ratio differs from 1, the phase relation between the data and the output clock is unpredictable.
- External triggering feature can not be used

4.4.2 Interface Description

Figure 11: Static operating mode for the arbitrary wave application



STATIC MODE

☑ Enables the static operating mode when checked.

☐ This button is automatically cleared when the *File Mode* is selected.

FILE MODE

☑ Switches to File Mode when checked (refer to section "4.5 File Mode Operation".

☐ This button is automatically cleared when the *Static Mode* is selected.

ⓘ *It is not possible to switch between static and file mode while a job is running. All data transfers must be terminated before to switch mode.*

DATA VALUE

These 16 check boxes are used to define the pattern to apply on the system connector. Only the bits enabled by the *Data mask* in the configuration tab are applied on the output pins. Disabled outputs are set to Hi-Z.

☑ Defines a logic 1 level on the corresponding data bit when checked.

☐ Defines a logic 0 level on the corresponding data bit when unchecked.

ⓘ *Instead of clicking on the different check box of the data value, the expected value can directly be entered in hexadecimal format with prefix 0x in the field on the right side.*

APPLY

The user defined pattern is applied on the system connector when this button is clicked. Each time the button is clicked, the device is reprogrammed if any of the configuration settings is changed and the data pattern is sent out. The data pattern remains stable until the *Apply* button is used to define a new data value.

If the output clock is generated in continuous mode, it will be started the first time the *Apply* button is pressed.

4.5 File Mode Operation

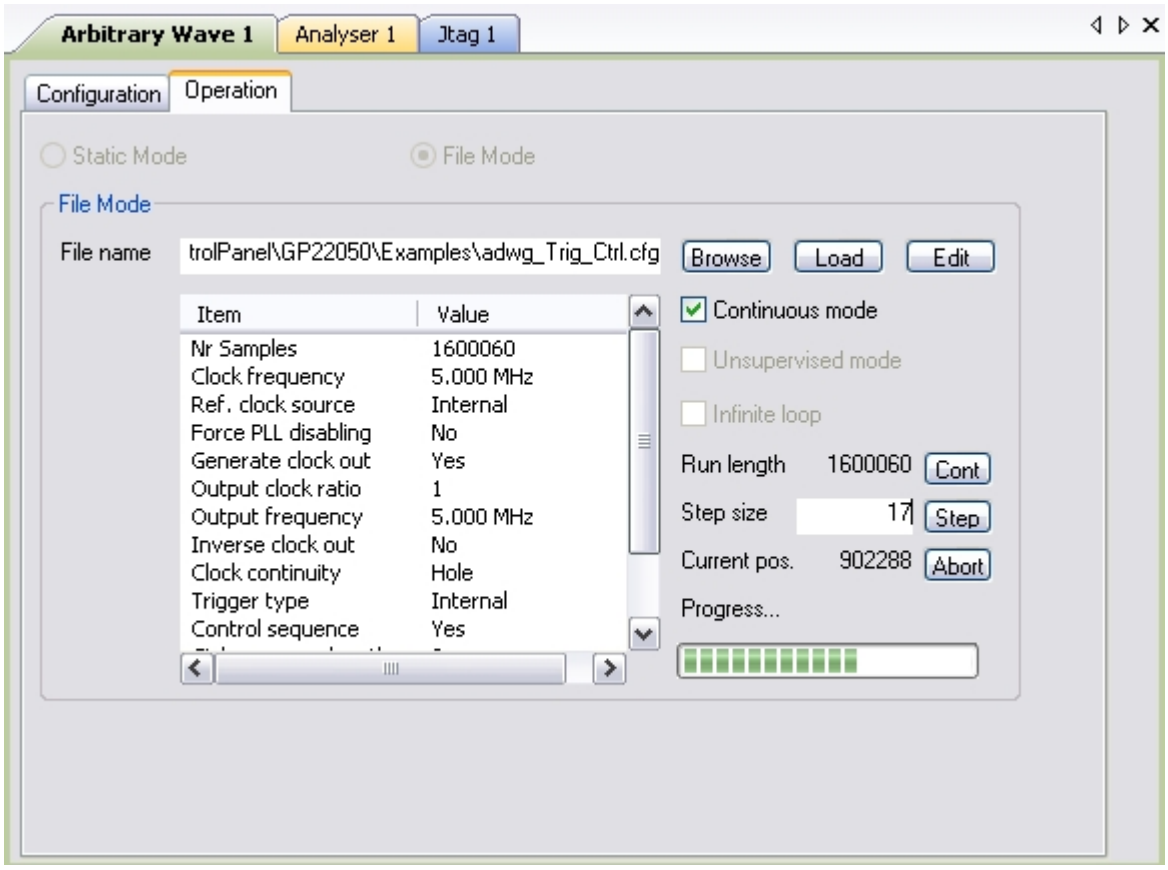
4.5.1 Features

The file mode operation consists in applying a sequence of data stored in a file to the system connector. The file contains the sequence of data to send out and also the periodic sequence that can be applied on the control lines. It can also contain the device configuration. The user just has to load the file and then execute the sequence. The sequence can be executed in one burst or a step by step mode can be used.
The main features of this mode are:

- A maximum of 16 output data lines can be used to generate up to 100 MSample/100 MByte data depth per run (refer to Table 4).
- A maximum of 6 output control lines are available. One line can be reserved for generating the output clock. Another one can be reserved to operate on an external clock provided by the user.
- External triggering can be used to start the sequence. If not used the sequence is started upon user request (*Run* or *Step* button).
- The control lines can be used to apply static default levels or a periodic sequence. The waveform of the control sequence has to be defined in the data file.
- The operating clock can be defined from 800 Hz the the device maximum frequency (50 MHz for GP-2205 and Xpress devices; 100 MHz for GP-24100 device) and the reference clock can be internal or external.
- A continuous output clock can be generated with a ratio between 1 and 65535 with respect to the defined operating clock.
- In hole clock operating mode, a single clock edge is generated each time a data is applied on the system connector. In that case, the ratio must be equal to 1. In continuous clock mode, the output clock is permanently applied on the connector pin.
- When working with a ratio equal to 1, the output clock rising edge can be generated in phase with the output data or half a period of the operating clock after the output data. When the ratio differs from 1, the phase relation between the data and the output clock is unpredictable.
- Single burst or step-by-step running mode
- A loop mode is available to load a sequence in the device's internal memory and repeat it infinitely (until the *Stop* button is pressed). In this case, the maximum sequence length is limited to the device internal memory (e.g.: 16 kB for GP-22050, 8 MB for GP-24100 – please check the data sheet of your device).

4.5.2 Interface Description

Figure 12: File mode operation for arbitrary wave application



FILE NAME

Field used to define the path and name of the user data file. The user can directly edit its content or use the browse button to select the file.

BROWSE

A browsing dialog window is opened when clicked. The user can browse its directory structure to select the data file. When the file is selected, the *File name* field is automatically updated and the user is prompt to load the configuration and data.

LOAD

Used to load or reload the file defined in the *File name* field. If no file is selected, a browse dialog window is automatically opened. When a file is loaded, the configuration of the device is updated if configuration information is stored in the file. The settings defined in the file overrule the current settings. The control and data sequence are loaded in memory. The application is then ready to start applying the waveform on the system connector.

EDIT

The *Edit* button is used to open the file specified in the *File name* field in a text editor. The user can then modify the file content (data, control and/or configuration) and save it. When the editor is closed, the file is reloaded if requested.
If the *File name* field is blank, a template can be opened to create a new file.

CONFIGURATION SUMMARY

The *configuration summary* window provides a list with all the settings of the device and their current value. When the configuration is changed in the configuration tab, or by loading a new file, the summary is automatically updated.

CONTINUOUS MODE

☑ When checked, the continuous mode is enabled. When requesting continuous mode, as soon as a buffer underrun is detected in the device while data is transferred to the system connector, an error message is returned and the transfer is aborted. In this mode, the transfer has to be continuous otherwise an error is reported. A buffer underrun is due to an insufficient bandwidth to complete the transfer. To avoid this problem, the operating frequency has to be reduced. Another solution is to work in *unsupervised mode*

☐ When unchecked, the continuous mode is disabled. When a buffer underrun occurs, the transfer is paused until enough data is stored in the device buffer. Then the transfer is automatically restarted. In this mode, the transfer always completes without error, but the data stream is stopped each time a buffer underrun is detected. The transfer is then performed in several bursts.

ⓘ *When the transfer is paused in non-continuous mode, the output clock is also paused if it is configured in hole clock mode. In this mode, a clock pulse is only generated when a data is sent out.*

UNSUPERVISED MODE

☑ When checked, the unsupervised mode is enabled. This mode is used to reduce the traffic on USB bus to perform flow control. The job is started and no flow control is performed until all data have been sent to the device. At the end of the job, the transfer status is checked and an error message is reported if necessary. In this mode, the full USB bandwidth is reserved for data transfer.

☐ When unchecked, the unsupervised mode is disabled. During a data transfer, the status of the device is read on a regular basis to detect buffer underrun or other errors. This mode presents the advantage to stop a transfer as soon as an error is detected. But by reading the device status, a part of the USB bandwidth is used for the flow control, reducing the bandwidth available for the data transfer.

INFINITE LOOP

☑ When checked, the infinite loop mode is enabled. In this mode, the defined data pattern is stored in the internal device memory and sent out in a loop, as long as the *Stop* or *Abort* button is not pressed. In this case, the maximum data pattern size is limited to 16KB to fit the internal device memory. In this mode, as all the data is stored in advance in the device, no underrun can occur. In case of an external trigger and if the auto-ream feature is enabled, the infinite loop is automatically enabled.

☐ When unchecked, the standard "on-the-fly" operating mode is used. The data pattern read from the file is transferred to the device and applied on the system connector. The pattern length is not limited by the device internal memory size as data words are transferred continuously from the host PC. In this mode, if the bandwidth is not high enough, underrun can occur.

RUNLENGTH

This field contains the amount of data words to transfer to the device. This number can be smaller, equal or greater than the amount of words provided in the data file.
If the run length N is smaller than the file size, then only the first N words of the files are sent to the device.

If the run length N is equal to the file size, than all data words are sent to the device.
If the run length N is greater than the file size, then the data are automatically repeated. All the data of the file are transferred to the device. When the system arrives at the end of the file and data words still have to be sent out, it loops back to the file begin and starts over. This is repeated until the requested amount of data words are sent out.

START/CONT/STOP

The same button has different functionality depending of the status of the transfer.

- Before starting a job, the device is in idle mode waiting to start a transfer. The user can then click the *Start* button to start the job. The transfer of the N data words specified in the *runlength* field is then started. If external trigger mode is enabled, data will be applied on the connector 3 clock cycles after the detection of the trigger pattern.
- While the transfer is progressing, the button is functionality is changed to *Stop*. When clicked, the transfer is suspended.
- When the transfer is stopped, the button functionality is changed to *Cont*. When clicked, the transfer is continued. It can then be stopped again.

In supervised mode, as long as the N words of the *runlength* are not completely transferred to the device, the job can be stopped and continued without restriction. When operating in unsupervised mode, the job can not be interrupted, it can only be abotered.

STEP SIZE

This field is used to specify the number of data words to send to the device per step. The size can be defined between 1 and the *runlength* value.

STEP

This button is used to send *Step Size* data words in a burst. When *Step* is pressed with the step size is fixed to S, the S words following the current position in the file are sent out. The next time *Step* is pressed, the S following words are transferred. The *Step* button can be used to start a job, or to continuer a suspended transfer. If the external trigger mode is enabled and the *Step* button is pressed, no data will been sent out as long as the trigger pattern is not detected. The trigger is only used to initiate the transfer. The following steps are executed as soon as the user requests it.

> *Stepping is not supported when using the basic functions provided for the TCL scripting. This can be easily implemented using a loop and applying data to the GP Series device in static mode.*

ABORT

This button is used to terminate a job before all data have been applied on the connector. All application's and device's buffers are cleared. The device is reset in idle mode waiting for a new transfer to be started.

CURRENT POS

This read only field displays the current position in the file during the transfer and when the transfer is suspended or completed. The current position is also displayed using a progress bar. The current position is estimated by software during a transfer to reduce USB bandwidth snooping. When a transfer is stopped or when stepping mode is used, the exact current position is retrieved from the device.

4.6 File Format

A total of 3 different files are handled by the 8PI Control Panel application.

- Configuration file Text file containing the device configuration (control lines usage, clock frequencies and setting, etc.)
- Configuration & Data file Text file containing device configuration and data words. The format of this file is compatible with the previous one.
- Data sample file File containing only data words. Different formats are supported (binary or text). These file have to be included in a configuration file to be used by the control panel.

4.6.1 Configuration & data file

All settings of the *configuration tab* can be saved in a text file. A set of commands is defined to be able to encode the device configuration and data/control sequences. The file decoding is based on these commands. The following rules have to be respected.

- Leading and tailing spaces or tabs are ignored
- # is used as comment character. All characters following # are ignored
- A command can not be split over several lines
- Commands must be lower case

The following list gives all the commands that can be used to configure the device for an ADWG job.

@transfer continuous <true/false>

When set on 'true', this parameters forces the device to send a new sample at each clock cycle (usual type of tranfer for a pattern generator). When set on 'false', the system relaxes this constraint and allows outputting the sample when they are available in the device.

@transfer infinite <true/false>

This parameter must be set to 'true' when using the ADWG in infinite loop mode, to repeat the same set of data indefinitely.

@trigger type <internal/external>

This command defines if an *internal* or *external* trigger must be used.

@trigger autorearm enable

This comman enables / disables the trigger 'auto rearm' feature.

@trigger sensitivity <level/edge>

Defines whether the device should detect a level transition on the trigger or a level.

@trigger ctrl <pattern> <mask>

If external triggering is selected, this command defines the trigger *pattern* to detect on the control lines to generate the trigger event. The *mask* defines which control lines have to be considered for the trigger pattern detection.

If the internal trigger mode is selected, this command is ignored.

The user must take care not to use the same control line for triggering and control sequence definition. Each control line can only have a single functionality: trigger or control sequence. If a conflict is created, the trigger has the priority.

@freq <freq_in_Hz>

Defines the requested operating frequency. The frequency must be a decimal value representing the frequency expressed in Hz. The value must be defined between 50MHz and 800Hz (between 100 MHz and 800 Hz for GP-24100 device)

@clock source <internal/external>

This command selects the clock source. It can be *internal* or *external*.

@clock pll <enable/disable>

When used with the *enable* parameter, the device internal PLL is enabled if the requested frequency is higher than 24 MHz. For lower frequencies, the PLL is automatically disabled.
When used with the *disable* parameter, the device internal PLL is disabled, independently of the clock frequency.

@clock output <enable/disable>

To *enable* or *disable* the generation of the output clock. For device hardware revision 01, when this is enabled, control line 5 (pin D21) is reserved and configured as output to provide the generated clock. For device hardware revisions 02 and higher, the dedicated CkOut pin is used.

@clock outputratio <int_ratio>

Defines the clock ratio used to generate the output clock. The operating clock is divided by the integer ratio value. The ratio must be coded as a decimal number. Its value must be defined between 1 and 65535.
This command is ignored if the output clock is not enabled.

@clock type <continuous/hole>

This command defines the output clock continuity. When *continuous* is selected, a permanent clock is driven. When *hole* is selected, a not continuous clock is generated. In this mode, a single clock pulse is generated each time a data is sent out. If no data is sent out, the clock remains low.
This command is ignored if the output clock is not enabled.

@clock polarity <positive/negative>

This command defines the phase relation between the output clock and the data. If the *positive* polarity is selected, the rising edge of the output clock is in phase with the data. If the *negative* polarity is selected, the falling edge of the output clock is in phase with the data.
This command is ignored if the output clock is not enabled.



The clock polarity can only be used if the ratio is equal to 1. If the ratio differs from 1, the clock phase with respect to the data is variable.

The following list gives all the commands available to define the control and data sequences.

@ctrl_begin

Define the beginning of the control section. All command lines placed between *@ctrl_begin* and *@ctrl_end* are used to define the periodic sequence to apply on the control lines.

@ctrl_end

Define the end of the control section.

@header <xxxxxx>

This command defines the control mask. The parameter is composed of a 6 bits binary coded word. Each bit of the parameter corresponds to a control line. The most left bit is assigned to control line 5 (pin D21) and the most right bit is assigned to control line 0 (pin D16). When a bit is set to "O" (upper case letter), the corresponding control line is configured as output and can not be used for the trigger. When a bit is set to "-", the corresponding control line is not used as output. If the control line is also not used for the triggering, then it remains disabled (Hi-Z). Any other characters are not allowed in ADWG mode.

This command can only be used within the control section. If used in the data section, the parameter must be 16 bits long.
If several *@header* commands are used, only the last one is taken into account

@default <xxxxxx>

This command defines the control default value. The parameter is composed of a 6 bit binary coded word. Each bit of the parameter corresponds to a control line. The most left bit is assigned to control line 5 (pin D21) and the most right bit is assigned to control line 0 (pin D16). When a bit is set to 1, the corresponding default level is set to 1, otherwise a 0 level is defined. Only the default level of control lines configured as output with the *@header* command will be taken into account.
This command can only be used within the control section.
If this command is used, the *@ctrl* command can not be used.
If several *@default* commands are used, only the last one is taken into account

As soon as the @default command is used, the control sequence with the use of default value is enabled. No specific command is defined to enable/disable the sequence. If no @default command is used, the control sequence remains disabled.

@ctrl <xxxxxx>

This command is used to define the periodic sequence that can be applied on the control lines. The parameter is composed of a 6 bits binary coded word. Each bit of the parameter corresponds to a control line. The most left bit is assigned to control line 5 (pin D21) and the most right bit is assigned to control line 0 (pin D16).
Each time a *@ctrl* command is used, the pattern defined by the parameter is added to the sequence to apply on the control lines. A maximum of 250 *@ctrl* commands can be used in a file. This limits the maximum length of the control sequence to 250 clock cycles.
This command can only be used within the control section.
If this command is used, the *@default* command can not be used.
This command can be combined with the *@loop* command.

As soon as the @ctrl command is used, the control sequence is enabled. No specific command is defined to enable/disable the sequence. If no @ctrl command is used, the control sequence remains disabled.

@data_begin

Define the beginning of the data section. All command lines placed between *@data_begin* and *@data_end* are used to define the waveform to apply on the data lines.

@data_end

Define the end of the data section.

@header <xxxxxxxxxxxxxxxx>

This command defines the data mask. The parameter is composed of a 16 bits binary coded word. Each bit of the parameter corresponds to a data line. The most left bit is assigned to data line 15 (pin D15) and the most right bit is assigned to data line 0 (pin D0). When a bit is set to "O" (upper case letter), the corresponding data line is configured as output. When a bit is set to "-", the corresponding data line is disabled (Hi-Z). Any other characters are not allowed in ADWG mode.
This command can only be used within the data section. If used in the control section, the parameter must be 6 bits long.
If several *@header* commands are used, only the last one is taken into account

@data out <xxxxxxxxxxxxxxxx>

This command is used to define the waveform to apply on the data lines. The parameter is composed of a 16 bits binary coded word. Each bit of the parameter corresponds to a data line. The most left bit is assigned to data line 15 (pin D15) and the most right bit is assigned to data line 0 (pin D0).
Each time a *@data* command is used, the pattern defined by the parameter is added to the waveform to apply on the data lines. The maximum length of the data waveform is set to 10M words.
This command can only be used within the data section.
This command can be combined with the *@loop* command.

@loop <line_cnt> <repeat_cnt>

This command gives the possibility to repeat *line_cnt* lines *repeat_cnt* times. This command can be used in data or control section. The lines that can be repeated have to be *@data* or *@ctrl* commands. Any other command can not be repeated with this command.

@loop_begin <repeat_cnt>

Define the beginning of a loop section. The sequence defined by the commands placed between *@loop_begin* and *@loop_end* is repeated *repeat_cnt* times. The only commands that can be placed in a loop section are *@ctrl*, *@data* and *@file*. It is not allowed to place a *@loop* command within a loop section.

@loop_end

Define the end of the loop section.

@file <type> <path_name>

The *@file* command gives to possibility to include a *data sample* file into a configuration and data file. The content of the included file is read and each line is handled as it was the pattern of a *@data* command. The values read from the file are added to the data waveform.
Different types are supported: txt_rawbin, txt_rawhex and bin. The two first are text formatted files, the third one is a binary encode file. Refer to section "4.6.2 Data sample file" for more details concerning these files.
If a *@file* command is placed in a loop section, then the file is read *repeat_cnt* times.

Figure 13: Example of configuration & data file

```
#
# ADWG w/ trigger and w/ ctrl sequence
#

# Reference clock & output clock settings
@clock source internal
@freq 1000000
@clock pll enable
@clock output enable
@clock outputratio 1
@clock type hole
@clock polarity positive

# Transfer parameters
@transfer continuous false
@transfer unsupervised true

# Define the trigger type, mask and pattern
@trigger type external
@trigger ctrl 0x01 0x01

# --------------------------------
#  Begin of control section
#
@ctrl_begin

#  pin      543210
   @header  --OOO-

   @ctrl    001110
   @ctrl    001100
   @ctrl    001000
   @loop 1 3
   @ctrl    000000
   @ctrl    001000
   @ctrl    001110
@ctrl_end
#
#  End of control section
# --------------------------------
```

```
# --------------------------------
#  Begin of data section
#
@data_begin

#  pin          FEDCBA9876543210
   @header      --------OOOOOOOO

   @loop_begin 20000
      @file txt_rawbin ".\datafile_rawbin.dat"
   @loop_end
   @file txt_rawhex ".\datafile_rawhex.dat"
   @file bin        ".\datafile_bin.bin"

   @data out    0000000000000001
   @data out    0000000000000000
   @data out    0000000000000101
   @data out    0000000000001000
   @data out    0000000000010001
   @data out    0000000000100000
   @data out    0000000001000001
   @data out    0000000010000000

# The following three lines are repeated 5 times
   @loop 2 5
   @data out    0000000001010101
   @data out    0000000010101000

# The following section is repeated 3 times
   @loop_begin 3
   @data out    0000000000000001
   @data out    0000000000000000
   @data out    0000000000000101
   @data out    0000000000001100
   @data out    0000000000011101
   @data out    0000000000111100
   @data out    0000000001111101
   @data out    0000000011111100
   @loop_end

   @data out    0000000000000011

@data_end
#
#  End of data section
# --------------------------------
```

4.6.2 Data sample file

A data sample files only contain 16 bits words data values.

- No command is allowed in this file
- Comments can be defined using character #
- Blank lines are skipped
- Leading and tailing spaces or tabs are ignored

The content of the file is read and appended to the waveform data buffer defined in the *Configuration & Data* file. Three different sample file types are supported.

RAW BINARY TEXT FILE

This file is a text formatted file. Each line contains a 16 bits binary encoded value providing the data pattern to apply on the output connector. Each bit of the binary value corresponds to a data line. The most left bit is assigned to data line 15 (pin D15) and the most right bit is assigned to data line 0 (pin D0).

In this mode, each line must contain 16 bits, even if less bits are used on the system connector.

Figure 14: Example of raw binary text file

```
# raw binary file format

0000000000000001
0000000000000000
0000000000000101
0000000000001000
0000000000010001
0000000000100000
0000000001000001
0000000010000000
```

RAW HEXADECIMAL TEXT FILE

This file is a text formatted file. Each line contains a 16 bits hexadecimal value providing the data pattern to apply on the output connector. The LSB of the hexadecimal correspond to data line 0 (pin D0). The "0x" prefix is supported but is not mandatory. If the hexadecimal value is coded with less than 16 bits, then the missing MSB are automatically set to 0.

Figure 15: Example of raw hexadecimal text file

```
# raw binary file format

0x0001
0x00
0005
0x0008
0x0011
0x0020
0x0041
0x0080
```

BINARY FILE

This file is a binary encoded file. No comments can be placed in this file. The first byte in the binary file is associated to the least significant byte of the data word and the second byte to the most significant byte. In the example of Figure 16, the first two bytes 01 and 00 are defining the 0x0001 pattern for the data lines. The next two bytes 0A and 0B are defining the next pattern 0x0B0A, and so on.

Figure 16: Example of binary file

```
 0 1 2 3             Hexadecimal 16-bits

01000A0B               0x0001  0x0B0A
05000800               0x0005  0x0008
11002000               0x0011  0x0020
41008000               0x0041  0x0080
```

5 State Analyser (logic analyser)

Applies to GP Series devices.

5.1 Features

This application is dedicated to the GP Series device. It provides to the user a graphical interface to configure and control the device in *State Analyser* mode. It gives the possibility to capture data from the device system connector. This page provides a very fast and easy way to capture and analyse data applied on the system connector.
The following list describes the main features that can be controlled through the graphical interface.

- Single device support. Even if several *Analyser* pages can be opened simultaneously, they all access the same device. Only one of the pages can be used at a time
- 16 bits output data lines available for sampling up to 100 MByte / 100 MSample data per run.
- 6 bits control lines available to apply default static levels.
- Up to 6 bits external trigger pattern definition
- Internal or external reference clock operation
- Output clock generation with a programmable ratio with respect to the operating clock
- Clock polarity control
- Configuration save/restore
- Real time monitor mode to displaying the current level applied on the 16 data lines
- File based operating mode
- Support of different file formats (text, binary and VCD) to save captured data
- Wave viewer to analyse the captured signals

Only the 16 data lines can be sampled and stored on the host computer. The 6 control lines can be used as output to generate static levels or as input to be used as external trigger. The level applied on the 6 control lines can't be sampled and transferred to the host computer.

Almost all these features are also accessible through the provided C/C++ DLL and TCL library. Refer to [4] and [5] for more details on how to control the GP Series device ADWG mode, using respectively the C/C++ library and the TCL scripting facilities.

5.2 Analyser Application Page

The *Analyser* application page can be opened using the File > New application menu and then by selecting *Analyser* in the dialog. The page can also be opened using the *New Page* pane. If the pane is not visible go to menu View > New Page Pane.

The *Analyser* page is composed of two tabs (refer to Figure 17): a first tab used to define the device configuration, and the other to control the data capture process.

5.3 Device Configuration

5.3.1 Control Lines Configuration & Mapping

This group of settings is used to configure the operating mode of the 6 control lines, to define the sampling clock and to select the data lines to be captured. For more details on the system connector refer to [1] and to section "Appendix A".

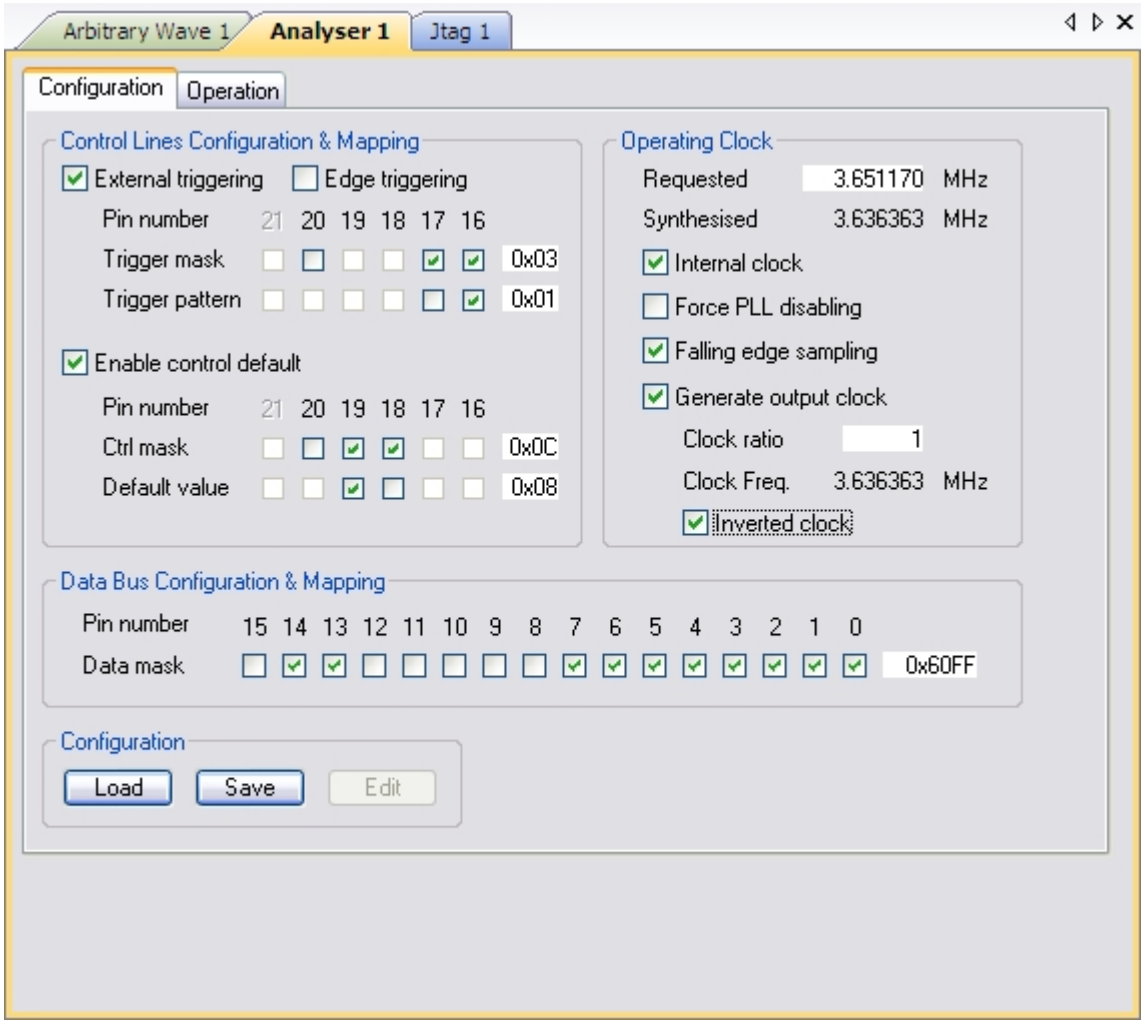
In analyser mode, the level applied on the 6 control lines can not be captured and stored on the host computer. But they can be used for different functionalities:
When operating in output mode, they can be used to:

- Drive a static default 0 or 1 logic level
- Generate a clock signal related to the system clock

When operating as input signals, they can be used:

- As external triggering lines to start the data capture
- As input for an external reference clock

Figure 17: Analyser wave configuration tab



ENABLE EXTERNAL TRIGGERING

☑ When checked, it enables the use of an external trigger to start the data capture. At least one control line has to be selected to operate as an input and be used as trigger event. Several lines can be used to define the trigger event. When the defined pattern on the select control lines is detected, the trigger event is generated and the analyser is started. When the external trigger is enabled, the start latency is equal to 5 clock cycles.

☐ When not checked, no external trigger event is used and the analyser capture is started as soon as requested by the user.

EDGE TRIGGERING

☑ When checked, it selects the 'edge triggering mode'. In this mode, the Analyser run is triggered when the selected trigger lines transit to the programmed trigger pattern.

☐ When not checked, the 'level triggering mode' is selected. In this mode, the Analyser run is triggered when the programmed trigger pattern is present on the selected trigger lines.

TRIGGER MASK

☑ When selected, the corresponding control line is selected to be used as input for the external triggering. The control line can not be used to drive a default logic level. It remains operating as an input for the complete capture sequence.

☐ When not selected, the pin is not reserved for external triggering and is operating as an output.

TRIGGER PATTERN

The *trigger pattern* can only be defined if at least one control line is selected as input in the external *trigger mask*. The *trigger pattern* defines the logic levels that must be detected on the input control lines to generate the trigger event. The level applied on the control lines not selected for the trigger is ignored while detecting the trigger pattern.

☑ When selected, logic 1 is associated to the control line.

☐ When not selected, logic 0 is associated to the control line.

ⓘ *Instead of clicking on the different check box of the trigger mask and pattern, the expected values can be directly entered in hexadecimal format with prefix 0x in the corresponding field.*

ENABLE CONTROL DEFAULT

When configured as output, the control lines can be used to drive static levels.

☑ When checked, it enables the used of the control lines to drive static logic levels. At least one control line has to be selected in the *control mask*.

☐ When unchecked, no control line is driving any output signal. All lines not used in the *trigger mask* are set in Hi-Z.

CONTROL MASK

☑ When selected, the corresponding control line is configured as an output and can be used to drive the static level. Only the control lines not used for the external trigger can be used. They remain configured as an output during the complete analyser job.

☐ When not selected, the corresponding control line is set in Hi-Z if it is not used in the external *trigger mask*.

DEFAULT VALUE

The *default value* can only be defined if the *enable control default* is activated, and if at least one control line is selected as output in the *control mask*. The *default value* defines the static logic levels that must be applied on the selected output lines.

☑ A default logic 1 is defined when checked

☐ A default logic 0 is defined when unchecked

ⓘ *Instead of clicking on the different check box of the control mask and default value, the expected values can be directly entered in hexadecimal format with prefix 0x in the corresponding field.*

5.3.2 Operating Clock

The operating clock can be internally generated or provided by the user as an external reference clock.
When the clock is internally generated, it is derived from a 200 MHz internal reference clock. The operating clock is generated using a divider. Only integer ratio can be programmed and so not all frequencies can be achieved exactly. The accuracy of the clock divider is 4ns. The achieved frequency is displayed under the requested one.
When the external clock reference is used, the operating clock is equal to the external clock. No divider is used in this case. In external clock mode, the requested frequency field must be filled with the frequency of that clock. This is mandatory to ensure a correct configuration of the device PLL (refer to *Force PLL disabling*).

REQUESTED OPERATING CLOCK

This field defines the frequency of the operating clock used in the device. The frequency must be defined in a range between the maximum frequency of your device (50 MHz for GP-22050 and 100 MHz for GP-24100) and 800Hz. The value entered in this field must be a decimal value having a maximum of 6 decimal digits and representing the frequency in MHz.

INTERNAL CLOCK

☑ When checked, the 200 MHz internal reference clock is used to generate the operating clock. By default, the internal reference clock is selected.

☐ When not checked, an external reference clock must be provided by the user. For device hardware revision 01, the clock has to be applied on pin D20 of the system connector. In this case, when the external clock is enabled, control line D20 is reserved as input for that clock. This control line can then not be used for external triggering and for applying static levels. For device hardware revisions 02 and higher, the dedicated CKIn pin is used, with no effect on the control lines availability.

FORCE PLL DISABLING

Under defined circumstances, the GP Series device can use internal PLL to reduce the delay between the operating clock and the output clock. The operating clock frequency must be in the range 'Max frequency' (50 MHz for GP-22050 and 100 MHZ for GP-24100) to 18 MHz to be able to use the PLL. The PLL can be used for both internal and external clock.
The user has the possibility to disable the PLL or to let the application turning it ON or OFF automatically in function of the requested operating frequency.

☑ The PLL is disabled and never used when checked.

☐ The PLL is automatically enabled in function of the operating frequency when unchecked.

ⓘ *The external clock must be stable, continuous and applied to the device before starting to use it.*

ⓘ *If the PLL is used, refer to [1] to get the timing and jitter specifications of the external clock.*

FALLING EDGE SAMPLING

☑ When checked, the data are sampled on the falling edge of the operating clock.

☐ When not checked, the data are sampled on the rising edge of the operating clock.

GENERATE OUTPUT CLOCK

☑ For device hardware revision 01, when checked, clock signal is generated on output pin D21 (hardware revision 01) of the system connector. When the output clock is enabled, the control line corresponding to pin D21 is reserved for this clock. The control line can neither be used for external triggering nor for applying default logic levels. In case the device hardware revision is 02 or higher, the dedicated CkOut pin is used, with no effect on the control lines availability.

☐ When not checked, no clock is sent out. In case the device hardware revision is 01, pin D21 and the corresponding control line can be used for external trigger or driving logic levels. In case the device hardware revision is 02 or higher, there is no effect on the control lines availability.

CLOCK RATIO

If the output clock is enabled, then its frequency is determined using an integer ratio that can be defined between 1 and 65535. When set to 1, the output clock is the image of the operating clock (internal or external in function of the selected operating mode). If the ratio differs from 1, the integer ratio value is used to divide the operating clock.

The ratio value must be defined as a decimal number.

The achieved frequency of the output clock is displayed under the requested ratio.

INVERTED CLOCK

This parameter is related to the output clock and does not affect the clock edge used to sample the data.

☑ When checked, the output clock is generated on the falling edge of the operating clock.

☐ When unchecked, the output clock is generated on the rising edge of the operating clock.

5.3.3 Data Bus Configuration & Mapping

The 16 pins (D0 to D15) of the system connector are reserved for user data. In the *Analyser* application, these lines can either be used as input to be sampled, either be disabled and set in Hi-Z. The user just has to click on the check box associated to a data input pin to enable it. An hexadecimal value with the prefix "0x" can also be directly entered in the field next to the check box.

5.3.4 Configuration

The user configuration can be saved to a file. This gives the possibility to retrieve a configuration by loading a saved configuration file. The configuration is saved in a text file. Refer to section "4.6 File Format" for more details concerning the file format.

LOAD

Load a configuration from a file.

SAVE

Save the current configuration to a file

EDIT

Open the loaded configuration file in a text editor to let the user modify it. When the text editor is closed, the user is prompt to reload the configuration.

5.4 Real Time Monitor Operation

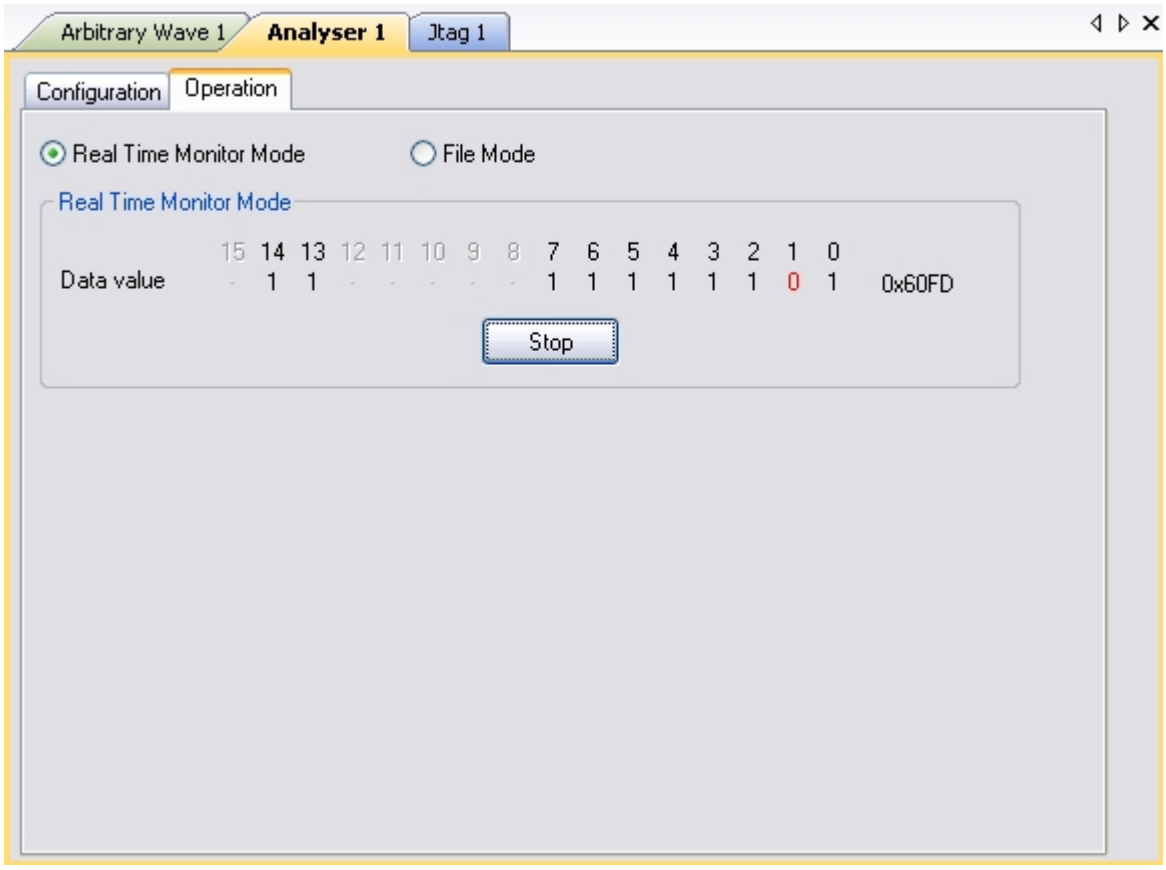
5.4.1 Features

The real time monitor is an application used to observe the logic levels applied on the 16 data lines. When activated, the system connector is sampled every 100ms. The logic level detected on the data lines enabled for the capture (refer to section 5.3.3) is then displayed.
The main features of this operating mode are:

- 16 input data lines available
- 6 control lines available to apply static levels. One can be reserved if the output clock is generated. The external triggering feature is disabled when operating in *Real Time Monitor* mode.
- A continuous output clock can be generated with a ratio between 1 and 65535 with respect to the defined operating clock.

5.4.2 Interface Description

Figure 18: Static operating mode for the arbitrary wave application



REAL TIME MONITOR MODE

☑ Enables the *real time monitor* operating mode when checked.

☐ This button is automatically cleared when the *File* Mode is selected.

FILE MODE

☑ Switches to *File* Mode when checked (refer to section "4.5 File Mode Operation".

☐ This button is automatically cleared when the *Real Time Monitor* Mode is selected.

ⓘ *It is not possible to switch between real time monitor and file mode while a job is running. All data transfers must be terminated before to switch mode.*

DATA VALUE

These 16 bits are used to display the logic levels detected on the system connector. Only the levels of the data lines enabled with the *Data Mask* in the *Configuration* tab are displayed. When a data line toggles, the displayed value is changed and its colour is turned to red for 1 second. If the data line remains stable for more than 1 second then its colour is set back to black.

On the right side of these 16 bits, the corresponding hexadecimal value is also displayed.

RUN / STOP

To start the *Real Time Monitor*, the *Run* button must be clicked. The data sampling is started and the button text is then changed to *Stop*. As long as the *Stop* button is not clicked, the data lines are sampled every 100ms.

When the monitor is running it is not possible to change operating mode or to go to the configuration tab. The monitor has to be stopped to be able to perform any other action.

The real time monitor operates taking into account all the settings defined on the configuration tab except the external triggering which is always disabled in *Real Time Monitor* mode. This means that it is possible to generate static levels on the control lines and an output clock while the monitor is running.

ⓘ *It is not possible to save to a file the data sampled from the system connector when operating in real time monitor mode.*

5.5 File Mode Operation

5.5.1 Features

The file mode operation consists in loading a configuration from a file to capture data from the system connector. The captured samples are then stored in memory or in a file and/or displayed in a wave viewer. Different file format are supported.
If no configuration file is loaded, the sampling is performed based on the settings defined in the *Configuration* tab.

The main features of this mode are:

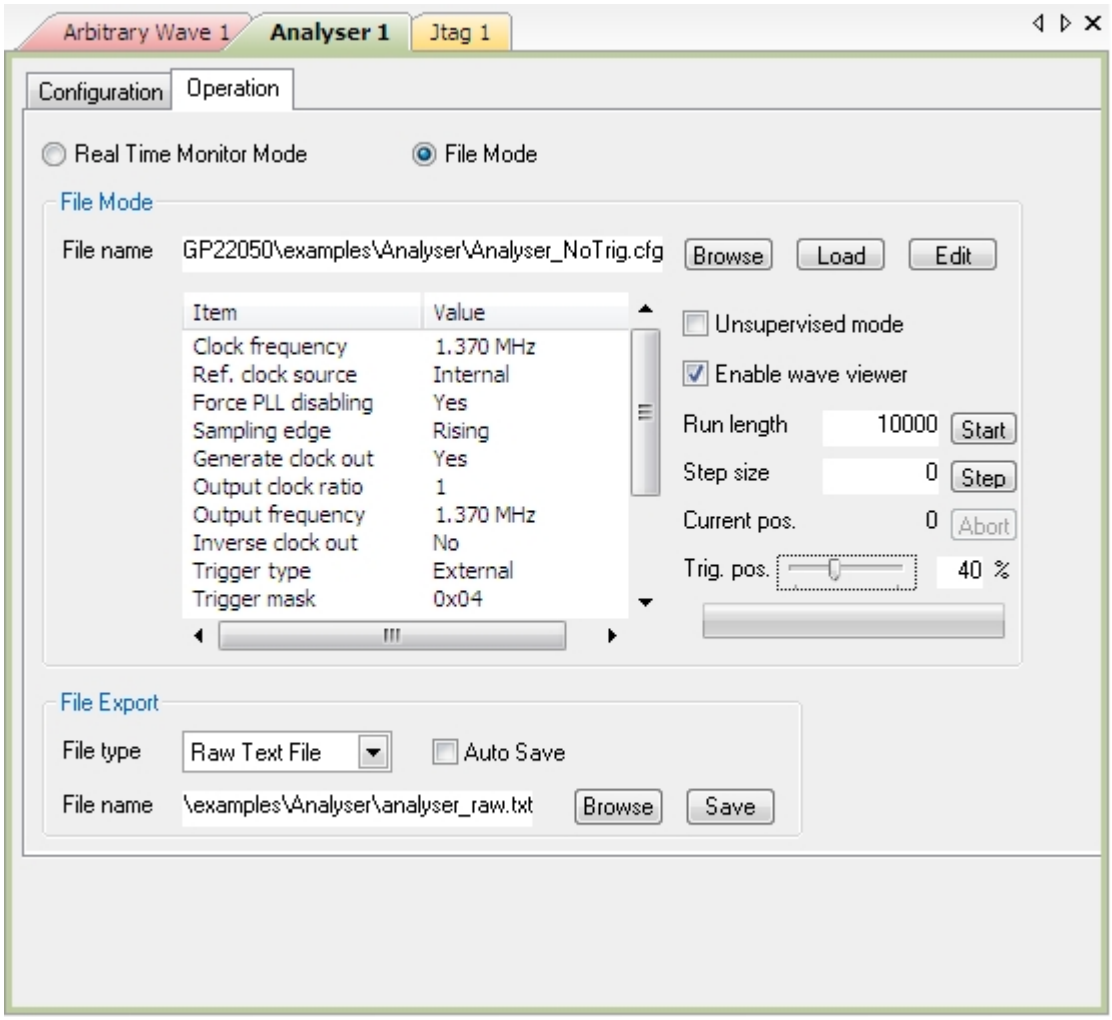
- A maximum of 16 output data lines can be sampled
- A maximum of 6 output control lines are available to drive static output pattern while capturing data.
- A maximum 100 MSample and 100 MByte data depth per run is allowed.
- One control line can be reserved to generate an output clock. Another one can be reserved to operate as an external reference clock provided by the user.
- External triggering can be used to start the sequence. If not used the sequence is started upon user request (*Run* button).
- When the internal reference is used, the sampling clock can be defined between the device maximum clock frequency (50 MHz for GP-22050 and 100 MHz for GP-24100) and 800 Hz. When a user provided reference clock is used, the frequency can be defined between 50MHz and DC.
- Selectable clock edge to sampled data from the system connector (rising or falling)
- A continuous output clock can be generated with a ratio between 1 and 65535 with respect to the defined operating clock.
- Single burst or step-by-step running mode
- Different output file format are supported: raw text, raw binary and VCD.
- Auto-save mode feature
- Integrated wave viewer

Table 5 : Analyser maximum data depth per run

Number of pins used on the connector	***Maximum data depth per run (Samples)***	***Maximum data depth per run (Byte)***
1 to 8	100 MSample	100 MByte
9 to 16	50 MSample	100 MByte

5.5.2 Interface Description

Figure 19: File mode operation for state analyser application



FILE NAME

Field used to define the path and name of the user data file. The user can directly edit its content or use the browse button to select the file.

BROWSE

A browsing dialog window is opened when clicked. The user can browse its directory structure to select the data file. When the file is selected, the *File name* field is automatically updated and the user is prompt to load the configuration.

LOAD

Used to load or reload the file defined in the *File name* field. If no file is selected, a browse dialog window is automatically opened. When a file is loaded, the configuration of the device is updated if configuration information is stored in the file. The settings defined in the file overrule the current settings. The application is then ready to start capturing data from the system connector.

EDIT

The *Edit* button is used to open the file specified in the *File name* field in a text editor. The user can then modify the file content (data, control and/or configuration) and save it. When the editor is closed, the file is reloaded if requested.
If the *File name* field is blank, a template can be opened to create a new file.

CONFIGURATION SUMMARY

The *configuration summary* window provides a list with all the settings of the device and their current value. When the configuration is changed in the configuration tab, or by loading a new file, the summary is automatically updated.

UNSUPERVISED MODE

☑ When checked, the unsupervised mode is enabled. This mode is used to reduce the traffic on USB bus to perform flow control. The job is started and no flow control is performed until all data have been retrieved from the device. At the end of the job, the transfer status is checked and a message is reported if an error has been detected. In this mode, the full USB bandwidth is reserved for data transfer.

☐ When unchecked, the unsupervised mode is disabled. During a data transfer, the status of the device is read on a regular basis to detect buffer overflow or other errors. This mode presents the advantage to stop a transfer as soon as an error is detected. But by reading the device status, a part of the USB bandwidth is used for the flow control, reducing the bandwidth available for the data transfer.

ⓘ *When a buffer overflow is detected the job is aborted. The data correctly captured before the error are saved to the specified file in auto-save mode or can be saved to a file by using the SAVE button.*

ENABLE WAVEVIEWER

☑ When checked, the GTK Wave waveviewer will start automatically and display the sampled data at the end of the run.

☐ When unchecked, the GTK Wave waveviewer won't start automatically at the end of the run.

RUNLENGTH

This field contains the amount of data words to capture.

START/CONT/STOP

The same button has different functionality depending of the status of the transfer.

- Before starting a job, the device is in idle mode waiting to start a transfer. The user can then click the *Start* button to start the job. The capture of the N data words specified in the *runlength* field is then started. If the external trigger mode is enabled, the data capture is started 5 clock cycles after the detection of the trigger pattern.
- While the transfer is progressing, the button is functionality is changed to *Stop*. When clicked, the transfer is suspended.
- When the transfer is stopped, the button functionality is changed to *Cont*. When clicked, the transfer is continued. It can then be stopped again.

When using the supervised mode, as long as the N words of the *runlength* are not completely retrieved from the device, the job can be stopped and continued without restriction. When operating in unsupervised mode, the job can not be suspended, it can only be aborted.

STEP SIZE

This field is used to specify the number of data words to capture per step. The size can be defined between 1 and the *runlength* value. Each time the step button is used, the specified amount of data is captured.

STEP

This button is used to capture *Step Size* data words in a burst. When *Step* is pressed with the step size is fixed to S, the S words are captured. The next time *Step* is pressed, S other samples are captured. The *Step* button can be used to start a job, or to continuer a suspended transfer.

> *Stepping is not supported when using the basic functions provided for the TCL scripting. This can be easily implemented using a loop and applying data to the GP Series device in static mode.*

ABORT

This button is used to terminate a job before all data have been captured from the connector. The data already captured are or can be saved to a file depending on the auto-save mode. The device is reset in idle mode waiting for a new transfer to be started.

CURRENT POS

This read only field displays the current position in the file during the transfer and when the transfer is suspended or completed. The current position is also displayed using a progress bar. The current position is estimated by software during a transfer to reduce USB bandwidth snooping. When a transfer is stopped or when stepping mode is used, the exact current position is retrieved from the device.

TRIGGER POS

When an external trigger is used, this control selects the relative position of the trigger within the full run. This feature is currently disabled on GP-24100 devices.

FILE TYPE

This drop down menu gives the possibility to select the format of the output file used to store the captured data. Refer to section 5.5.3 for more details concerning the supported format.

AUTO-SAVE

☑ When checked, the captured data are automatically stored in the selected output file.

☐ When unchecked, the data are captured and stored in a volatile buffer in the host computer memory. When the job is completed, the content of the memory buffer can be saved to a file by using the *Save* button. The same data buffer can be saved to different file name and format. The content of the data buffer is cleared only when a new job is started.

FILE NAME

It specifies the file name and path of the output file used to store the captured data. This name must be defined before to start a job if the auto-save mode is enabled. The user can edit its value directly in the field or use the *Browse* button.

BROWSE

A browsing dialog window is opened when clicked. The user can browse its directory structure to select the output file to be used to store the captured data.

SAVE

When clicked, the content of the captured data buffer is saved to the output file specified in the *File Name* field. A data buffer can be saved to different files. Saving the captured data buffer do not clear the buffer. The buffer is only cleared when a new job is started.

If an error occurs during the capture of data (e.g.: buffer overflow) the data already in the buffer are automatically saved is the Auto-save mode is activated, or it can be saved using the Save button.

5.5.3 Export File Format

Three file formats are supported to store the captured data to a file. Independently of the format, the captured data are always stored as 16-bits words. Each time a sample is captured a 16 bits word is stored in memory. All disable bits are set to 0. The other bits are set to the level detected on the system connector. Data line N on the system connector is represented by bit N in the word. Bit 0 is the LSB and is the most right bit in the word. Bit 15 is the MSB and is the most left bit in the word.
Independently of the selected file format, the captured data buffer can be directly observed using the wave viewer. Refer to "Appendix C: Wave Viewer" for more detail on how to start the wave viewer.

5.5.3.1 Raw Text

The raw text file format is a text file containing 16 bits per line.
The first line provides the bit index in the word for each column. The second line represents the data mask defined by the user when the file has been captured. "I" indicates the columns enabled for sampling. "-" indicates a disable bit and its column only contains 0.
In the example depicted on Figure 21, bits 15,14,13,12,5,4,2 and 0 are enabled. The other ones are disabled. The two firsts rows start with a "#" to indicate a comment and represent the header.

Figure 20: Example of Raw Text file

```
# FEDCBA9876543210
# IIII------II-I-I

  0001000000110101
  0010000000100001
  0011000000110001
  0100000000110101
  0101000000110001
  0110000000100101
  0111000000100001
```

5.5.3.2 Raw Binary

The raw binary file format contains 16 bits words representing the sampled data lines coded in binary format. The file is coded in little endian mode. Bit 0 of the first sampled word is place in the LSB of the first byte. The first byte of the word represents the least significant byte of the first sampled word. The second byte in the file represents the most significant byte of the first sampled word. Byte 3 is the LSByte of the second sampled word and so on.
The file does not contain any header.
In the example depicted on Figure 21, a counter has been sampled. The sampled words are 0000h, 0001h, 0002h, etc. The first line in the picture is not part of the file and is placed in the picture just to visualise the byte positions in the file.

Figure 21: Example of Raw Binary file

```
 0 1 2 3  4 5 6 7  8 9 A B  C D E F
00000100 02000300 04000500 06000700
08000900 0A000B00 0C000D00 0E000F00
10001100 12001300 14001500 16001700
18001900 1A001B00 1C001D00 1E001F00
20002100 22002300 24002500 26002700
```

5.5.3.3 VCD

The VCD file format is a standard text file format used to stored waveform. Only the changes of each line are stored in the file. This has the advantage that it can be opened directly with standard wave viewers provided with HDL simulators and so gives the possibility to view the waveform corresponding to the sampled data.

6 JTAG Protocol

Applies to GP Series devices.

6.1 Features

This application is delivered to use the GP Series device as a JTAG port controller. The graphical interface provides a fast and easy way to perform data transfer over a JTAG bus interface. In this particular mode, the GP Series device is configured to operate as a bi-directional device able to send and capture data.
A first low level protocol controller mode can be used to perform basic JTAG transfers, and a second operating mode is available to transfer SVF formatted files through the JTAG interface.
The following list describes the main features that can be controlled through the graphical interface.

- Easy JTAG chain configuration
- Programmable operating frequency
- Instruction and data register shift
- Direct control of the JTAG pins TDI, TDO, TMS, TCK and TRST
- SVF file support

All these features are also accessible through the provided C/C++ DLL and TCL library. Refer to [6], [7], [8] and [9] for more details on how to control a JTAG interface using the GP Series device C/C++ library and the Tcl scripting facilities.

6.2 JTAG Interface Signals

With the JTAG application, the GP Series device can control the 5 signals of a JTAG interface. The application controls the logic levels applied on the TMS, TCK, TDI and TRST signals and capture the level of the data coming from the device on TDO.
The position of these signals is fixed on the system connector. Their location is provided on Figure 22 and Table 6. This pin mapping has to be used for all JTAG applications.

Figure 22: Pin mapping of the GP Series device system connector

Vext	*0*	*1*	*Vext*
GND	**2**	**3**	GND
TMS - D0	**4**	**5**	D1 - **TDI**
TDO - D2	**6**	**7**	D3 - **TRST**
D4	**8**	**9**	D5
D6	**10**	**11**	D7
GND	**12**	**13**	GND
D8	**14**	**15**	D9
D10	**16**	**17**	D11
D12	**18**	**19**	D13
D14	**20**	**21**	D15
GND	**22**	**23**	GND
D16	**24**	**25**	D17
D18	**26**	**27**	D19
D20	**28**	**29**	D21 - **TCK**
SCL	**30**	**31**	SDA
GND	**32**	**33**	GND

Table 6: JTAG interface pin mapping on the system connector

JTAG Signals	Connector Signals	Connector Pin Number
TMS	D0	4
TDI	D1	5
TDO	D2	6
TRST	D3	7
TCK	D21	29

6.3 JTAG Low Level Protocol

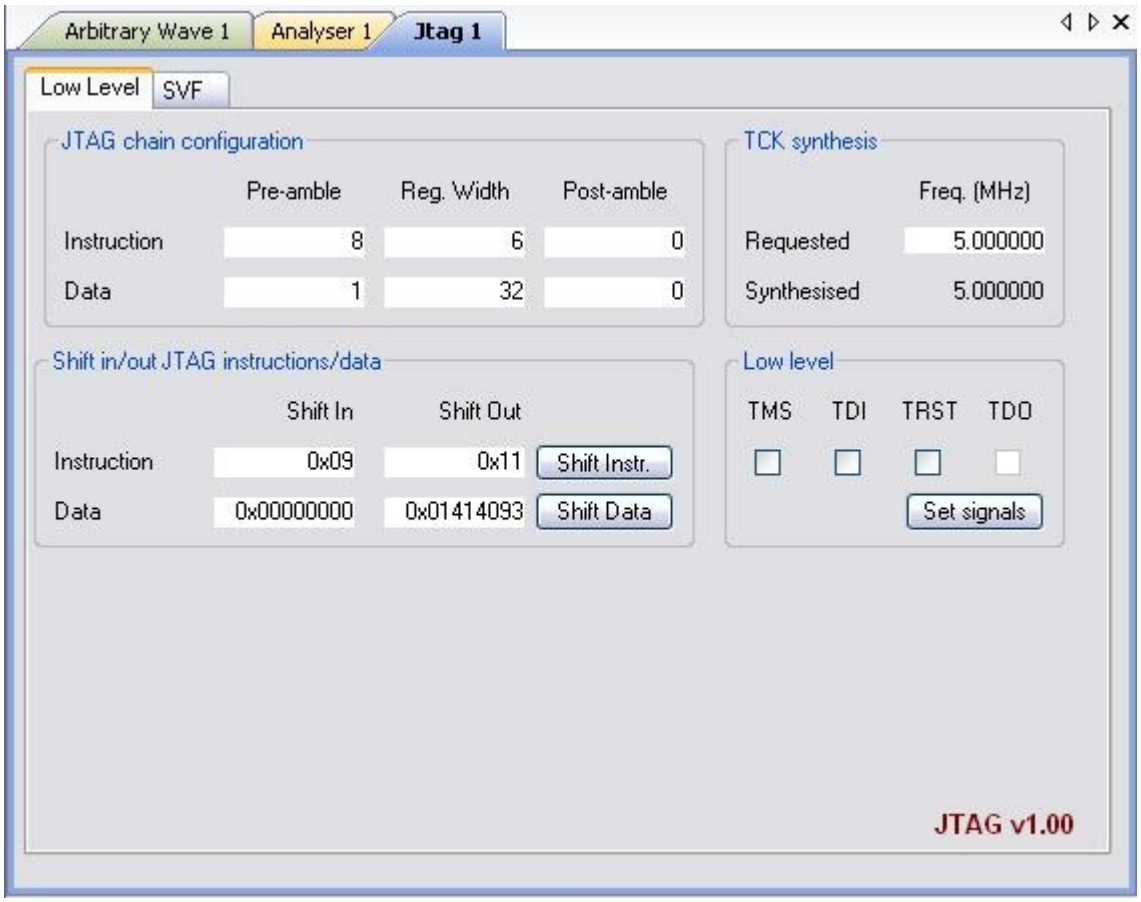
This page gives the possibility to control the JTAG protocol on a *Low Level* of abstraction. The user can directly control the logic level applied on the JTAG interface line or perform basic instruction and data register accesses.

The JTAG *Low Level* application page can be opened using the *File* > *New* application menu and then by selecting *JTAG* in the dialog. The page can also be opened using the *New Page* pane. If the pane is not visible go to menu *View* > *New Page* Pane.

The control of the logic level applied on the JTAG interface signals is very easy. The user just has to define the level of each signal by clicking on the check box associated to each signal and then apply the levels to the interface. The level of TDI, TCK, TMS and TRST can be defined. The level present on TDO is captured and displayed each time the apply button is clicked.

The data and instruction register shifting gives access to any device within a JTAG chain. Only one device can be accessed at a time. The device position in the chain is defined by configuring the pre-amble, post-amble and size of the data and instruction register of the target device. The user must then defined the data or instruction register value to shift in the device and click on the corresponding *Shift* button. Each time a *Shift* button is clicked the value shifted out of the device is captured and displayed.

Figure 23: JTAG low level control



The JTAG *Low Level* control page is divided in 4 different parts. The *JTAG Chain Configuration* gives the possibility to define the size of the target device instruction and data registers and their position in the chain.

INSTRUCTION PRE-AMBLE

Decimal value representing the number of bits in the scan chain following the instruction register.

INSTRUCTION WIDTH

Decimal value representing the number of bits in the instruction register.

INSTRUCTION POST-AMBLE

Decimal value representing the number of bits in the scan chain preceding the instruction register.

DATA PRE-AMBLE

Decimal value representing the number of bits in the scan chain following the data register.

DATA WIDTH

Decimal value representing the number of bits in the data register.

DATA POST-AMBLE

Decimal value representing the number of bits in the scan chain preceding the data register.

The *TCK Settings* defines the frequency of the JTAG clock. For more details related to the JTAG interface timing, refer to [1].

REQUESTED FREQUENCY

The JTAG clock frequency is generated by dividing the 200 MHz internal reference clock with an integer ratio. The requested frequency must be defined in MHz with a decimal value. The GP Series device can generate a clock between 50MHz and 800Hz.

ACHIEVED FREQUENCY

If the requested frequency can not be generated from 200MHz with an integer ratio, then the first achievable frequency lower than the requested one is generated. This field represents the frequency that will be used for the JTAG clock.

The *Register Shift* section gives the possibility to shift data and instruction register to or from the target device defined in the *Chain Configuration* section. To shift a data or instruction value to/from the target device, the corresponding *Shift* button must be used.

INSTRUCTION SHIFT IN

Hexadecimal value (prefixed with "0x") representing the instruction register value to shift in the target device.

INSTRUCTION SHIFT OUT

Hexadecimal value (prefixed with "0x") representing the instruction register value shifted out of the target device.

DATA SHIFT IN

Hexadecimal value (prefixed with "0x") representing the data register value to shift in the target device.

DATA SHIFT OUT

Hexadecimal value (prefixed with "0x") representing the data register value shifted out of the target device.

The *Low Level* section is used to directly apply a logic level on the signals of the JTAG interface. The requested level must be defined for all output signals (TMS, TDI and TRST) and then the *Set Signals* button must be clicked. Each time the button is used, the defined levels are applied on the output signals and a TCK clock pulse is automatically generated and the level present on the TDO pin is captured and displayed. The level of each signal is defined using the check box placed under each signal name.

☑ Defines a logic 1 on the corresponding signal.

☐ Defines a logic 0 on the corresponding signal.

6.4 SVF File Support

The *SVF* tab of the JTAG protocol application gives the possibility to execute files compatible with the *Serial Vector Format* to control a transfer to and/or from a JTAG chain. The SVF format gives the possibility to describe the complete sequence of transfers to perform on a JTAG chain with a reduce set of commands. The following commands are *supported*:

ENDDR	Specifies default end state for DR scan operations.
ENDIR	Specifies default end state for IR scan operations.
FREQUENCY	Specifies maximum test clock frequency for IEEE 1149.1 bus operations.
HDR	(Header Data Register) Specifies a header pattern that is prepended to the beginning of subsequent DR scan operations.
HIR	(Header Instruction Register) Specifies a header pattern that is prepended to the beginning of subsequent IR scan operations.
RUNTEST	Forces the IEEE 1149.1 bus to a run state for a specified number of clocks or a specified time period.
SDR	(Scan Data Register) Performs an IEEE 1149.1 Data Register scan.
SIR	(Scan Instruction Register) Performs an IEEE 1149.1 Instruction Register scan.
STATE	Forces the IEEE 1149.1 bus to a specified stable state.
TDR	(Trailer Data Register) Specifies a trailer pattern that is appended to the end of subsequent DR scan operations.
TIR	(Trailer Instruction Register) Specifies a trailer pattern that is appended to the end of subsequent IR scan operations.
TRST	(Test ReSeT) Controls the optional Test Reset line.

The complete SVF specification can easily be found on the Internet.

The following commands are *not supported*:

PIO	(Parallel Input/Output) Specifies a parallel test pattern.
PIOMAP	(Parallel Input/Output Map) Maps PIO column positions to a logical pin.

The following features are *not supported*:

HDR	The size of the data header can be defined, but not the pattern. "0" is used as default pattern.
HIR	The size of the instruction header can be defined, but not the pattern. "1" is used as default pattern to place the devices in bypass mode.
TDR	The size of the data trailer can be defined, but not the pattern. "0" is used as default pattern.
TIR	The size of the instruction trailer can be defined, but not the pattern. "1" is used as default pattern to place the devices in bypass mode
RUNTEST	Only the TCK clock is supported, not the SCK clock (system clock)

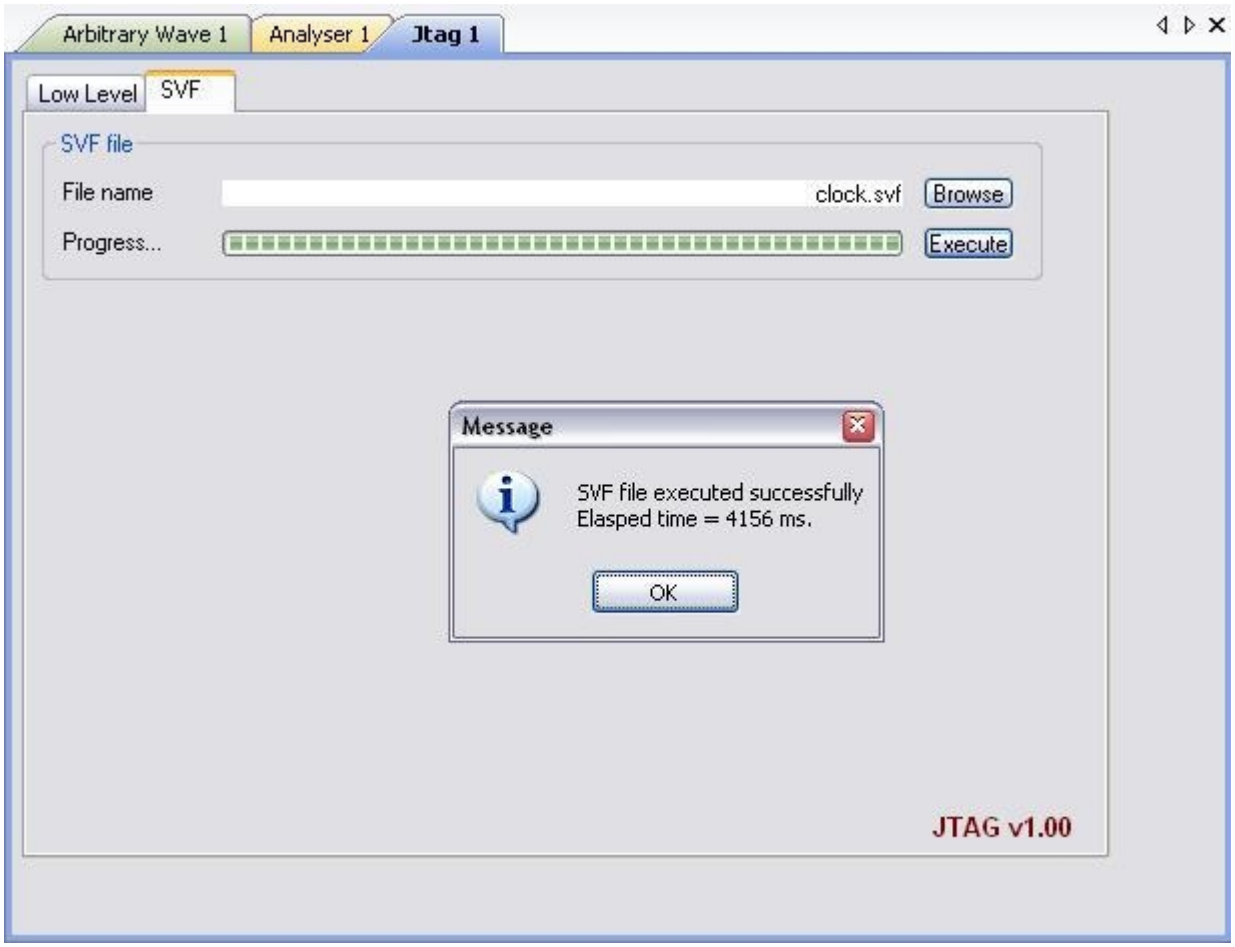
Figure 24 provides an example of SVF file. This example retrieves the IDCODE of an FPGA device. The JTAG controller of the device is reset and the tap controller is set to the IDLE mode.

Figure 24: Example of SVF file (read device IDCODE)

```
TRST OFF;
ENDIR IDLE;
ENDDR IDLE;
STATE RESET IDLE;
TIR 0 ;
HIR 0 ;
TDR 0 ;
HDR 0 ;
SIR 6 TDI (3f) SMASK (3f) TDO (01) MASK (03) ;
//Loading device with 'idcode' instruction.
SIR 6 TDI (09) ;
SDR 32 TDI (00000000) SMASK (ffffffff) TDO (f1414093) MASK (0fffffff) ;
```

The *SVF* application is very easy to use (refer to Figure 25). The file name and path must be defined and the *Execute* button must be used. The file is executed and the JTAG transfers are performed. A bar is displaying the progress of the execution. When the execution is completed, a message is displayed with the status and elapsed time.

Figure 25: JTAG SVF file support



FILE NAME

This field is used to define the name and path of the SVF file to execute. The user can type it directly in the field or used the *Browse* button to open a dialog to browse the directory structure to select the SVF file.

PROGRESS BAR

Shows the file execution progress

EXECUTE BUTTON

Start the execution of the SVF file defined in the *File Name* field.

7 SPI

Applies to GP Series devices and SPI Xpress device.

7.1 Features

This application is delivered to use the GP Series devices as a serial protocol master and serial protocol analyser. It is also the default application for the SPI Xpress device. The graphical interface provides a fast and easy way to configure and control the device. With the SPI mode of operation of the 8PI control Panel, you can use your GP Series or SPI Xpress device to control or analyse interfaces according to 'SPI-like' serial protocols. As a master, it can be used to write data to one or several SPI slaves and read data from them. As an analyser, it is used to sample, visualize and decode transactions from a SPI interface.
The following list describes the SPI mode graphical interface main features:

- Selection of clock frequency and characteristics
- Selection of the SPI mode (MOSI toggling edge, MISO sampling edge, clock phase and clock polarity)
- Selection of the number of SPI slaves
- SPI Master type selection: 4-wires SPI (full-duplex) or 3-wires SPI with bidirectional data line (half-duplex), with optional WE line for information on data line direction
- Data bit ordering selection
- Selectable WR/RD length for 4-wires SPI Master
- Separately selectable WR, WR-to-RD latency and RD length for 3-wires SPI Master
- Selectable control signals polarity
- Optional SSn start and stop edge positioning, with a resolution of 1/4th of the SPI clock period
- Script replay, log file recording
- Selectable external / internal triggering onto 6 control lines, with trigger positioning for SPI Analyser
- Automatic display into GTK Wave waveviewer
- SPI analyser at sample, transition or decoded transaction level
- SPI4 / SPI3 protocol types decoding for SPI Analyser
- SPI Analyser / SPI Master compatible export and source file format for SPI recorder/player implementation.

These features are also accessible with the provided C/C++ DLL and TCL library. Refer to [12] and [13] for more details on how to interface SPI ports using respectively the C/C++ library and the Tcl scripting environment.

From the *8PI Control Panel* interface, the *SPI mode of operation* can be used as a 4-wires interface (SPI4) and as a 3-wires interface (SPI3).

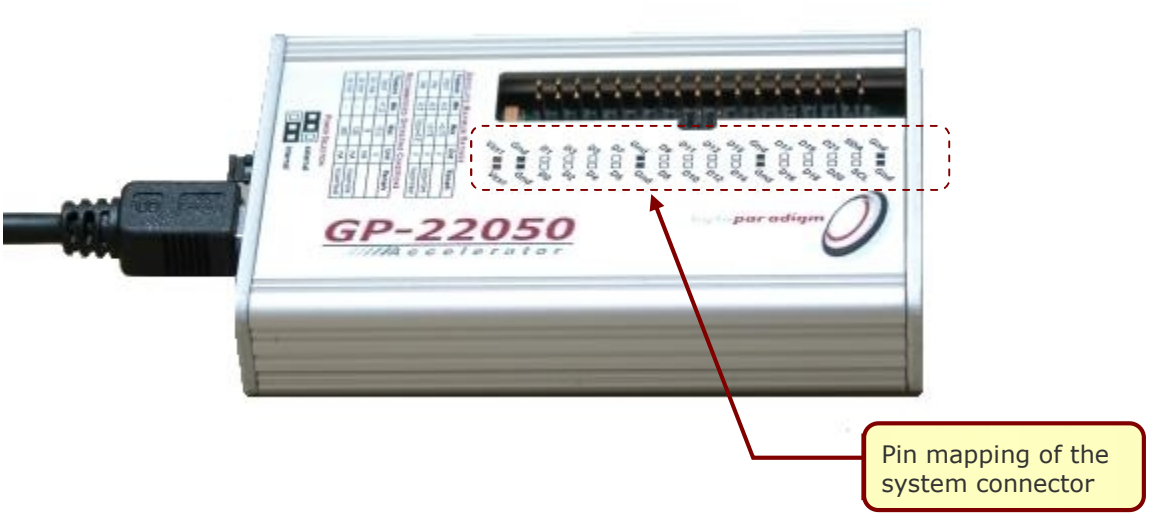
The **4-wires SPI** type is the 'traditional' SPI interface, with the standard signals: MOSI (Master Out Slave In), MISO (Master In Slave Out), CLK (Clock) and SSi (Slave i Select) signals. With this type of protocol, every signal is input-only or output-only.

The **3-wires SPI** type is a serial interface that uses a single bidirectional signal line (MOSI/MISO) to exchange data between the master and the slave(s). The other signals are: CLK (Clock) and SSi (Slave i Select). A 4th signal line is added to it for maximum flexibility: WE (Write Enable), active when the master writes data onto the MOSI/MISO signal line.

7.2 SPI interface signals

Whether you use a GP Series device or a SPI Xpress device, the pin mapping of the connector is described on the top side of the device (Refer to Figure 27). For more details about the system connector, refer to [1].

Figure 26: View of the GP-22050 top side



The SPI pin mapping is identical on the GP Series. SPI labelling is directly available on top of the SPI Xpress device; it is mapped onto the data/control lines of the GP Series devices (Figure 27).

Figure 27: Pin mapping of SPI signals on the GP Series system connector (hardware revision 01)

Vext	***0***	***1***	***Vext***
GND	**2**	**3**	GND
MOSI (MISO[1]) - D0	**4**	**5**	D1 - **MISO**
SCLK input (Analyser mode) D2	**6**	**7**	D3
D4	**8**	**9**	D5
D6	**10**	**11**	D7
GND	**12**	**13**	GND
SS1 / D8	**14**	**15**	D9 / **SS2**
SS3 / D10	**16**	**17**	D11 / **SS4**
SS5 / D12	**18**	**19**	D13
D14	**20**	**21**	D15
GND	**22**	**23**	GND
WE / C0 / D16	**24**	**25**	D17 / C1
C2 / D18	**26**	**27**	D19 / C3
C4 / CKin / D20	**28**	**29**	D21 / CKout / C5 / **SCLK output (master mode)**
SCL	**30**	**31**	SDA
GND	**32**	**33**	GND

[1] for SPI 3-wire bus architecture only

Figure 28: Pin mapping of SPI signals on the GP Series system connector (hardware revision 02 and higher)

CkIn	***0***	***1***	**SCLK output (master mode)**
GND	**2**	**3**	GND
MOSI (MISO[1]) - D0	**4**	**5**	D1 - **MISO**
SCLK input (Analyser mode) D2	**6**	**7**	D3
D4	**8**	**9**	D5
D6	**10**	**11**	D7
GND	**12**	**13**	GND
SS1 / D8	**14**	**15**	D9 / **SS2**
SS3 / D10	**16**	**17**	D11 / **SS4**
SS5 / D12	**18**	**19**	D13
D14	**20**	**21**	D15
GND	**22**	**23**	GND
WE / C0 / D16	**24**	**25**	D17 / C1
C2 / D18	**26**	**27**	D19 / C3
C4 / CKin / D20	**28**	**29**	D21 / CKout / C5
SCL	**30**	**31**	SDA
GND	**32**	**33**	GND

[1] for SPI 3-wire bus architecture only

Figure 29: Pin mapping of SPI signals on the SPI Xpress connector (hardware revision 01)

Vext	***0***	***1***	***Vext***
GND	**2**	**3**	GND
MOSI (MISO[1])	**4**	**5**	**MISO**
SCLK input (Analyser mode)	**6**	**7**	nc
nc	**8**	**9**	nc
nc	**10**	**11**	nc
GND	**12**	**13**	GND
SS1	**14**	**15**	**SS2**
SS3	**16**	**17**	**SS4**
SS5	**18**	**19**	nc
Nc	**20**	**21**	nc
GND	**22**	**23**	GND
WE / C0	**24**	**25**	**C1**
C2	**26**	**27**	**C3**
C4	**28**	**29**	**SCLK output (master mode)**
nc	**30**	**31**	nc
GND	**32**	**33**	GND

[1] for SPI 3-wire bus architecture only

Figure 30: Pin mapping of SPI signals on the SPI Xpress connector (hardware revision 02 and higher)

CkIn	***0***	***1***	**SCLK output (master mode)**
GND	**2**	**3**	GND
MOSI (MISO[1])	**4**	**5**	**MISO**
SCLK input (Analyser mode)	**6**	**7**	nc
nc	**8**	**9**	nc
nc	**10**	**11**	nc
GND	**12**	**13**	GND
SS1	**14**	**15**	**SS2**
SS3	**16**	**17**	**SS4**
SS5	**18**	**19**	nc
Nc	**20**	**21**	nc
GND	**22**	**23**	GND
WE / C0	**24**	**25**	**C1**
C2	**26**	**27**	**C3**
C4	**28**	**29**	**C5**
nc	**30**	**31**	nc
GND	**32**	**33**	GND

[1] for SPI 3-wire bus architecture only

7.3 SPI Application Page

The SPI mode of operation page can be opened using the *File* > *New* application menu and then by selecting *Spi* in the dialog. Figure 31 gives an overview of the SPI page. The page can also be opened using the *New Page* pane. If the pane is not visible go to menu *View* > *New Page Pane*. When opening an SPI configuration file using the *File* > *Open* menu, a new page is automatically created.

7.4 Device Configuration in SPI mode of operation

7.4.1 SPI Configuration

Figure 31: SPI configuration tab



7.4.1.1 SPI Configuration

CONTINUOUS CLOCK

- ☑ When checked, the output clock is continuous and independent of the data stream.
- ☐ When not checked, the output clock is generated as a "hole" clock. Clock edges are only sent out when a data written or read or during the latency time between write and read

in SPI3 mode. When no data transits on the MOSI/MISO signal lines, the clock signals remains at its default level.

REQUESTED

This field defines the frequency of the operating clock used in the device. The frequency must be defined in a range between 50 MHz and 800Hz. The value entered in this field must be a decimal value having a maximum of 6 decimal digits and representing the frequency in MHz. This is the clock used for the SPI mode: it represents the SPI clock of the SPI Master and the sample clock of the SPI Analyser.

SPI MODE

This drop-down menu allows to choose the desired SPI Mode.

NUMBER OF SLAVES

This field defines the total number of slaves connected to the device used as a SPI master or SPI analyser. Up to 5 slaves can be specified.

LOG WINDOW

☑ When checked, displays the SPI Master/Analyser log window.

☐ When not checked, the SPI Master/Analyser log window is hidden.

7.4.1.2 Configuration

LOAD

Clicking on this button opens a browse window to select a configuration file to load.

SAVE

Clicking on this button opens a browse window to select a path and specify a file name where to save the current SPI configuration.

EDIT

Clicking on this button opens the NotePad to edit the current configuration file. If no configuration file is currently loaded, it opens a browse window to select it.

7.4.2 SPI Mode Summary

This area of the Configuration tab summarises the chosen mode configuration: CPOL (clock polarity), CPHA (clock phase), Clock idle level, MOSI toggling edge and MISO sampling edge.

7.5 SPI Master

7.5.1 Overview of the SPI Master protocols

This section summarises the range of SPI protocols that can be used with the GP-22050 SPI mode of operation.

7.5.1.1 Modes

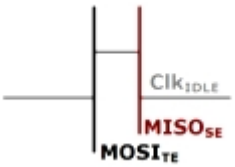
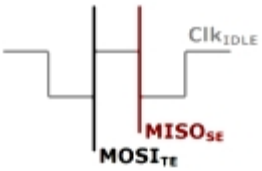
SPI protocol modes define the following parameters, that relate to the SPI interface clock (CLK) signal:

- CPOL : clock polarity
- CPHA : clock phase
- Clock idle level
- MOSI toggling edge
- MISO sampling edge

Actually, CPOL and CPHA define the other parameters. If the phase of the clock is zero, i.e. CPHA = 0, data is latched at the rising edge of the clock with CPOL = 0, and at the falling edge of the clock with CPOL = 1. If CPHA = 1, the polarities are reversed. CPOL = 0 means falling edge, CPOL = 1 rising edge.

Modes are defined by a pair of values for CPOL, CPHA, as shown in Table 7.

Table 7: SPI modes overview

Mode	*CPOL*	*CPHA*	*Overview*
0	0	0	
1	0	1	
2	1	0	
3	1	1	

Legend:
$MOSI_{TE}$ = MOSI toggling edge;
$MISO_{SE}$ = MISO sampling edge;
CLK_{IDLE} = Clock idle level

7.5.1.2 Bit ordering

When using the SPI mode of operation the data written on the MOSI line and read from the MISO line are formatted as hexadecimal numbers. *The LS Byte is always transmitted first*. The bit ordering within each byte can be selected.

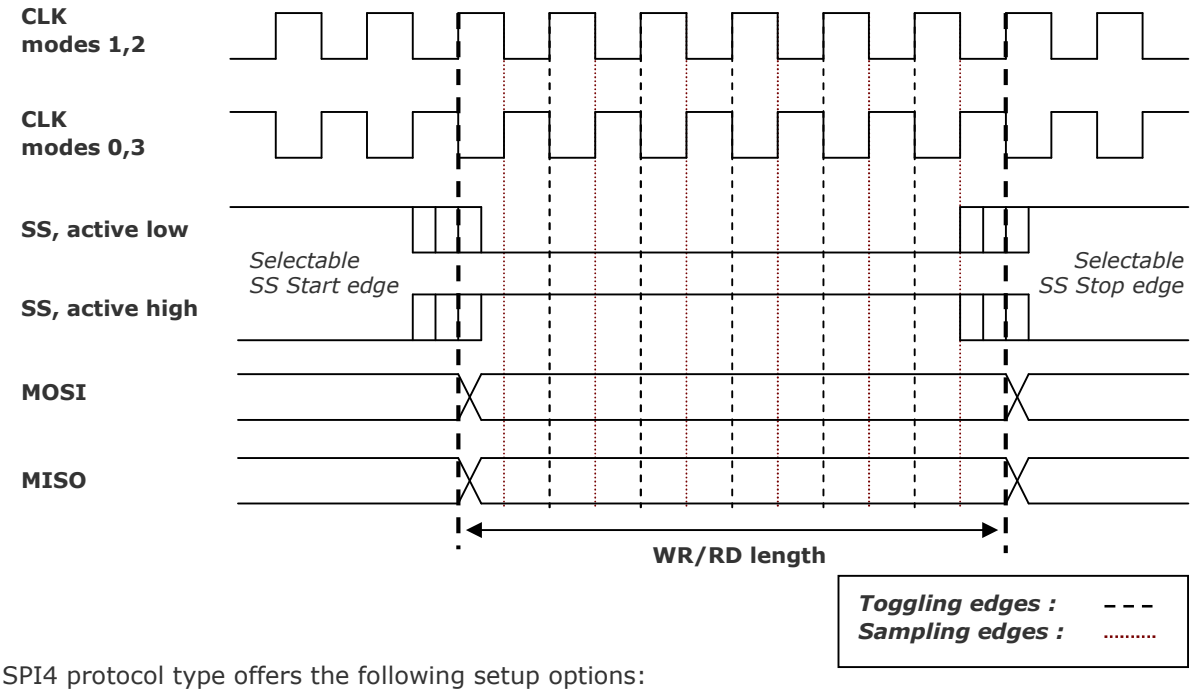
Example: transmit 24 bits: Out = 0xF1E2D3

Byte ordering		***0xD3***	***0xE2***	***0xF1***
Bit ordering within byte	LS bit first	<1100><1011>	<0100><0111>	<1000><1111>
	MS bit first	<1101><0011>	<1110><0010>	<1111><0001>

Time line

7.5.1.3 4-wires SPI procotol – SPI4

Figure 32: 4-wires SPI protocol overview



SPI4 protocol type offers the following setup options:

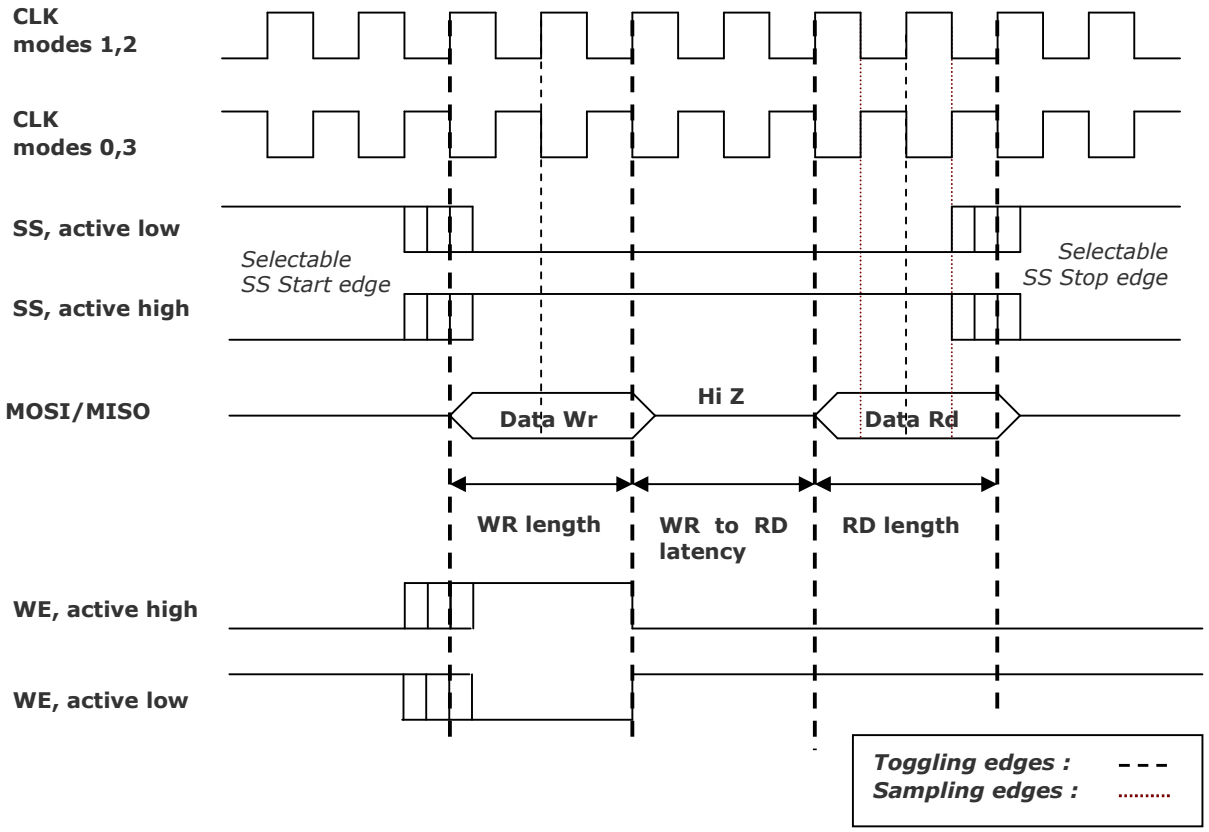
- Clock polarity / phase selection (through mode selection). This also defines the MOSI / MISO toggling and MISO sampling edges.
- Continuous / Hole clock (not represented on the above picture).
- Data bit ordering within each Byte.
- SS polarity selection (active high / active low).
- SS start and stop edges selection, with the following limitations:

Reference	***SS Start edge position***	***Condition***
First MOSI toggling edge	- 2 x Tclk / 4	fCLK ≤ 25 MHz
	- 1 x Tclk / 4	fCLK ≤ 12.5 MHz
	0	none
	+ 1 x Tclk / 4	fCLK ≤ 12.5 MHz
Reference	***SS Stop edge position***	***Condition***
Last MOSI toggling edge	- 2 x Tclk / 4	fCLK ≤ 25 MHz
	- 1 x Tclk / 4	fCLK ≤ 12.5 MHz
	0	none
	+ 1 x Tclk / 4	fCLK ≤ 12.5 MHz

- Selectable write/read length for each SPI access. Read sequence length sampled on MISO is equal to write sequence length generated on MOSI. WR/RD length is max. 32.000 bits.
- Up to 5 selectable slaves (each slave is connected to a separate SS line).
- **Refer to Table 10 for more information on SPI parameters.**

7.5.1.4 3-wires SPI procotol – SPI3

Figure 33: 3-wires SPI protocol overview



SPI3 protocol type offers the following setup options:

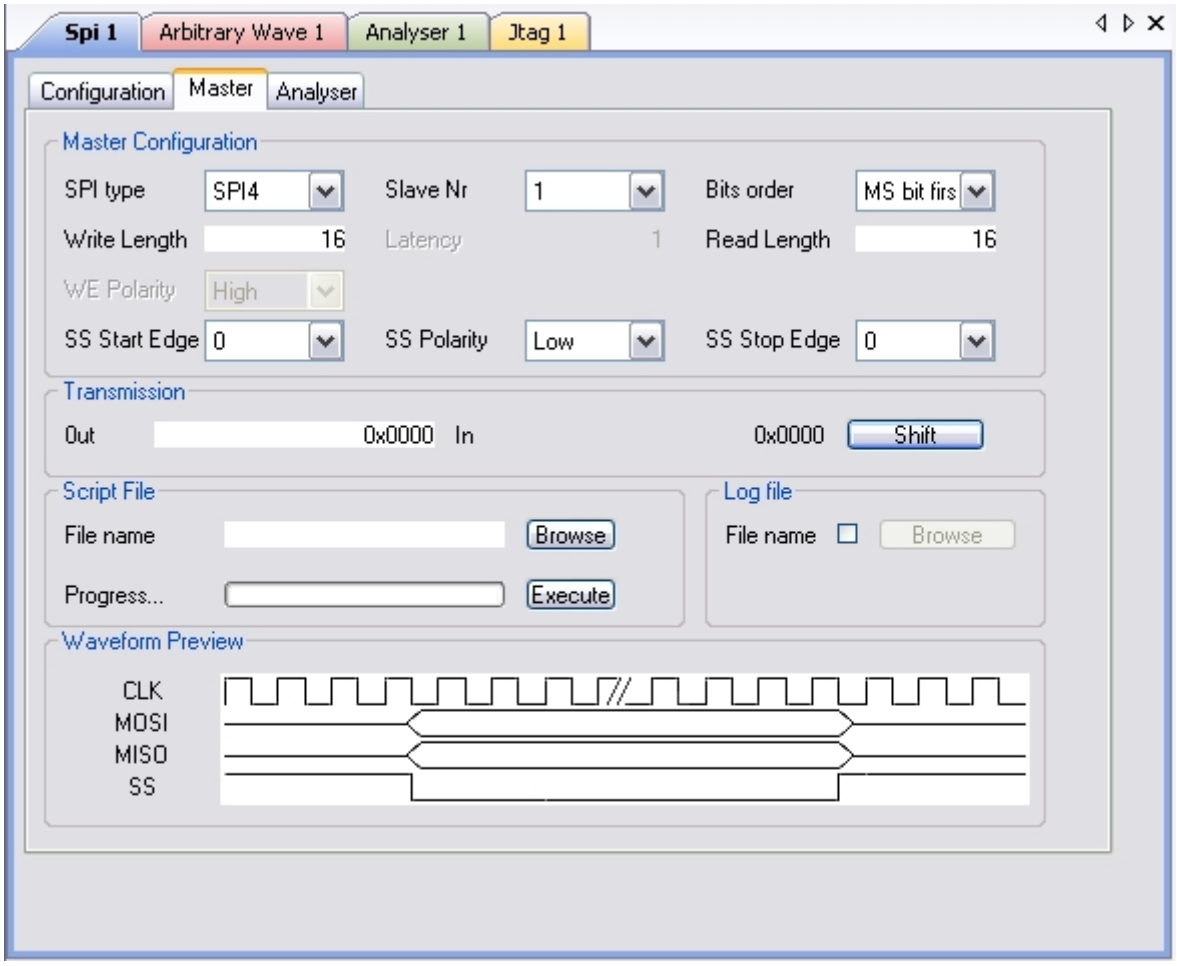
- Clock polarity / phase selection (through mode selection). This also defines the MOSI / MISO toggling and MISO sampling edges.
- Continuous / Hole clock (not represented on the above picture)
- Data bit ordering within each Byte.
- SS polarity selection (active high / active low).
- SS start and stop edges selection, with the following limitations:

Reference	***SS Start edge position***	***Condition***
First MOSI toggling edge	- 2 x Tclk / 4	fCLK ≤ 25 MHz
	- 1 x Tclk / 4	fCLK ≤ 12.5 MHz
	0	none
	+ 1 x Tclk / 4	fCLK ≤ 12.5 MHz
Reference	***SS Stop edge position***	***Condition***
Last MOSI toggling edge	- 2 x Tclk / 4	fCLK ≤ 25 MHz
	- 1 x Tclk / 4	fCLK ≤ 12.5 MHz
	0	none
	+ 1 x Tclk / 4	fCLK ≤ 12.5 MHz

- A standard transaction SPI3 transaction is composed of the following steps:
  - Write phase: the master writes data onto the MOSI/MISO signal lines. During the Write phase, no slave is allowed to drive the MOSI/MISO line (all slaves MOSI/MISO signal must be HI Z).
  - Write to read latency: the master MOSI line returns to HI Z. During the Write to read latency, the MOSI/MISO line is not driven.
  - Read phase: the slave selected during the write phase returns data onto the MOSI/MISO signal line. During the read phase, the master holds its MOSI/MISO line HI Z – and so for the other slaves.
- Write phase length, Read phase length and Write to Read latency can be specified separately.
- Maximum length: Write: 4.095 cycles; Latency: 400 cycles; Read: 4.095 cycles[1].
- Additional write enable signal (WE) marking the master write phase. By default, this signals becomes active together with the SS signal. The polarity of the WE signal can be selected.
- Up to 5 selectable slaves (each slave is connected to a separate SS line).

7.5.2 Interface description

Figure 34: SPI Master tab (SPI4 selected)



[1] **Refer to Table 10 for more information on SPI parameters.**

7.5.2.1 Master Configuration

SPI TYPE

This drop-down menu allows choosing between 4-wires SPI type protocol (SPI4) and 3-wires SPI type protocol (SPI3). According to the selected SPI type, different SPI Master configuration options are available.

SLAVE NR

This drop-down menu selects the target slave number for the next transaction. During the next transaction, the corresponding SS line will be used.

BITS ORDER

This drop-down menu selects data bits ordering within each data byte. Refer to section 7.5.1.2 for a description of this feature.

WRITE LENGTH

This field defines the length of the write transaction in clock (CLK) cycles. In SPI4 mode, changing this field automatically updates the Read Length field as well. In SPI3 mode, the two fields can be set up independently. Maximum run length is 32.000 for SPI4 mode and 4.095 for SPI3 mode **(refer to Table 10 for more information on SPI parameters)**

LATENCY

Available in SPI3 mode only. This field defines the length in clock (CLK) cycles of the latency time between the write phase and the read phase in SPI3 mode. Maximum latency length is 400 **(refer to Table 10 for more information on SPI parameters).**

READ LENGTH

This field defines the length of the read transaction in clock (CLK) cycles. In SPI4 mode, changing this field automatically updates the Write Length field as well. In SPI3 mode, the two fields can be set up independently. Maximum run length is 32.000 for SPI4 mode and 4.095 for SPI3 mode **(refer to Table 10 for more information on SPI parameters).**

WE POLARITY

Available in SPI3 mode only. This drop-down menu selects the polarity of the WE signal line.

SS POLARITY

This drop-down menu selects the polarity of the SS signal line(s).

SS START EDGE

This drop-down menu selects the position of the SS signal start edge. Values -2, -1, 0 and 1 can be selected. The selected value defines the position of the SS start edge relatively to its conventional clock (CLK) starting edge, in $1/4^{th}$ of the clock period.

SS STOP EDGE

This drop-down menu selects the position of the SS signal stop edge. Values -2, -1, 0 and 1 can be selected. The selected value defines the position of the SS stop edge relatively to its conventional clock (CLK) stopping edge, in $1/4^{th}$ of the clock period.

Positionning SS at ½ or ¼ of SCLK period requires internal overclocking of the GP or Xpress device. This oversampling cannot exceed the device internal maximum clock frequency. This defines the following maximum SCLK frequencies if the SS edge prosition is chosen.

SS positioning case	***Max. SCLK frequency***
No positioning (SS synchronous to SCLK)	50 MHz
½ SCLK	25 MHz
¼ SCLK	12.5 MHz

If a different positioning is applied to the SS start edge and SS stop edge, then the lowest Max. SCLK frequency is selected.

7.5.2.2 Transmission

OUT

This field is used to enter the data to be written out onto the MOSI line in SPI4 mode and onto the MOSI/MISO line during the Write phase in SPI3 mode. The data must be entered as a hexadecimal number preceded by "0x". Refer to section 7.5.1.2 for information about byte and bit ordering.

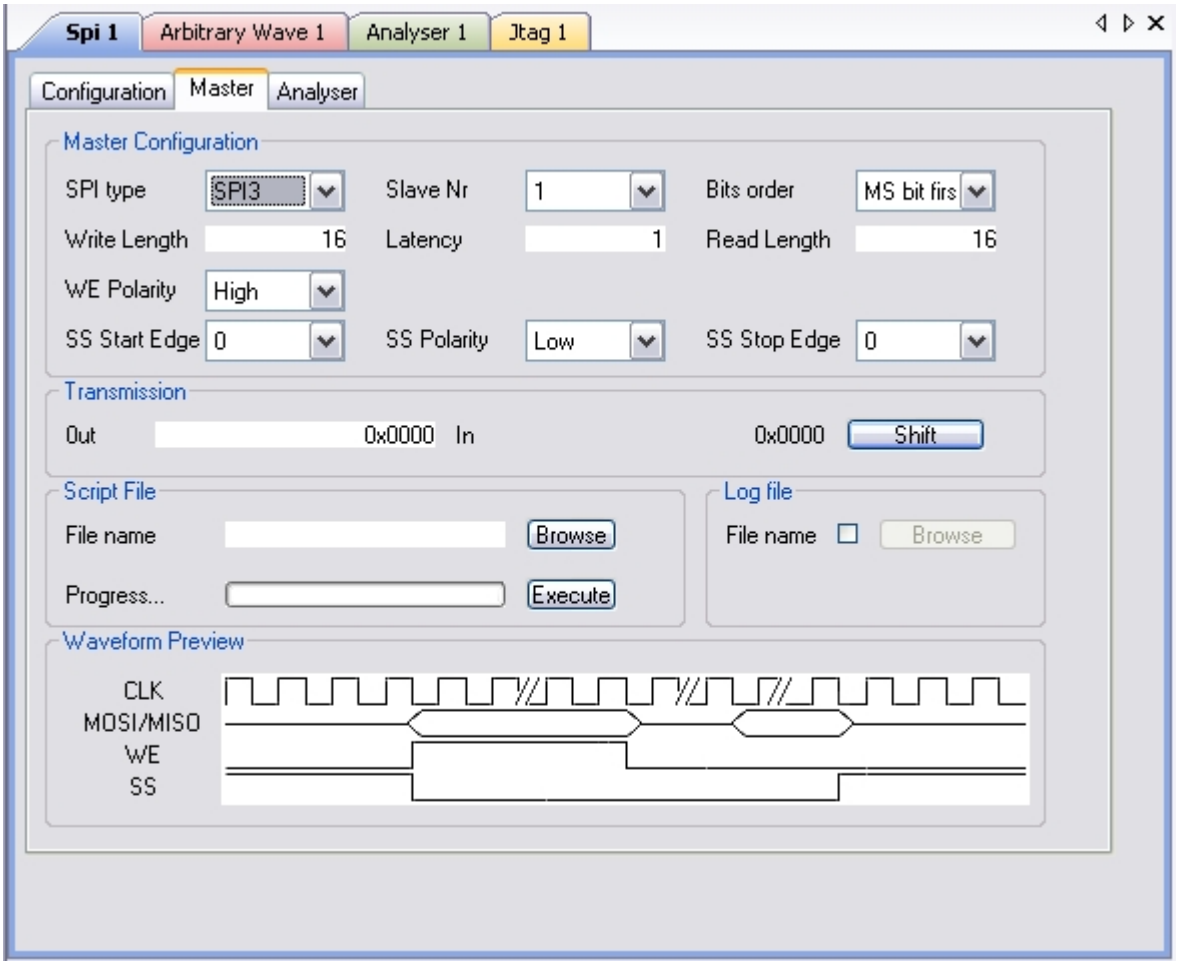
IN

(Not editable). This field returns the data read during the SPI transaction, according to the master settings.

SHIFT

Clicking on this button starts the programmed transmission.

Figure 35: SPI Master tab (SPI3 selected)



7.5.2.3 Script File

FILE NAME

This field is used to enter the path and file name of the script file to play with the SPI master. The 'Browse' button can be select the file from a browse window.

BROWSE

This button opens a window to select the script file to execute.

PROGRESS...

(Not editable). This progress bar displays the chosen script execution progress.

7.5.2.4 Log File

FILE NAME

☑ When checked, enables logging and activates a file name selection field / browse button.

☐ When not checked, the logging is disabled.

7.5.2.5 Waveform Preview

This area gives a pictural overview of most of the selected settings. This picture is for information and ease of use only; please refer to the documentation for more accurate information on signal sequences and timings.

7.6 SPI Analyser

7.6.1 Overview of the SPI Analyser features

The SPI Analyser basically works like a **logic analyser**. As such, the user is supposed to correctly select the adequate sampling frequency, with the right oversampling with respects to the source signal frequency in order to have sufficient resolution and visibility over the signals. Please refer to the SPI Configuration section (section 7.4.1.1) about how to select the sampling clock frequency.

Each SPI protocol analysis is based on a sampling window, which is defined by specifying a total number of samples. When performing an analysis, the user has got the choice to manually start the analysis (internal trigger mode) or to use a trigger defined with a combination of up to 6 external control signals. The trigger position within the sampling window can also be selected.

The SPI Analyser automatically performs an analysis of the sampled SPI transaction according to a set of parameters:

- **the polarity of the SS# signals** (active high or active low);
- **the type of SPI protocol** (4 wires or 3 wires);
- in the case of a 3-wires SPI protocol, the **expected length** of the write, latency and read phases (please refer to section 7.5.1.4)
- **data bits ordering**.

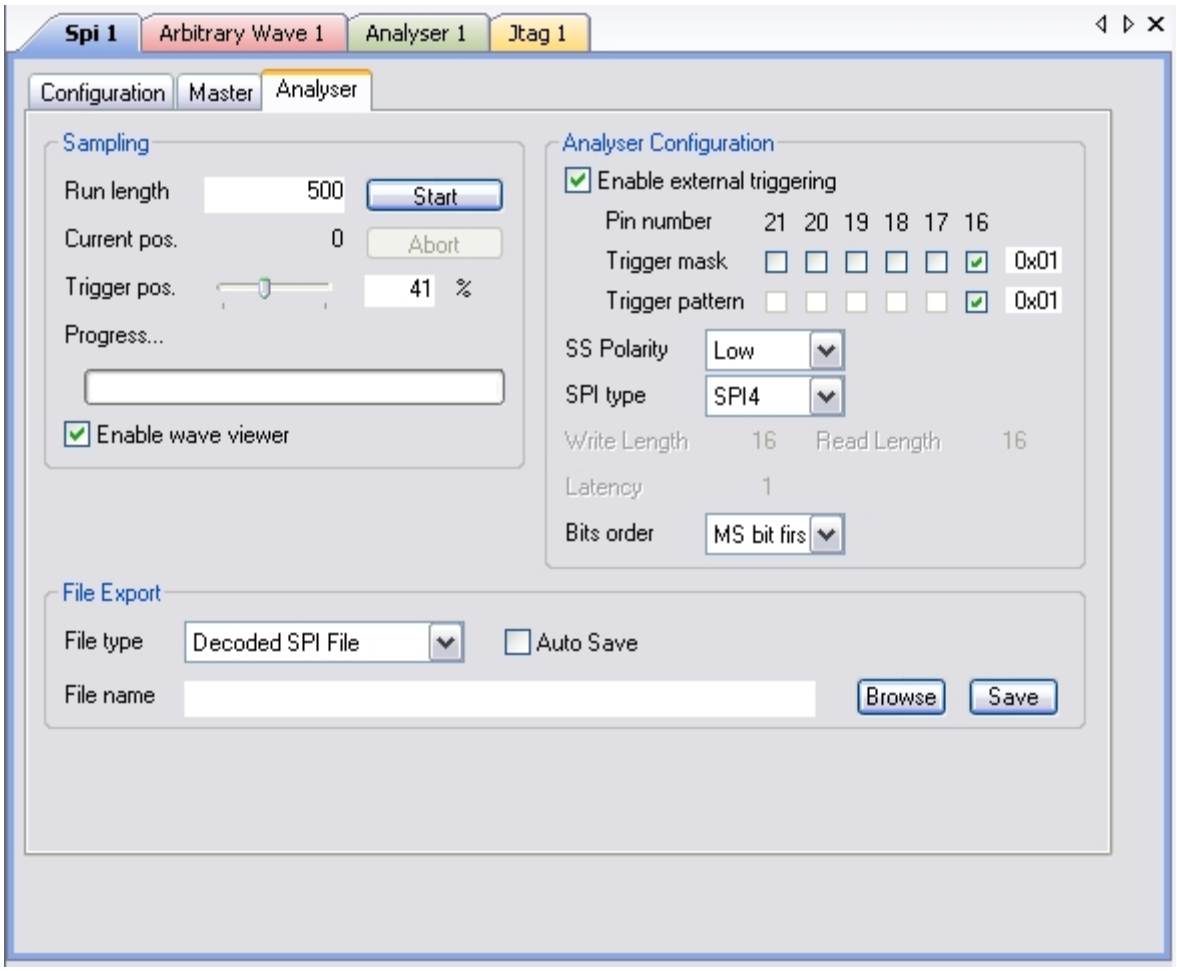
The SPI analysis can be displayed into a logging window and/or saved to a file. 3 different data formats are available for analysis display and saving:

- **Raw Data**: in this mode, the SPI Analyser just displays all the samples of each of the SPI protocol signal line.
- **Raw SPI**: in this mode, the SPI Analyser detects the SPI clock edges and displays each of the SPI protocol signal line at the clock transitions.
- **Decoded SPI**: in this mode, the SPI Analyser decodes the sampled window into fully decoded SPI transaction (SPI type, shift length, …).

Additionally to the decoded display / saving, the waveform data can be displayed into the embedded GTK Wave waveform viewer.

7.6.2 Interface description

Figure 36: SPI Analyser tab (SPI4 selected)



7.6.2.1 Sampling

RUN LENGTH

This field defines the number of samples to collect and analyse during the next run.

START

Clicking on this button starts the SPI Analyser run.

ABORT

Clicking on this button stops the SPI Analyser before the end of the run.

CURRENT POS.

(Not editable). This field returns the sample index being sampled during a run.

TRIGGER POS.

This cursor selects the position of the trigger within the sampling window. A percentage field displays the position currently selected.

PROGRESS...

(Not editable). This progress bar displays the run execution progress.

ENABLE WAVE VIEWER

☑ When checked, the GTK Wave waveform viewer is automatically started at the end of the SPI analyser run to display the sampled data.

☐ When not checked, the GTK Wave waveform viewer is not started at the end of the run.

7.6.2.2 Analyser Configuration

ENABLE EXTERNAL TRIGGERING

☑ When checked, an external trigger is used to start the SPI Analyser run. In this case, the trigger mask and trigger pattern controls are accessible from the SPI Analyser page.

☐ When not checked, the SPI Analyser run will start directly when clicking the Start button.

TRIGGER MASK

☑ When selected, the corresponding control line is selected to be used as input for the external triggering. It remains operating as an input during the complete SPI Analyser sequence.

☐ When not selected, the pin is not reserved for external triggering.

TRIGGER PATTERN

The *trigger pattern* can only be defined if at least one control line is selected as input in the external *trigger mask*. The *trigger pattern* defines the logic levels that must be detected on the input control lines to generate the trigger event. The level applied on the control lines not selected for the trigger is ignored while detecting the trigger pattern.

☑ When selected, logic 1 is associated to the control line.

☐ When not selected, logic 0 is associated to the control line.

ⓘ *Instead of clicking on the different check box of the trigger mask and pattern, the expected values can be directly entered in hexadecimal format with prefix 0x in the corresponding field.*

SS POLARITY

This drop-down menu selects the polarity of the SS signal line(s).

SPI TYPE

This drop-down menu allows choosing between 4-wires SPI type protocol (SPI4) and 3-wires SPI type protocol (SPI3). According to the selected SPI type, different SPI Analyser configuration options are available.

WRITE LENGTH

Available in SPI3 mode only. This field defines the length of the write transaction in clock (CLK) cycles.

LATENCY

Available in SPI3 mode only. This field defines the length in clock (CLK) cycles of the latency time between the write phase and the read phase in SPI3 mode.

READ LENGTH

Available in SPI3 mode only. This field defines the length of the read transaction in clock (CLK) cycles.

7.6.2.3 File Export

FILE TYPE

This drop-down menu selects the decoding format of the SPI Analyser.

AUTO SAVE

☑ When checked, enables autosave feature: at the end of the SPI Analyser run, the data decoded according to the selected format are automatically saved.

☐ When not checked, the auto save feature is disabled.

FILE NAME

This field is used to enter the path and file name of the SPI Analyser output file.

BROWSE

This button opens a browse window to select the SPI Analyser output file.

SAVE

Clicking on this button saves the SPI Analyser data, according to the selected format.

7.6.2.4 Logging Window

In SPI Analyser mode, the logging window displays the sampled data in the selected format. Please refer to the SPI Configuration tab to enable/disable it (section 7.4.1) and to the File Export group on the Analyse tab (section 7.6.2.3) to select the SPI Analyser output format.

7.7 File format

A total of 3 different files are handled by the SPI mode of operation.

- Configuration file — Text file containing device configuration (SPI protocol type, clock settings, etc.)
- SPI Master script / log file — Text file holding a set of SPI command for the master. The script file and the log file have the same format; the log file records the commands sent with the SPI Master; the script file allows to replay a set of previously recorded SPI commands.
- SPI Analyser export file — File containing the SPI Analyser recorded data. Several formats are supported (raw data, raw SPI, decoded SPI). Decoded SPI format is identical to the SPI master script/log file format.

7.7.1 Configuration file

The SPI mode of operation configuration can be saved to / recalled from an external configuration file. The available commands and syntax are listed hereafter.

@clock type <continuous/hole>

This command defines the output clock continuity. When *continuous* is selected, a permanent clock is driven. When *hole* is selected, a non continuous clock is generated. In this mode, a single clock pulse is generated each time a data is sent out. If no data is sent out, the clock remains at the defined idle level.

@freq <freq_in_Hz>

Defines the requested operating frequency. The frequency must be a decimal value representing the frequency expressed in Hz. The value must be defined between 50MHz and 800Hz.

@type <SPI3/SPI4>

Defines the type of SPI interface. Possible SPI type values: SPI4 or SPI3.

@mode <SPI mode>

Defines mode of the SPI interface – integer value from 0 to 4.

@slave amount <number of slaves>

Defines the number of slaves connected to the SPI interface. Valid value from 1 to 5.

@slave selected <slave reference>

Selects the slave reference number for SPI Master operation. Valid value from 1 to 5.

@trigger type <internal/external>

This command defines if an *internal* or *external* trigger must be used.

@trigger ctrl <pattern> <mask>

(SPI Analyser mode). If external triggering is selected, this command defines the trigger *pattern* to detect on the control lines to generate the trigger event. The *mask* defines which control lines have to be considered for the trigger pattern detection.

If the internal trigger mode is selected, this command is ignored.

The user must take care not to use the same control line for triggering and control sequence definition. Each control line can only have a single functionality: trigger or control sequence. If a conflict is created, the trigger has the priority.

@slave edges <start edge> <stop edge>

Defines the local phase shift for the SS signal(s). The *start edge* and *stop edge* parameters define the corresponding edge position relatively to the conventional SPI clock edge chosen to generate data. Each phase shift is programmed with an integer ranging from -2 to 1, which represents the number of 1/4th of the SPI clock period

@slave polarity <high/low>

Defines the polarity of the SS# signals. Valid values: 'high' for 'active high'; 'low' for 'active low'.

@firstbit <lsb/msb>

Defines bit ordering within each data byte. The data bytes are always transmitted with the least significant byte first. If 'lsb' is selected, the least significant bit of each byte is transmitted/decoded first; if 'msb' is selected, the most significant bit of each byte is transmitted/decoded first.

Figure 37: Example of SPI Configuration file

```
#
# SPI configuration file
#

# Clock continuity
@clock type continuous

# Requested clock frequency
@freq 9000000

# SPI type
@type SPI4

# SPI mode
@mode 2

# Number of slaves
@slave amount 5

# Selected slave
@slave selected 1

# Trigger source
@trigger type external

# Trigger mask and pattern
@trigger ctrl 0x50 0x50

# Slave select edges
@slave edges -1 -1

# Slave active level
@slave polarity high

# Bit ordering
@firstbit msb
```

7.7.2 SPI Master script / log file

2 types of files can be used with the SPI Master:

- All commands sent during a SPI Master session can be recorded to a log file.
- A sequence of commands can be replayed from an input script file.

Both the script file and the log file use the same syntax, so the user has got the ability to record and replay a sequence of SPI commands using the GUI.

The syntax used in the script / log file is described hereafter.

@idle <number of clock cycles>

Defines a number of clock cycles (relative to the SPI clock period) during which the SPI interface remains 'idle' (all lines are held in their default state).

@shift4 <slave nr> <length> <data out> <data in>

SPI4 shift command. With this command, the SPI master shifts data onto the MOSI line while it samples the data from the MISO line.

'slave nr': defines the slave number activated during the shift (corresponding SS active);

'length': defines the length of the shift in number of SPI clock cycles;

'data out': defines or shows the data put on or recorded from the MOSI line;

'data in': shows the data recorded from the MISO line (no effect if the file is used as a script).

@shift3 <slave nr> <write length> <latency length> <read length> <data out> <data in>

SPI3 shift command. With this command, the SPI master uses one single MOSI/MISO line in a 'write-then-read' sequence. First the master shifts (drives) data onto the MOSI/MISO line used as MOSI; then, the MOSI/MISO line is held in high impedance state (latency); finally, the connected slave answers onto the MOSI/MISO line used as MISO. The command parameters define the slave number, the respective length of the write phase, the latency phase and the read phase, the data written by the master and the data provided by the slave.

'slave nr': defines the slave number activated during the shift (corresponding SS active);

'write length': defines the length of the write phase in number of SPI clock cycles; during the write phase, the SPI master drives the MOSI/MISO line.

'latency length': defines the length of the write-to-read latency in number of SPI clock cycles. During the latency, the MOSI/MISO line is held in high-Z state.

'read length': defines the length of the read phase in number of SPI clock cycles; during the read phase, the SPI slave drives the MOSI/MISO line.

'data out': defines or shows the data put onto the MOSI/MISO line by the master during the write phase.

'data in': shows the data put onto the MOSI/MISO line by the slave during the read phase (no effect if the file is used as a script).

Parameter	***Min***	***Max***	***Unit***
SPI 4 mode Access length	1	32.000	bit / SCLK cycle
SPI 3 mode WRITE length	1	4.095	bit / SCLK cycle
SPI 3 mode latency length	0	400	bit / SCLK cycle
SPI 3 mode READ length	0	4.095	bit / SCLK cycle
Refer to Table 10 for complete details on SPI access length			

Table 8 : SPI mode of operation data length / mode / operation

7.7.3 SPI Analyser export file

The SPI Analyser records the activity on a given SPI interface. It can export its data into 3 formats:

- Raw Data File;
- Raw SPI File;
- Decoded SPI File.

7.7.3.1 Raw Data File

In this format, all the samples of each of the protocol signal lines are given out as raw data.

Figure 38: Example of Raw Data file

CLK	MOSI	MISO	Slave 0	Slave 1	Slave 2	Slave 3	Slave 4
1	1	0	1	1	1	1	1
1	1	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	0	1	1	1
0	1	0	1	0	1	1	1
1	0	0	1	0	1	1	1
1	1	0	1	0	1	1	1
0	1	0	1	0	1	1	1
0	1	0	1	0	1	1	1
0	1	0	1	0	1	1	1
1	0	0	1	0	1	1	1
1	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
1	0	0	1	1	1	1	1
1	0	0	1	1	1	1	1
0	1	0	0	1	1	1	1
0	1	0	0	1	1	1	1
0	1	1	0	1	1	1	1
1	0	1	0	1	1	1	1
1	0	1	0	1	1	1	1
0	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1
0	1	0	0	1	1	1	1

7.7.3.2 Raw SPI File

In this format, the SPI Analyser detects the SPI clock edges and saves each of the SPI protocol signal line at the SPI clock transitions.

Figure 39: Example of Raw SPI file

CLK	MOSI	MISO	Slave 0	Slave 1	Slave 2	Slave 3	Slave 4
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	1
0	0	0	0	1	1	1	1
0	1	1	0	1	1	1	1
0	0	1	0	1	1	1	1
0	1	0	0	1	1	1	1
0	1	0	0	1	1	1	1
0	0	1	0	1	1	1	1
0	0	1	0	1	1	1	1
0	0	1	0	1	1	1	1
0	1	1	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	1	0	1	1	1	1
0	1	0	0	1	1	1	1
0	0	1	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	1	0	1	1	1	1
0	0	0	1	1	1	1	1

7.7.3.3 Decoded SPI File

In this format, the SPI Analyser decodes the SPI transactions from the SPI interface. The used syntax is identical to this of the SPI Master script / log file – please refer to section 7.7.2 for more details.

8 I²C

Applies to GP Series devices and I2C Xpress device.

8.1 Features

This application is delivered to use the GP Series devices as a I²C protocol master and I²C protocol analyser. It is also the default application for the I²C Xpress device. With the I2C mode of operation of the 8PI Control Panel, you can use your GP Series or I²C Xpress device to control or analyse interfaces that use the I²C protocol. As a master, it enables access to one or several I²C slaves as a master, for write and read access. As an analyser, it is used to sample, visualize and decode transactions occurring on a I²C interconnect. Main features:

- 10 kbps, 100 kbps, 400 kbps and 1000 kbps speeds supported
- Multi-master bus support
- 7 and 10 bits slave addressing support
- Repeated start support
- Clock stretching support

These features are also accessible with the provided C/C++ DLL and TCL library. Refer to [15] and [16] for more details on how to interface I²C ports, using respectively the C/C++ library and the Tcl scripting environment.

8.2 I²C interface signals

Whether you use a GP Series device or a I²C Xpress device, the pin mapping of the connector is described on the top side of the device (Refer to Figure 40 and Figure 42). For more details about the system connector, refer to [1] and [14].

From **8PI Control Panel version 1.08f**, 2 separate SDA, SCL ports can be selected for the I2C Master / Analyzer. One of the two interfaces features pull-up resistors; the other does not. Controls are provided with the 8PI Control Panel GUI or the DLL / TCL libraries to select the port used.

Figure 40: Pin mapping of I²C signals on the GP Series system connector (hardware revision 01)

Vext	***0***	***1***	***Vext***
GND	**2**	**3**	GND
***SCL without pull-up* - D0**	**4**	**5**	**D1 – *SDA without pull-up***
D2	**6**	**7**	D3
D4	**8**	**9**	D5
D6	**10**	**11**	D7
GND	**12**	**13**	GND
D8	**14**	**15**	D9
D10	**16**	**17**	D11
D12	**18**	**19**	D13
D14	**20**	**21**	D15
GND	**22**	**23**	GND
D16	**24**	**25**	D17
D18	**26**	**27**	D19
D20	**28**	**29**	D21
***with pull-up* - SCL**	**30**	**31**	**SDA - *with pull-up***
GND	**32**	**33**	GND

Figure 41: Pin mapping of I²C signals on the GP Series system connector (hardware revision 02 and higher)

CkIn	***0***	***1***	*CkOut*
GND	**2**	**3**	GND
***SCL without pull-up* - D0**	**4**	**5**	**D1 – *SDA without pull-up***
D2	**6**	**7**	D3
D4	**8**	**9**	D5
D6	**10**	**11**	D7
GND	**12**	**13**	GND
D8	**14**	**15**	D9
D10	**16**	**17**	D11
D12	**18**	**19**	D13
D14	**20**	**21**	D15
GND	**22**	**23**	GND
D16	**24**	**25**	D17
D18	**26**	**27**	D19
D20	**28**	**29**	D21
***with pull-up* - SCL**	**30**	**31**	**SDA - *with pull-up***
GND	**32**	**33**	GND

Figure 42: Pin mapping of the I²C Xpress device system connector (hardware revision 01)

Vext	***0***	***1***	***Vext***
GND	**2**	**3**	GND
***SCL without pull-up* - D0**	**4**	**5**	**D1 – *SDA without pull-up***
Nc	**6**	**7**	Nc
Nc	**8**	**9**	Nc
Nc	**10**	**11**	Nc
GND	**12**	**13**	GND
Nc	**14**	**15**	Nc
Nc	**16**	**17**	Nc
Nc	**18**	**19**	Nc
Nc	**20**	**21**	Nc
GND	**22**	**23**	GND
Nc	**24**	**25**	Nc
Nc	**26**	**27**	Nc
Nc	**28**	**29**	Nc
***with pull-up* - SCL**	**30**	**31**	**SDA - *with pull-up***
GND	**32**	**33**	GND

Figure 43: Pin mapping of the I²C Xpress device system connector (hardware revision 02 and higher)

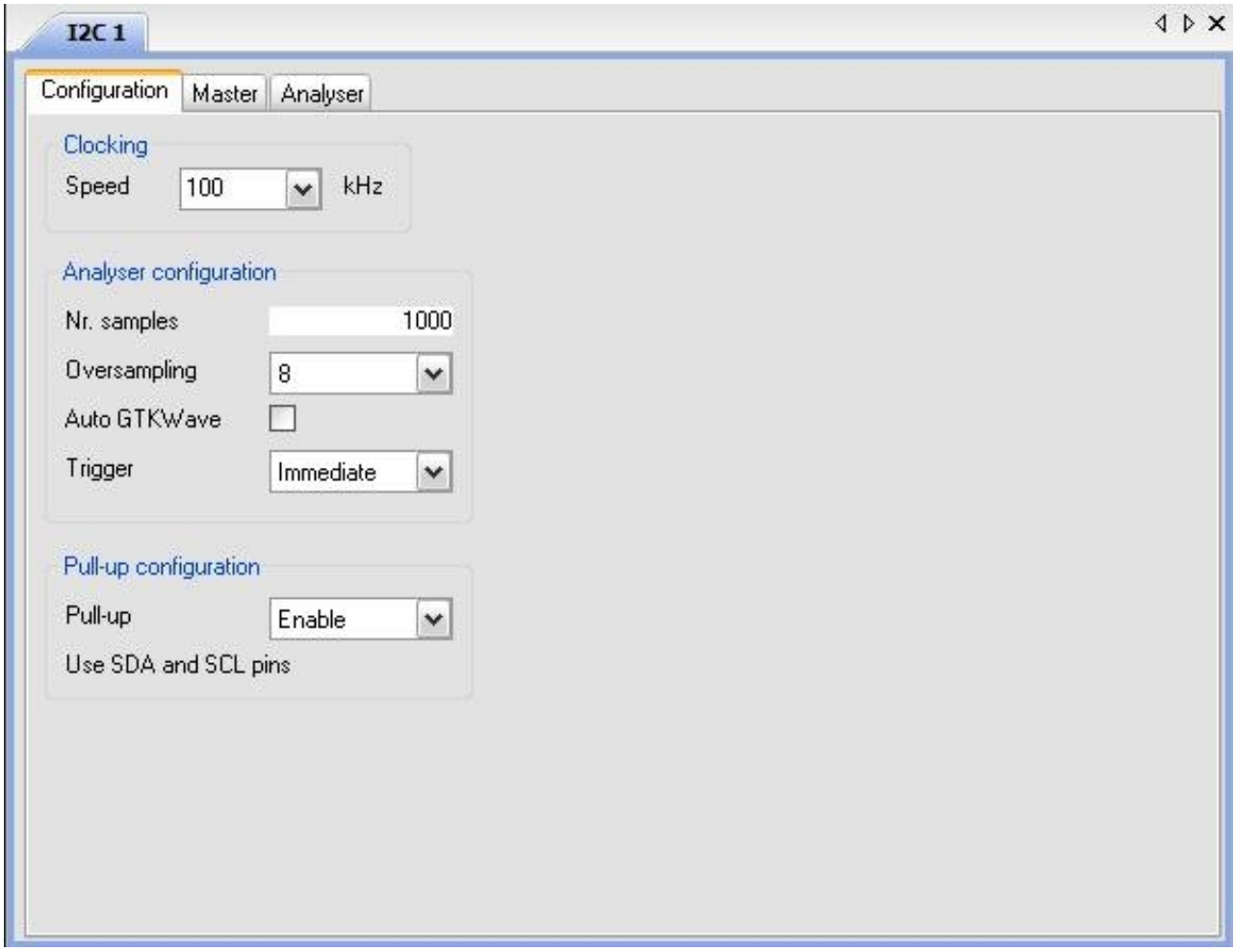
Nc	***0***	***1***	Nc
GND	**2**	**3**	GND
***SCL without pull-up* - D0**	**4**	**5**	**D1 – *SDA without pull-up***
Nc	**6**	**7**	Nc
Nc	**8**	**9**	Nc
Nc	**10**	**11**	Nc
GND	**12**	**13**	GND
Nc	**14**	**15**	Nc
Nc	**16**	**17**	Nc
Nc	**18**	**19**	Nc
Nc	**20**	**21**	Nc
GND	**22**	**23**	GND
Nc	**24**	**25**	Nc
Nc	**26**	**27**	Nc
Nc	**28**	**29**	Nc
***with pull-up* - SCL**	**30**	**31**	**SDA - *with pull-up***
GND	**32**	**33**	GND

8.3 I2C Application Page

The I2C mode of operation can be opened using the *File* > *New* application menu and then by selecting *I2C* in the dialog. Figure 44 gives an overview of the I2C page. The page can also be opened using the *New Page* pane. If the pane is not visible go to menu *View* > *New Page Pane*. When opening an I2C configuration file using the *File* > *Open* menu, a new page is automatically created.

8.4 Device configuration in I2C mode of operation

Figure 44 : I2C configuration tab



8.4.1.1 Clocking

SPEED

This drop-down menu allows choosing the I²C bus speed (bit rate / clock frequency). 10 kHz, 100 kHz, 400 kHz and 1000 kHz are available.

NR. SAMPLES

This field defines the number of samples to be collected during one run when using the device as I2C bus analyser. According to the bus speed and chosen oversampling, the device automatically selects the I2C analyser sampling frequency.

8.4.1.2 Analyser configuration

OVERSAMPLING

This drop-down menu allows choosing the oversampling level when using the device as a I²C bus analyser. According to the selected bus speed, the device automatically sets the sampling frequency. Available oversampling rates: 4 and 8.

AUTO GTK WAVE

☑ When checked, GTK Wave waveform viewer opens automatically at the end of the I²C analyser sampling run and the collected data is displayed.

☐ When not checked, GTK Wave waveform viewer does not automatically open at the end of the I²C analyser run.

TRIGGER

This drop-down menu defines the trigger mode used when using the device as a I²C bus analyser. Two modes are available: 'Immediate' – starts collecting data as soon as the user presses the 'Run' button (refer to the 'Analyser tab'). 'First Transfer – data sampling starts upon occurrence of the next I²C transfer (I²C bus START) after the 'Run' button (refer to 'Analyser tab') is pressed.

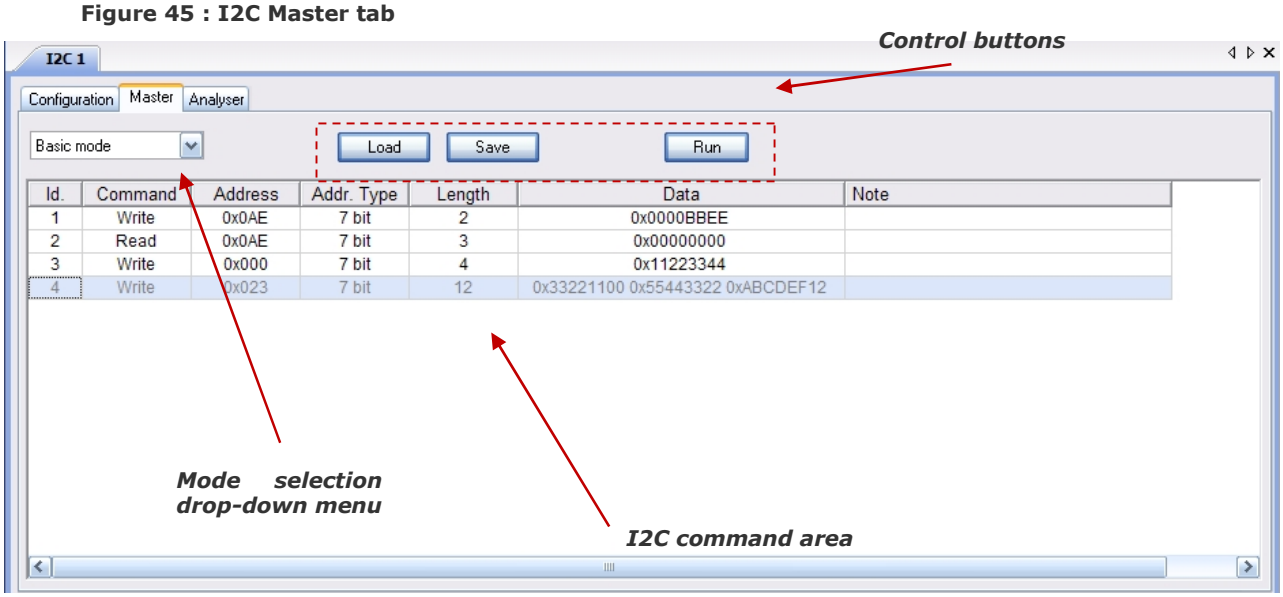
8.4.1.3 Pull-up configuration

PULL-UP

This drop-down menu allows enabling/disabling the pull-up resistors on the SDA/SCL pins. Actually, this selects from the 2 available I²C ports. With pull-ups enabled, the pins labeled SDA/SCL are used; when pull-ups are disabled, the D1/D0 pins are used. Please refer to the pin configuration section (8.2).

8.5 I2C Master

Figure 45 : I2C Master tab



MODE SELECTION

This drop-down menu allows choosing between 'Basic Mode' and 'Expert Mode'.
In 'Basic Mode', each I2C transaction is automatically started with a I²C START and ended with a I²C STOP.
The 'Expert Mode' allows selecting if I²C START and STOP have to be sent or not.

LOAD

This button allows loading a configuration and a set of transactions to be sent. Once pressed a browsing window opens to select the file.

SAVE

This button allows saving the configuration of the device in I2C mode into a file.

RUN

Executes the commands entered in the I2C command area.

I2C COMMAND AREA

Figure 46 : I2C Master command area

Id.	Command	Address	Addr. Type	Length	Data	Note	Start	Stop
1	Write	0x0AE	7 bit	2	0x0000BBEE		Yes	Yes
2	Read	0x0AE	7 bit	3	0x00000000		Yes	Yes
3	Write	0x000	7 bit	4	0x11223344		Yes	No
4	Write	0x023	7 bit	15	0x33221100 0x55443322 0xABCDEF12 0x00000000		Yes	Yes

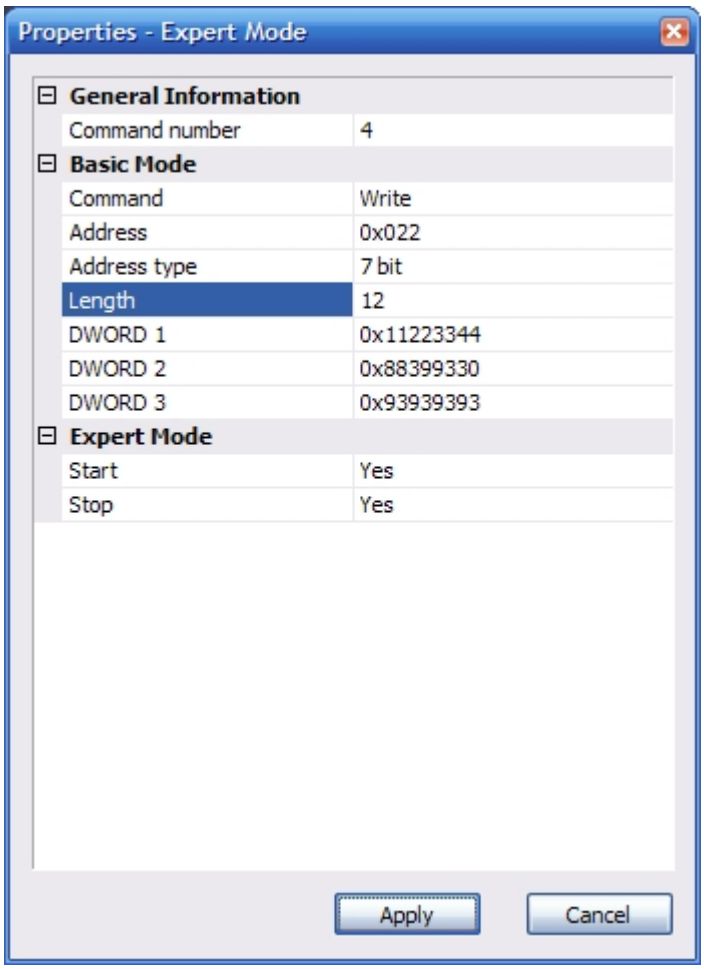
The command area presents the I²C transaction in a table.
Each table line corresponds to a single I²C transaction. Pressing the 'Run' button executes the set of I²C command entered in the command area line by line, from top to bottom.

Table 9 : I2C Command Area description

Id.	***Command***	***Address***	***Address Type***	***Length***	***Data***	***Note***	***Start***	***Stop***
I²C command reference number	*Write or Read*	*I²C slave address*	*7 bit or 10 bit address*	*Total transfer length in bytes*	*Data to write / Read data in groups of 32 bits.*	*Unused in master mode*	*Enables / disables the generation of a I²C START at the beginning of the I²C access.*	*Enables / disables the generation of a I²C STOP at the end of the I²C access.*

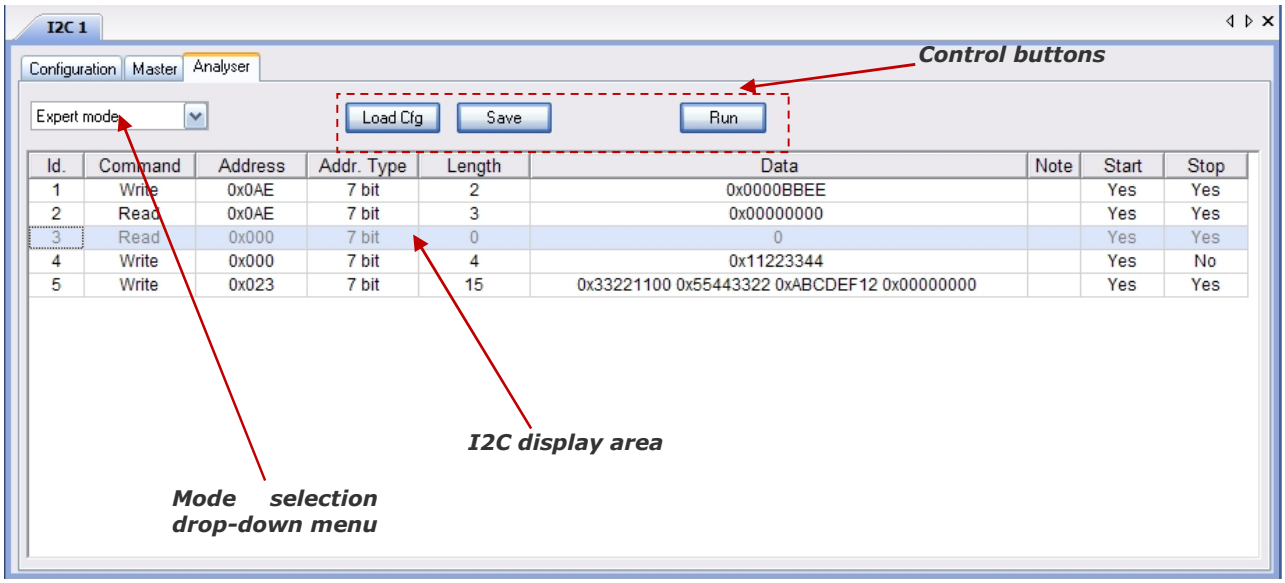
- Expert mode adds 2 columns to the Basic mode – 'START' and 'STOP' columns that allow enabling / disabling the generation of I²C START and I²C STOP conditions at the start and the end of the command, respectively.
- Right-click on the command area opens a context menu with the following options:
  - Insert Row >> inserts a new row in the table, at the cursor's position
  - Delete Row >> deletes the selected row from the table
  - Append Row >> inserts a new row after the cursor's position
  - Properties >> opens the properties window, that allows editing the selected I²C command.
- Shortcuts:
  - Insert row: I
  - Delete row: D
- The **Properties window** (refer to Figure 47) allows editing the command fields.

Figure 47 : I2C Properties window



8.6 I2C Analyser

Figure 48 : I2C Analyser tab



The I2C Analyser tab is pretty similar to the I2C Master tab (refer to Figure 45). It is used as a display / save only, to show the sampled I2C transactions and save them to an external file.

MODE SELECTION

This drop-down menu allows choosing between 'Basic Mode' and 'Expert Mode'.
The 'Expert Mode' allows showing if I²C START and STOP have been seen.

LOAD CFG

This button allows loading a configuration. Once pressed a browsing window opens to select the file.

SAVE

This button allows saving the sampled I²C transaction and configuration into a file. If a transaction is not complete, it is saved as comment. This mechanism allows using a file saved with the I²C analyser as a script for the I²C master. The data can also be saved as raw data and in VCD (value change dump) format. In such case, only the raw data is saved (no I²C decoding).

RUN

Runs the device in I2C analyser mode; once pressed, the device waits until occurrence of the defined trigger and then samples the specified number of data. Decoded I2C transactions are presented in the I2C display area.

I2C DISPLAY AREA

Table area showing the decoded I2C transactions. Only fully decoded transactions are shown here. For signal-level display, use the GTK Wave waveform viewer.

If an access is incomplete, it is underlined and more information is provided into the Note column, with the reason why the access was decoded as incomplete.

8.7 I2C Configuration / Data file format

The figure below gives an example of a data and configuration file for the I2C mode of operation.

```
# Define the clock frequency
@freq 100000

# Define the number of samples (analyser only)
@samples 1000

# Define the oversampling (analyser only)
@oversampling 8

# Define the trigger type (analyser only)
@trigtype immediate

# I2C transfers

# Transfer 0
@i2c_begin
@i2c_cmd       read
@i2c_addr      0x223
@i2c_addr_type 10
@i2c_length    0
@i2c_sta       1
@i2c_sto       1
@i2c_end

# Transfer 1
@i2c_begin
@i2c_cmd       write
@i2c_addr      0x02a
@i2c_addr_type 7
@i2c_length    2
@i2c_sta       1
@i2c_sto       1
@i2c_data      0x0000ffffff85ffffff96
@i2c_end

# Transfer 2
@i2c_begin
@i2c_cmd       read
@i2c_addr      0x008
@i2c_addr_type 7
@i2c_length    1
@i2c_sta       1
@i2c_sto       1
@i2c_end
```

The file syntax is described hereafter:
this character marks a comment line.

@freq
Configuration item – defines the chosen bus frequency.

@samples
Configuration item – defines the number of samples to collect (I²C analyser only).

@oversampling
Configuration item – defines the chosen oversampling for the I²C analyser.

@trigtype
Configuration item – defines the trigger type for the I²C analyser.

@i2c_begin
Marks the beginning of a single I²C transfer.

@i2c_cmd read
Defines whether the command to be sent (I2C master) or the decoded command (I2C analyser) is a read or a write operation.

@i2c_addr 0x223
Slave address for the transfer.

@i2c_addr_type 10
Slave address type for the transfer: 7 = 7 bit address; 10 = 10 bit address.

@i2c_length 8
Length of the data transfer in bytes.

@i2c_sta 1
Marks whether the START condition should be generated (I2C Master) or has been detected (I2C analyser). 1 : START active; 0 : START inactive.

@i2c_sto 1
Marks whether the STOP condition should be generated (I2C Master) or has been detected (I2C analyser). 1 : STOP active; 0 : STOP inactive.

@i2c_end
Marks the end of a I²C transfer.

8.8 Additional remarks about the generation / decoding of the I²C protocol.

GP Series and I²C Xpress devices used as a I²C master are of purpose to generate valid I²C traffic onto an I²C interconnect.

Each I²C transfer defined with the **I²C Master** is composed of the following phases:

- Optional START generation
- SLAVE ADDRESS is sent on the bus, followed by the OPERATION (READ / WRITE)
- DATA GENERATION / SAMPLING, with NACK checking
- Proper NACK generation in case of read operation.

- Optional STOP generation

Moreover, the tool returns error cases and interrupts transfer when:

- Arbitration lost – that is:
  - when a STOP signal is detected but not requested
  - when the master drives SDA high but SDA is low.
- No ACK received from a slave

The I²C Analyser samples the I²C bus like a logic analyzer and offers a I²C decoding capability. The I²C analyser is able to provide a fully decoded transfer if:

- START condition is generated.
- The transfer is compliant with the I²C protocol.

Incomplete transfers are reported with comments about why there were sampled as incomplete.

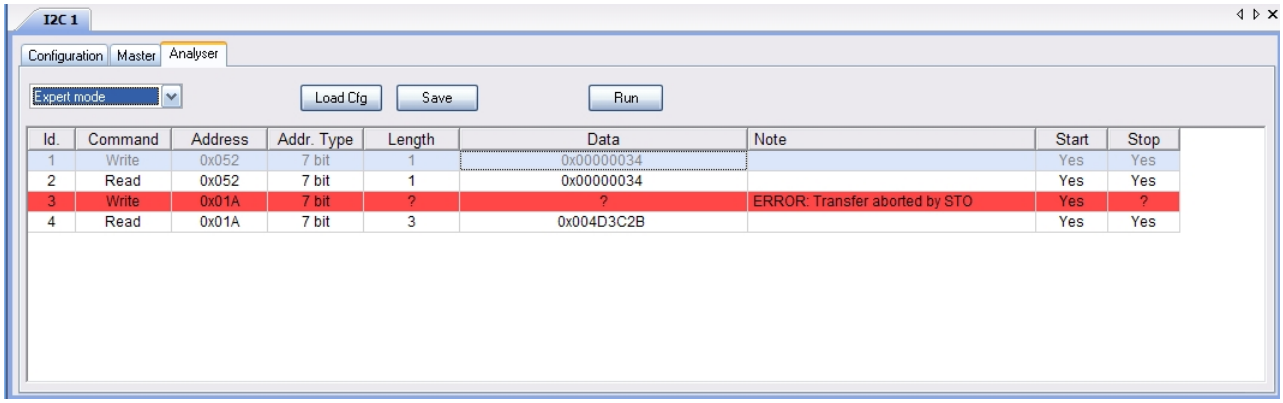


Figure 49 : I²C analyser with error in transfer

Possible sources of error (marked in RED):

- A START condition is following directly by a STOP condition;
- A transfer is interrupted by a START or a STOP condition.

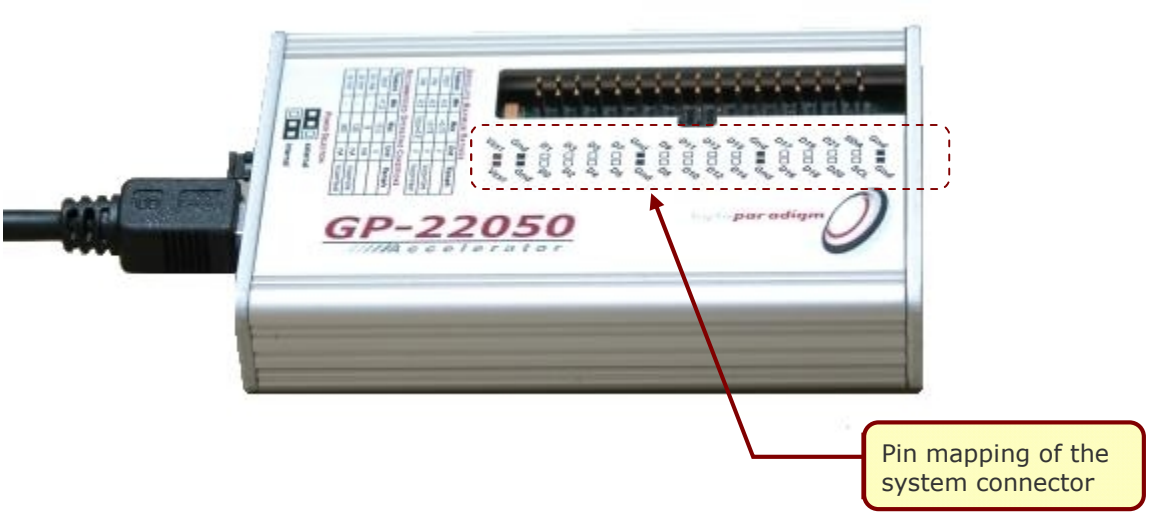
Possible sources of warning (marked in GREEN):

- Device address is not acknowledged (no device answers to the address call);
- Data is not acknowledged during write;
- SCL toggled without SDA toggling (activity on SCL without activity on SDA).

9 Appendix A: Devices Connectors

For GP and Xpress Series, the pin mapping of the connector is always described on the top side of the device (Refer to Figure 50 for an example with the GP-22050). For more details about the system connector, refer to the device data sheet.

Figure 50: View of the GP Series device top side



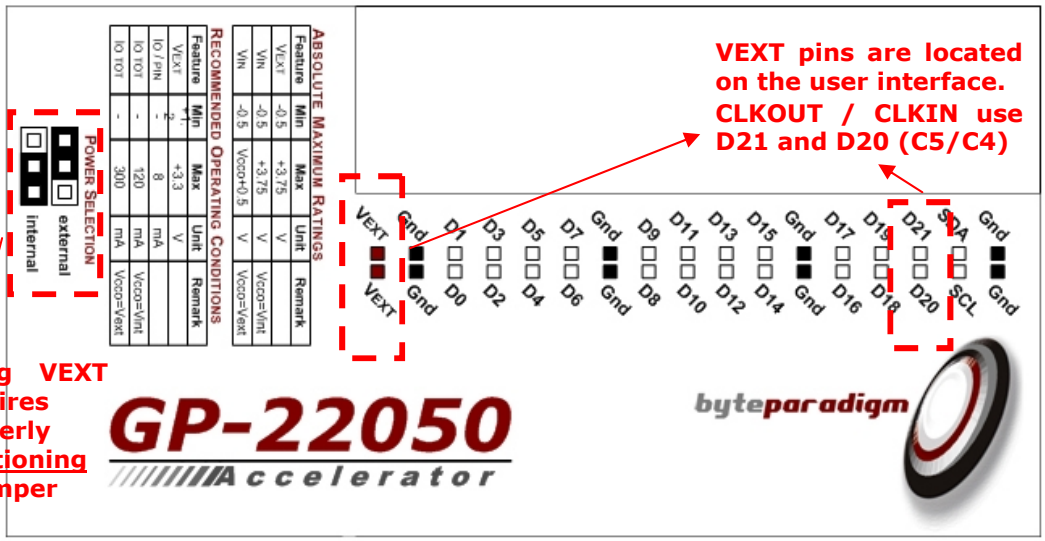
9.1 Check hardware revision

Check serial number located at the back of the device with a barcode, to find the hardware revision.

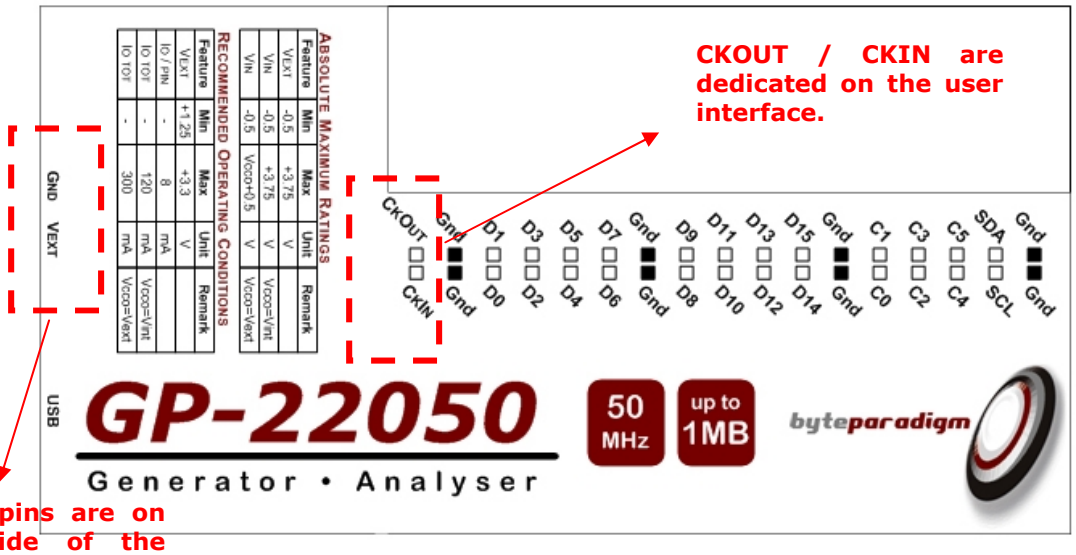
According to the hardware revision, the following elements vary:

Hardware revision 2 first digits	***In/out clocks on user interface connector***	***External power source selection***
01	**ClkOut** pin is located on pin labelled D21 / C5 **ClkIn** pin is located on pin labelled D20/C4	Using external power source for I/Os requires properly **positioning** the power selection jumper AND applying the external voltage onto the VEXT pins of the user interface connector
02	**ClkOut** pin is dedicated to output clock and is labelled CkOut (GP / Wave Generator Xpress) or SCLKo (SPI Xpress) **ClkIn** pin is dedicated to input clock and is labelled CkIn	Using external power source for I/Os requires its application on the side power source connector after having **removed** the jumper.
03		

HARDWARE REVISION STARTS WITH "01"



HARDWARE REVISION STARTS WITH "02" OR "03"



9.2 Connecting the GP or Xpress device

9.2.1 GP / Xpress device Connectors

The GP-22050 is taken as an example below. Xpress device have the same connectors.

Figure 51: GP-22050 Top View – hardware revision starting with 01

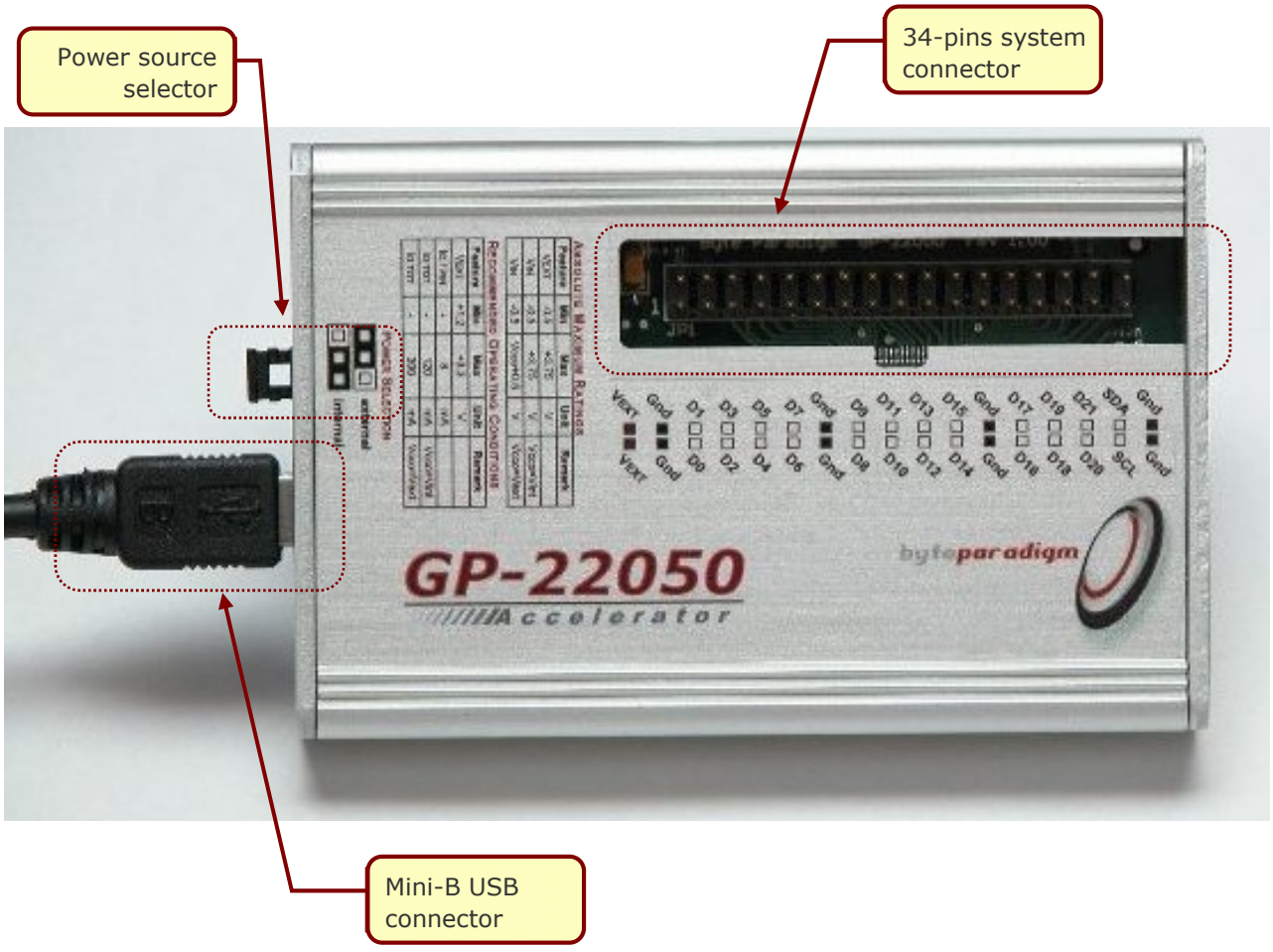
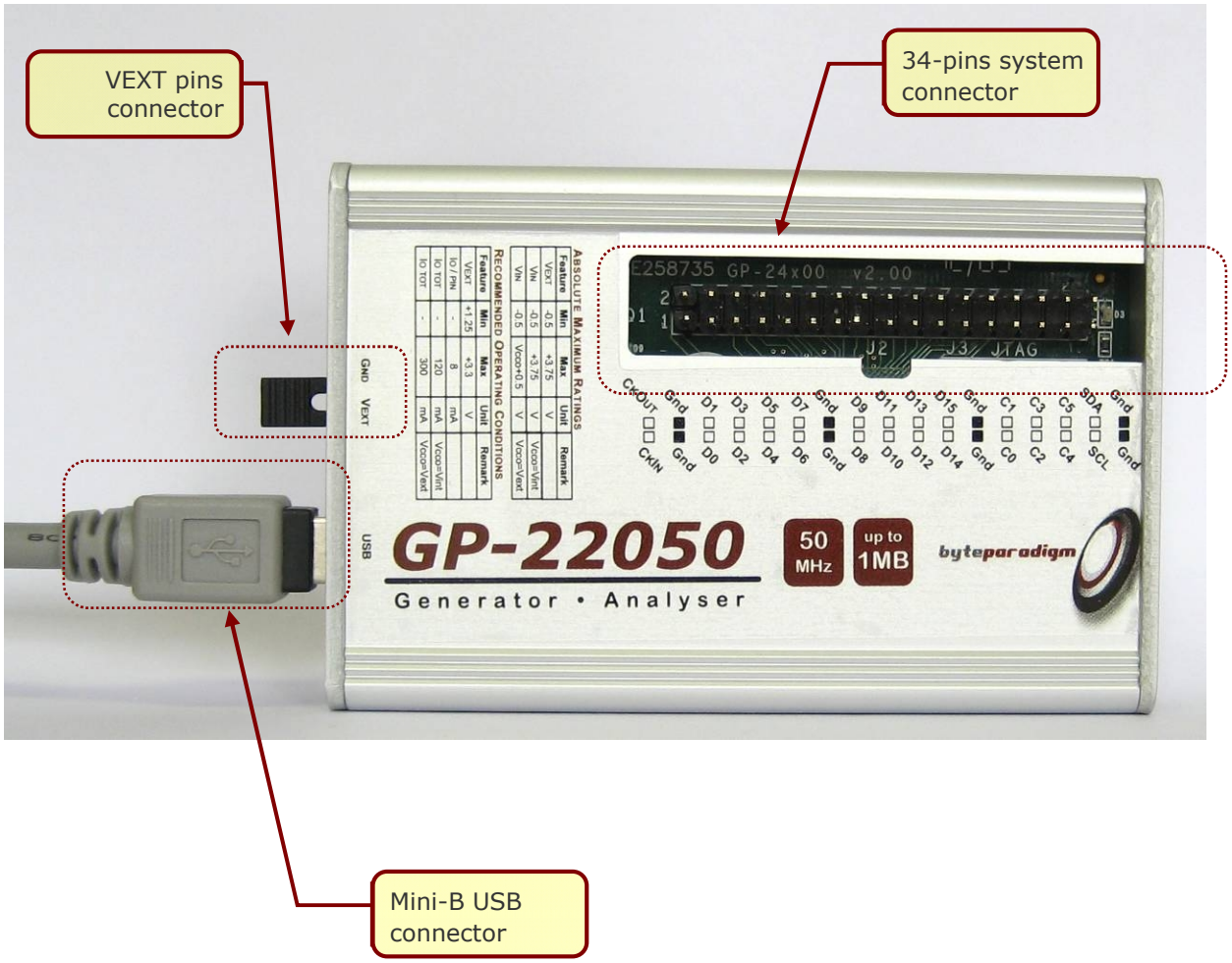
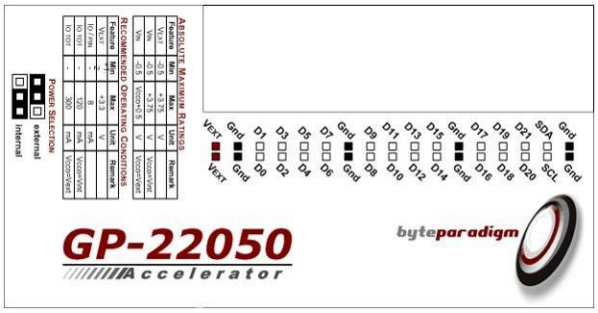
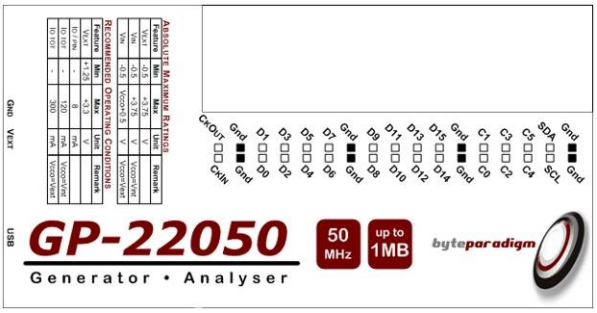
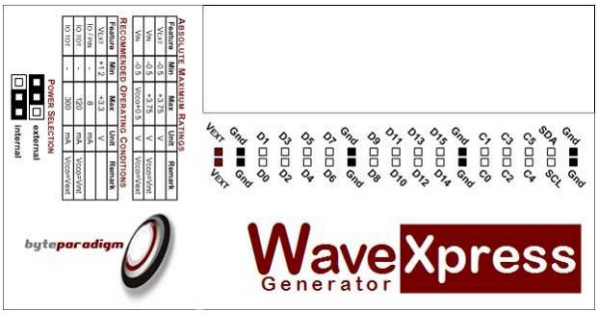
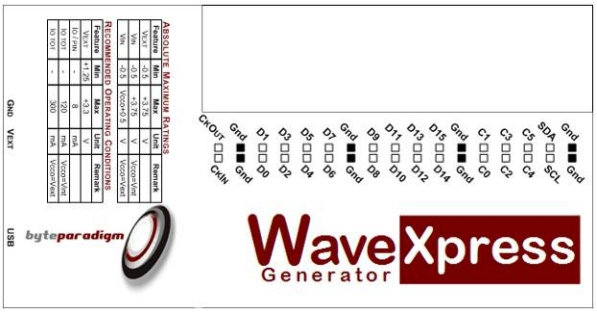
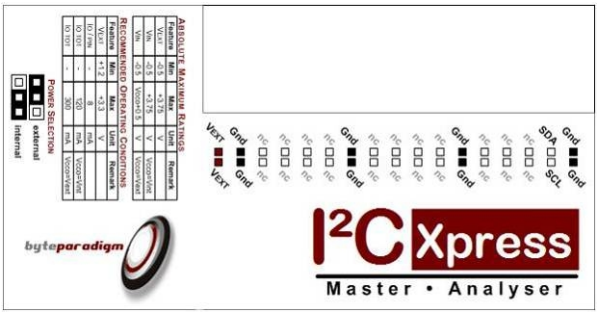
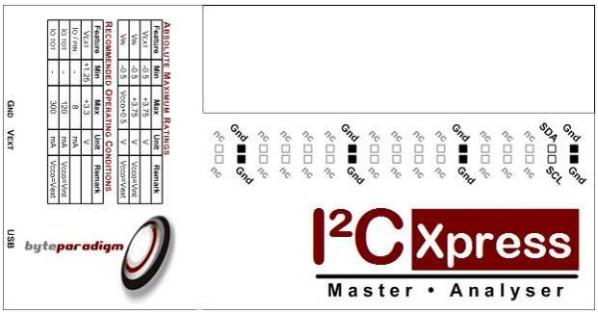
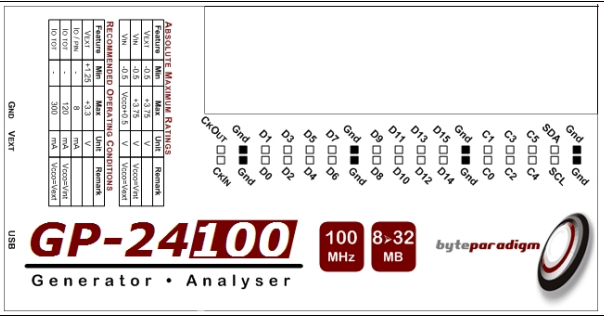


Figure 52: GP-22050 Top View – hardware revision starting with 02 or 03



The table below shows the front labels of the GP / Xpress devices according to their hardware revision.

Hardware revision starting with 01	*Hardware revision starting with 02 or 03*
	
	
	
	
	

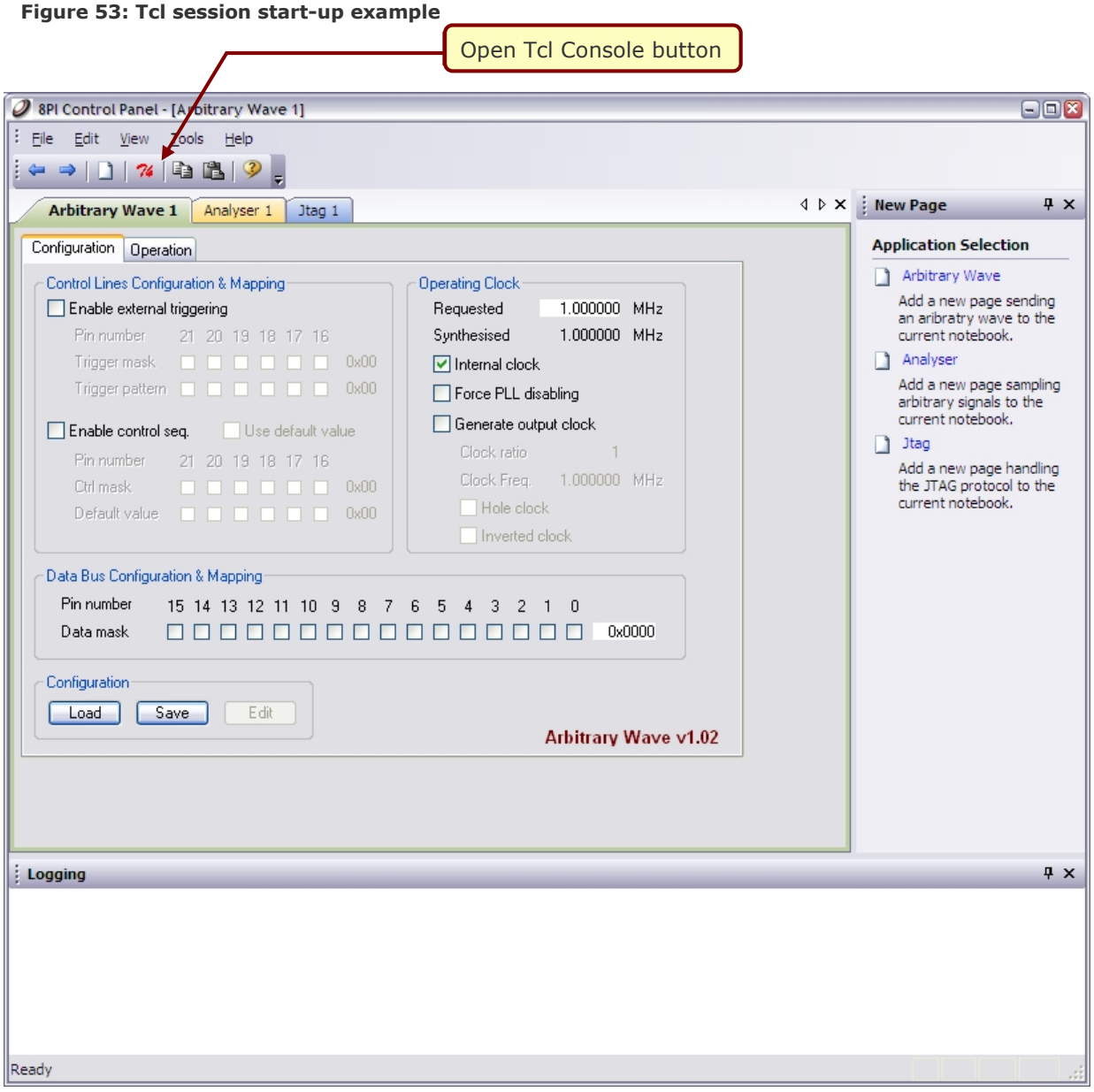
10 Appendix B: TCL Console

Aside of controlling the GP Series device using the graphical interface of the *8PI control Panel* application, the user can start a TCL console to execute scripts. Refer to the different TCL library guides [2], [4], [6] and [8] for more details on using the TCL scripting interface.

10.1 Starting a TCL session from the 8PI Control Panel GUI

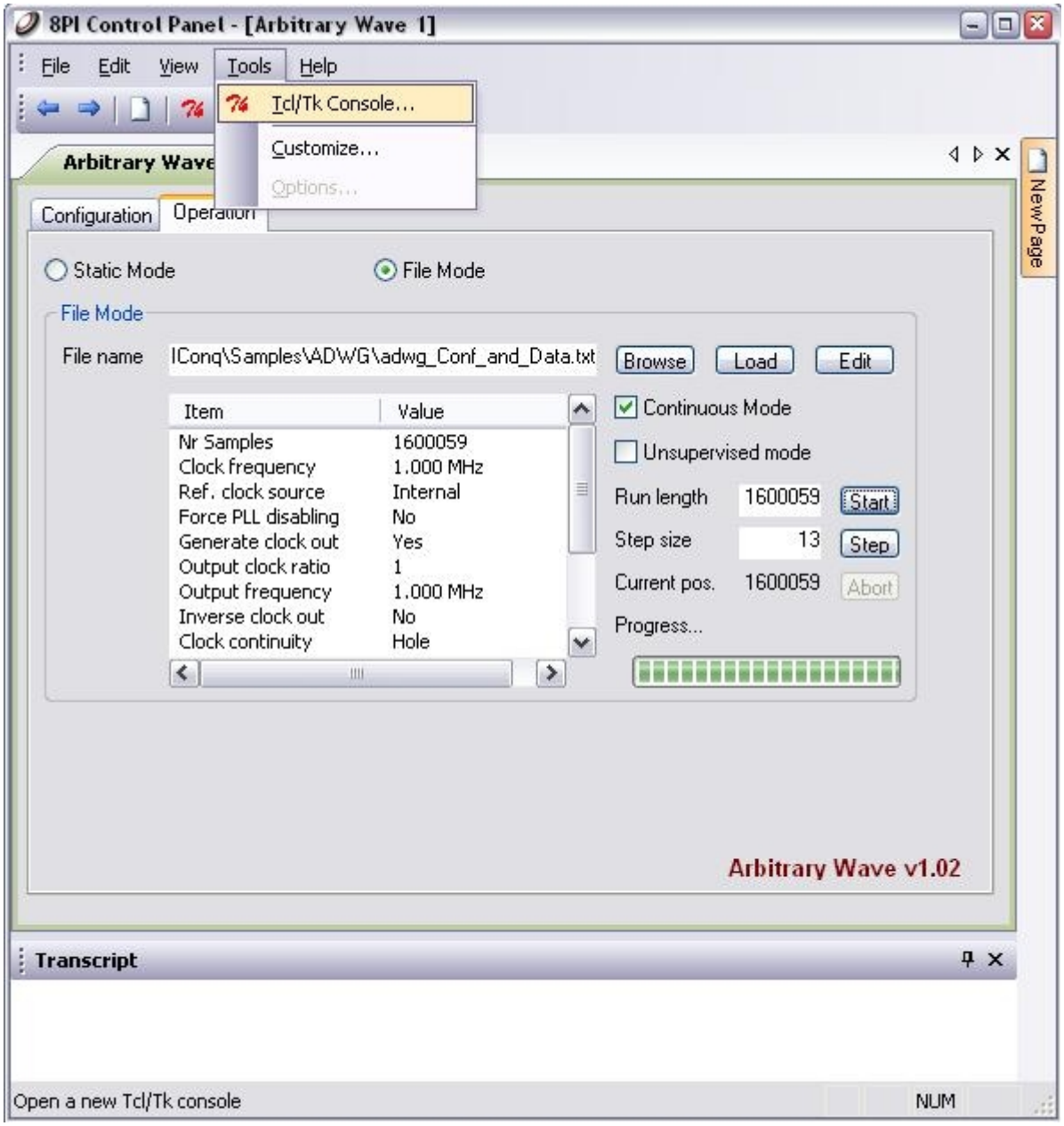
From the 8PI Control Panel GUI, access the desired operating mode sheet (Arbitrary Wave, Analyser, JTAG or SVF). Then click on the 'Open Tcl Console' button (Figure 53). This opens the Tcl console, loads the Tcl libraries relative to the chosen operating mode and initialises the Tcl session.

Figure 53: Tcl session start-up example



An alternative to the button in the shortcut bar is to use the *Tools* menu as depicted on Figure 54. This menu has exactly the same behaviour as the shortcut button.

Figure 54: Opening a TCL console from the tools menu

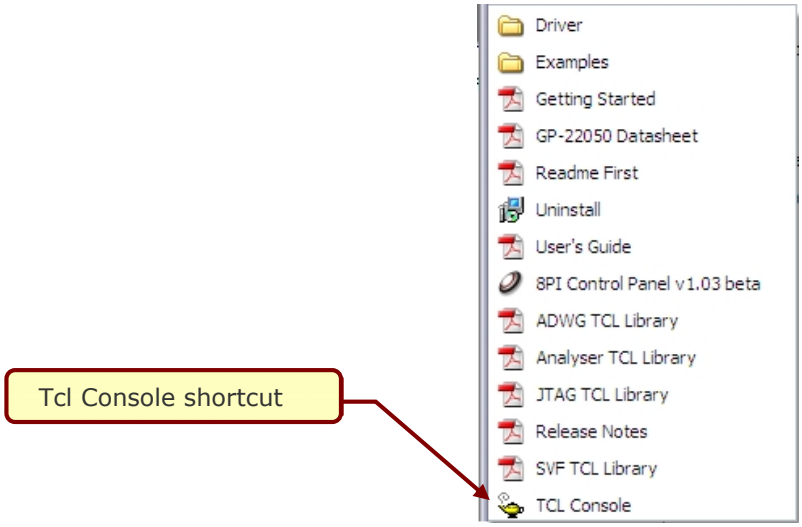


10.2 Starting a stand-alone TCL session

A stand-alone TCL session can be started without using the graphical interface of the *8PI Control Panel* application. To start a stand-alone TCL console, go to the GP Series device program group using the Windows *Start* menu. From the *Byte Paradigm > 8PI Control Panel* program group, click on the *TCL Console* shortcut. This starts the *Wish84* TCL interpreter with the *tkcon* console. As no application is running, this console is operating in stand-alone mode and no operating modes are configured. Before controlling the GP Series device, an operating mode must be initialised by loading the corresponding TCL library. Type the command line corresponding to the requested operating mode.

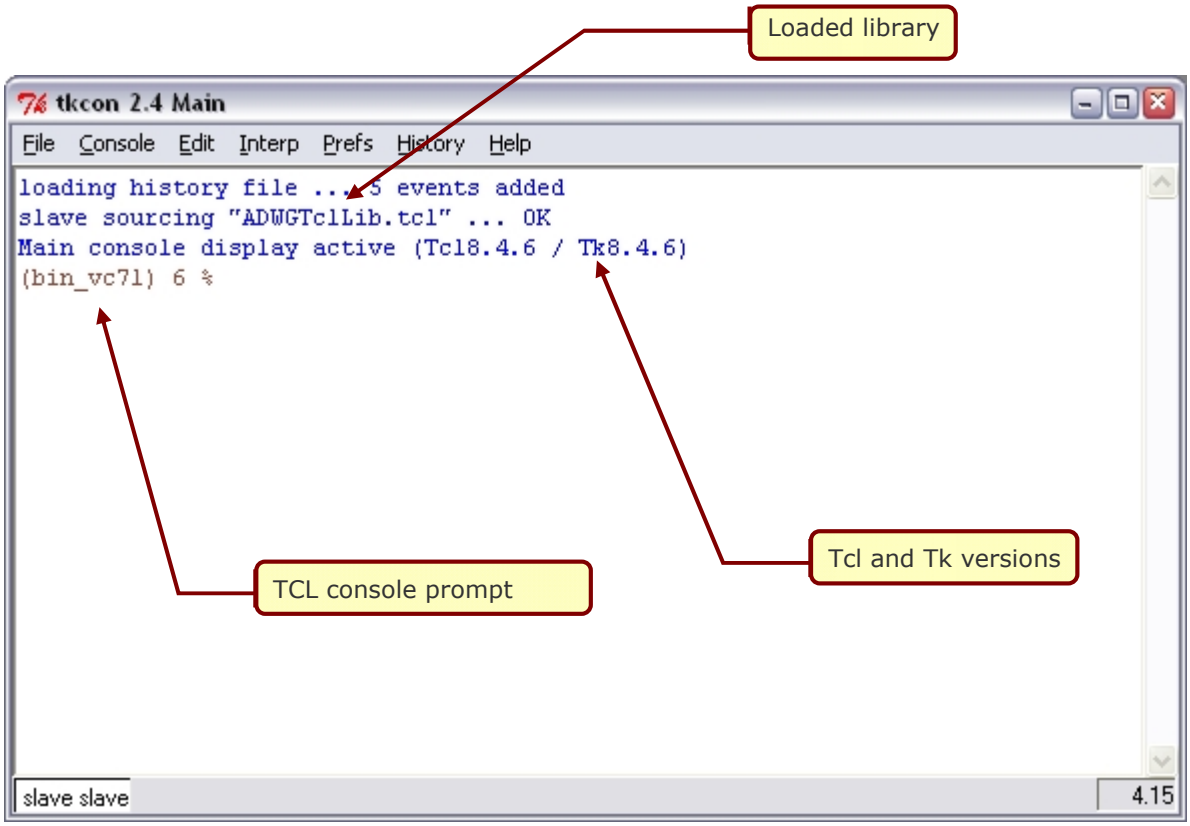
ADWG	`% source ADWGTclLib.tcl`
Analyser	`% source AnalyserTclLib.tcl`
JTAG	`% source JTAGITclLib.tcl`
SVF	`% source SVFTclLib.tcl`

Figure 55: TCL console shortcut in 8PI Control Panel program group



By default, the GP Series device software environment uses the WISH interpreter with the TKCON console (interactive mode) (Figure 56). For more information about the TKCON console, please check the following links: http://tkcon.sourceforge.net/ - http://wiki.tcl.tk/1878.

Figure 56: Tcl console at startup



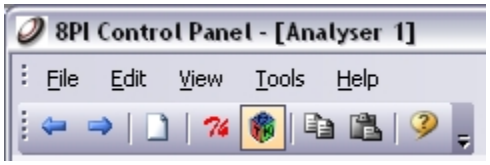
11 Appendix C: Wave Viewer

A wave viewer developed by GTKWave is integrated in the *8PI Control Panel*. This viewer can be used to display any VCD file. The viewer can be started from the menu *Tools* > *GTKWave*, or using the *GTKWave* button in the toolbar.

Figure 57: Wave viewer menu



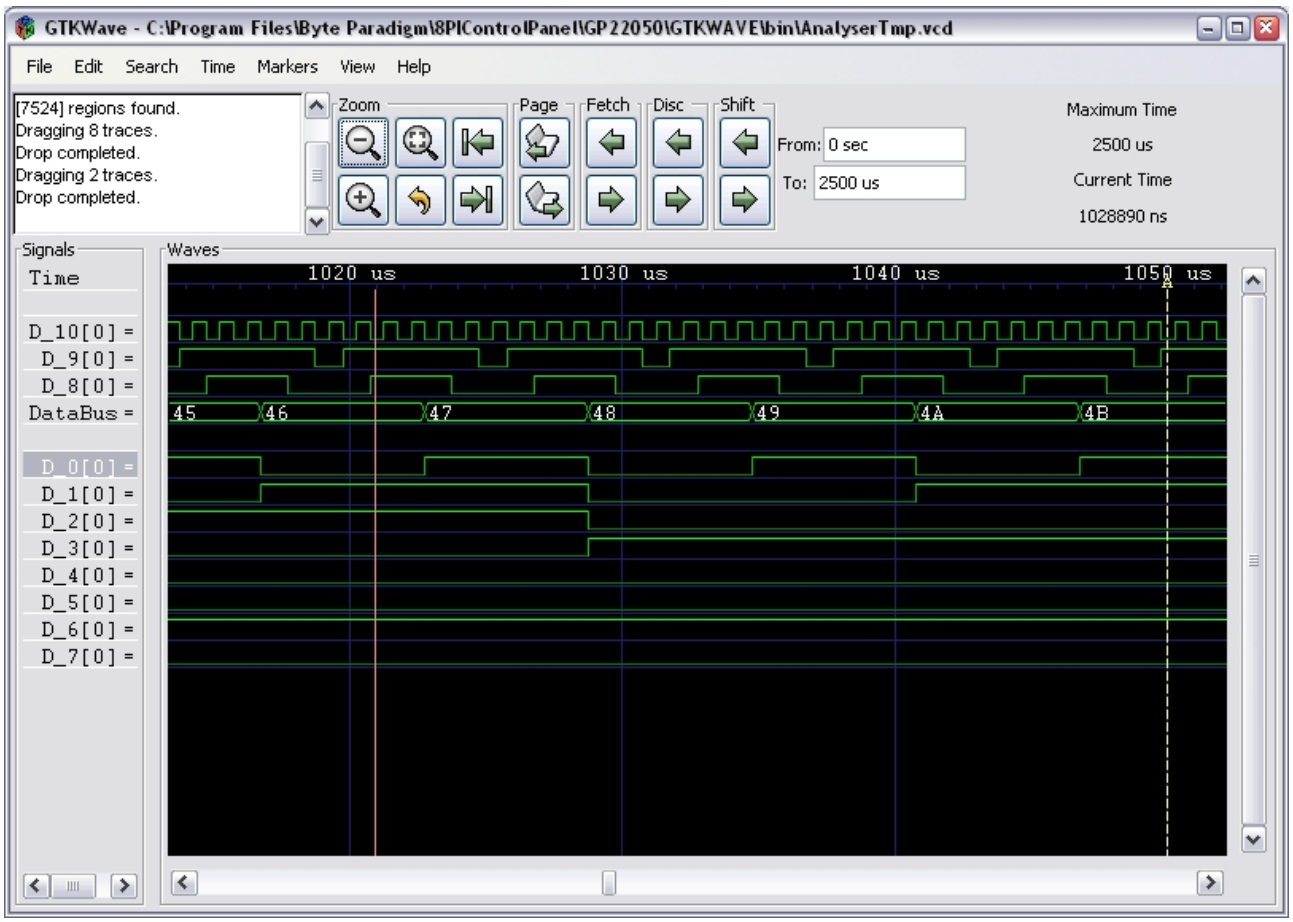
Figure 58: Wave viewer button



Even if the wave viewer is intended to be used for the analyser mode, it is possible to start it when any application is used (ADWG, JTAG, etc.). When the wave viewer is started and the current application is not the analyser, then the user is asked to provide a VCD file and a configuration file to be displayed. When the viewer is started from the analyser mode tab, the captured samples are displayed automatically.

> *In analyser mode, the captured signals can be displayed in the wave viewer independently of the export file format.*

Figure 59: GTKWave viewer window



12 Appendix E – Overview of SPI accesses, options and parameters

Figure 60 : Typical SPI access

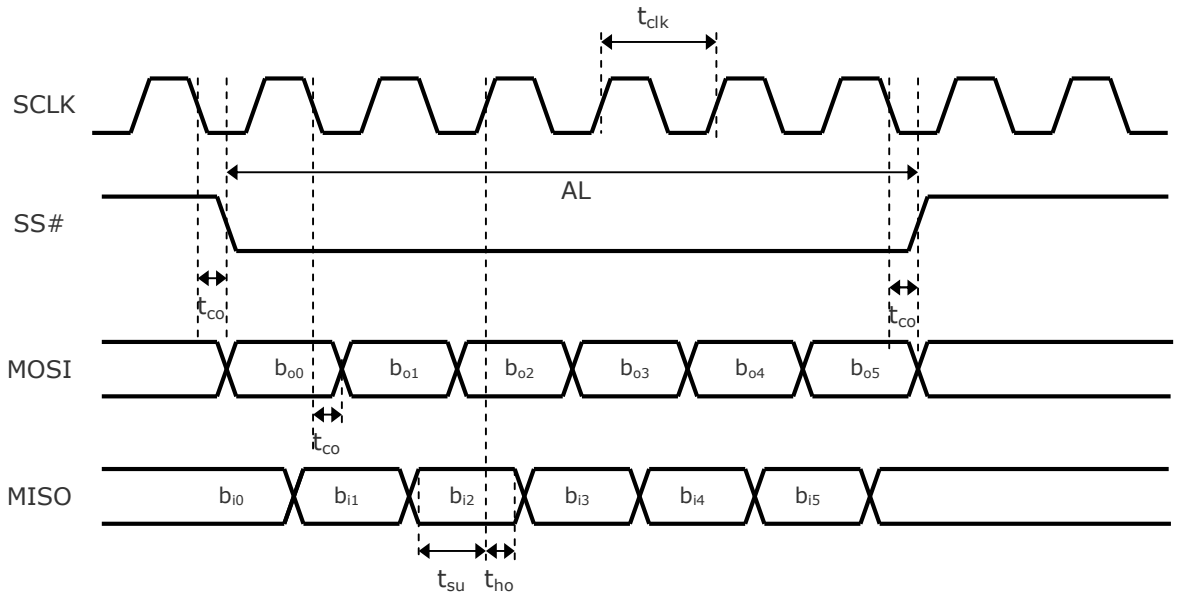


Figure 61 : Master SCLK and SS# generation options

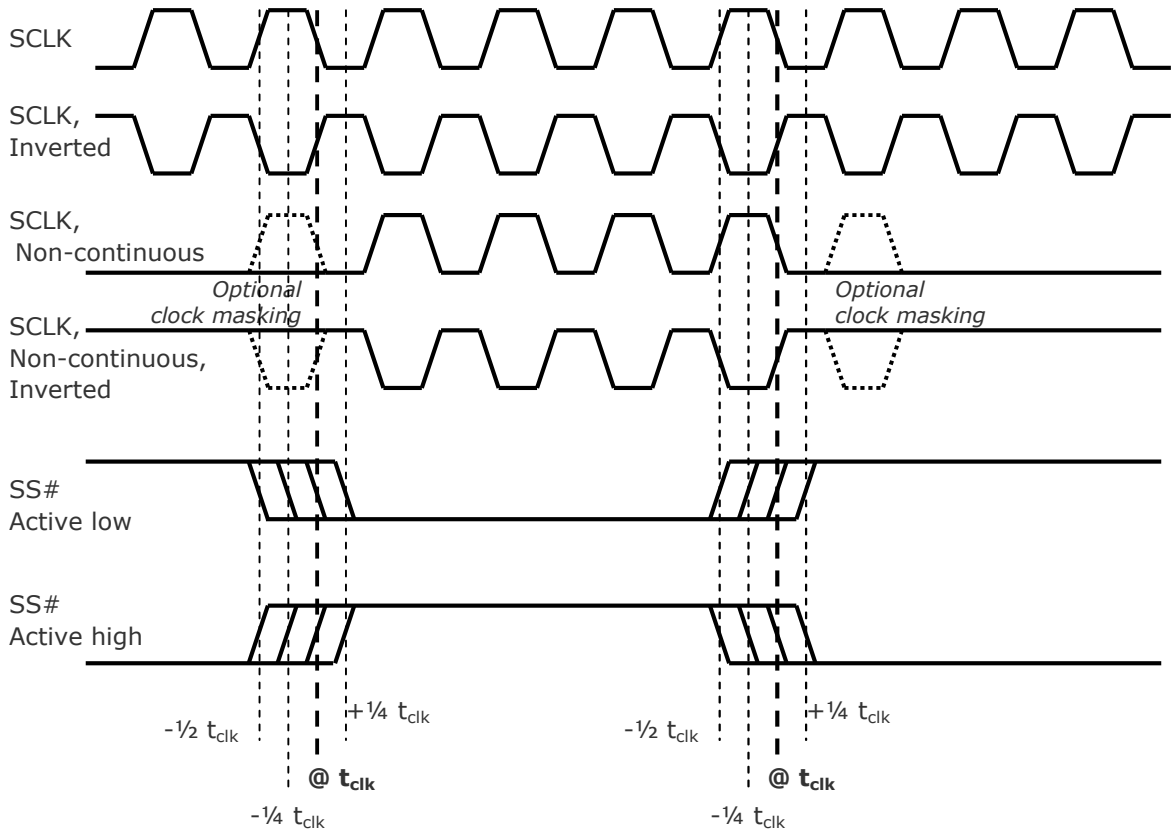


Figure 62 : 3 wires protocols overview & clock options

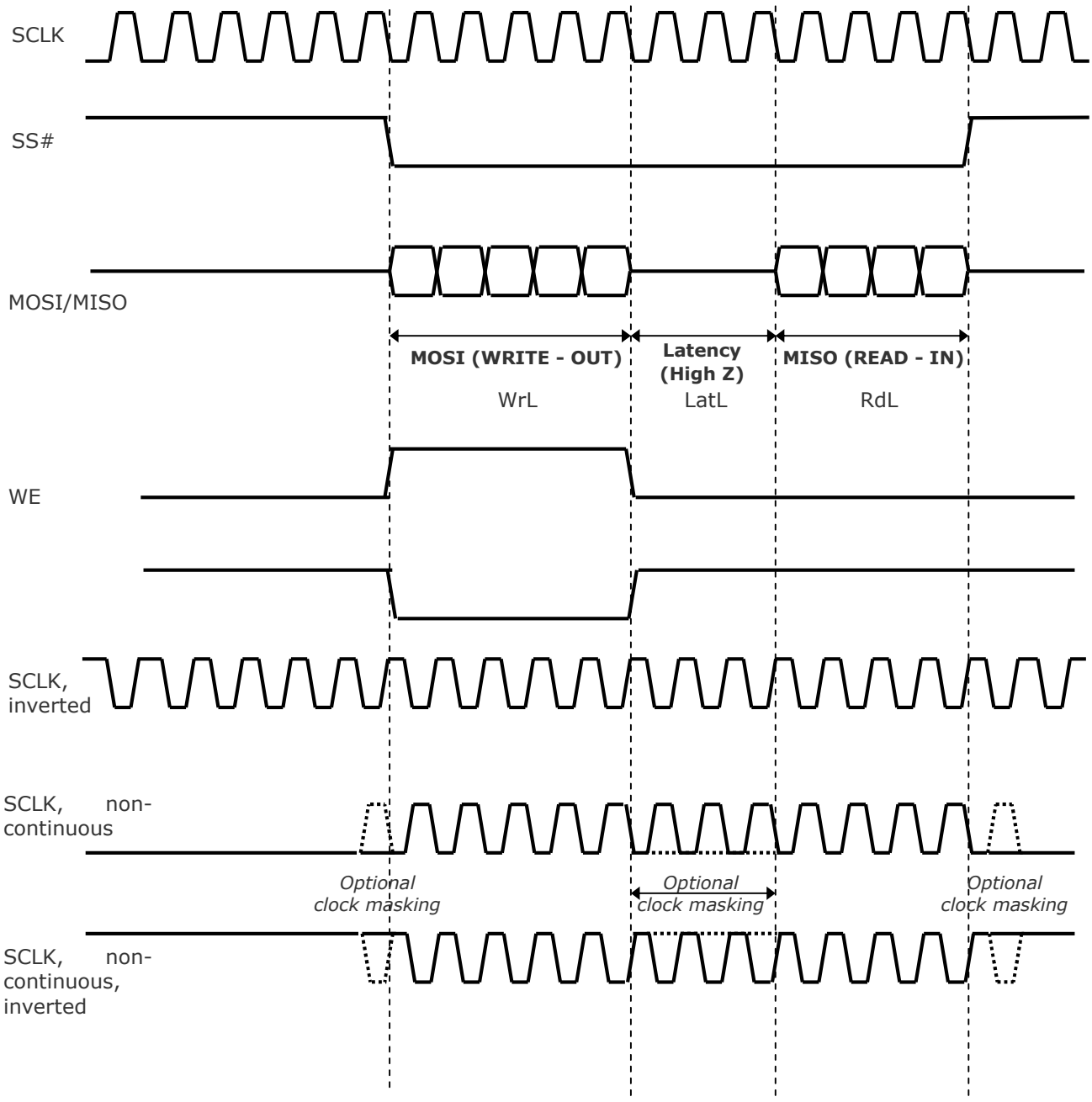


Table 10 : SPI parameters

SPI 4				
Symbol	***Parameter***	***Min***	***Max***	***Units***
t_{clk}	Clock period	20	1,250,000	ns
t_{co}	Clock to output	0	4	ns
t_{su}	Setup time	5.6	-	ns
t_{ho}	Hold time	-	0.0	ns
AL	Access length	1	see tables below	bits

SPI 3				
Symbol	***Parameter***	***Min***	***Max***	***Units***
WrL	Write length	1	see table below	bits
LatL	Write to Read latency length	0		bits
RdL	Read length	0		bits

		SPI 4 accesses	**SPI 3 accesses**		
SS start delay	**SS end delay**	**Max. AL**	**Max. WrL**	**Max. LatL**	**Max. RdL**
(don't care)	-1/4 SCLK	8.000 bits	1.023 bits	400 bits	1.023 bits
(don't care)	+1/4 SCLK	8.000 bits	1.023 bits		1.023 bits
-1/2 SCLK	-1/2 SCLK	16.000 bits	2.047 bits		2.047 bits
-1/4 SCLK	(don't care)	8.000 bits	1.023 bits		1.023 bits
+1/4 SCLK	(don't care)	8.000 bits	1.023 bits		1.023 bits
no delay		**32.000 bits**	**4.095 bits**		**4.095 bits**

Delay on SS (START or STOP)	**Max. SCLK frequency**
0	50 MHz
+/- ½ SCLK	25 MHz
+/- ¼ SCLK	12.5 MHz