THORLABS

PAX1000
Write Your Own Application

- Drivers
- Samples
- Programmer's Reference

Version: 1.3
Date: 19-Oct-2022

1 Write Your Own Application

In order to write your own application, you need a specific instrument driver and some tools for use in different programming environments. The driver and tools are included in the installer package and cannot be found as separate files on the installation CD.

In this section the location of drivers and files, required for programming in different environments, are given for installation under Windows 7, Windows 8.1 and Windows 10 (32 and 64 bit)

Note

PAX1000 software and drivers contains 32 bit and 64 bit applications.

In 32 bit systems, only the 32 bit components are installed to

`C:\Program Files\`…

In 64 bit systems the 64 bit components are being installed to

`C:\Program Files\`…

while 32 bit components can be found at

`C:\Program Files (x86)\`…

In the table below you will find a summary of what files you need for particular programming environments.

Programming environment	Necessary files
C, C++, CVI	*.fp (function panel file; CVI IDE only) *.h (header file) *.lib (static library) *.dll (dynamic linked library)
C#	.net wrapper dll
Visual Studio	*.h (header file) *.lib (static library) or .net wrapper dll
LabView	*.fp (function panel) and NI VISA instrument driver Beside that, LabVIEW driver vi's are provided with the *.llb container file

Note

All above environments require also the NI VISA instrument driver dll !

During NI-VISA Runtime installation, a system environment variable `VXIPNPPATH` for including files is created. It contains the information where the drivers are installed to, usually to `C:\Program Files\IVI Foundation\VISA\WinNT\.`

This is the reason, why after installation of a NI-VISA Runtime a system reboot is required: This environment variable is necessary for installation of the instrument driver software components.

In the next sections the location of above files is described in detail.

© 2022 Thorlabs GmbH

1.1 32 Bit Operating Systems

Note

According to the VPP6 (Rev6.1) Standard the installation of the 32 bit VXIpnp driver includes both the WINNT and GWINNT frameworks.

VXIpnp Instrument driver:

`C:\Program Files\IVI Foundation\VISA\WinNT\Bin\TLPAX_32.dll`

Note

This instrument driver is required for all development environments!

Header file

`C:\Program Files\IVI Foundation\VISA\WinNT\include\TLPAX.h`

Static Library

`C:\Program Files\IVI Foundation\VISA\WinNT\lib\msc\TLPAX_32.lib`

Function Panel

`C:\Program Files\IVI Foundation\VISA\WinNT\TLPAX\TLPAX.fp`

Online Help for VXIpnp Instrument driver:

`C:\Program Files\IVI Foundation\VISA\WinNT\TLPAX\Manual\TLPAX.html`

NI LabVIEW driver (including an example VI)

The LabVIEW Driver is a 32 bit driver and compatible with 32bit NI-LabVIEW versions 8.5 and higher only.

`C:\Program Files\National Instruments\LabVIEW xxxx\instr.lib\TLPAX…`
`…\TLPAX.llb`

(LabVIEW container file with driver vi's and an example. "LabVIEW xxxx" stands for actual LabVIEW installation folder.)

.net wrapper dll

`C:\Program Files\Microsoft.NET\Primary Interop Assemblies…`
`…\Thorlabs.PAX1000.Interop.dll`

`C:\Program Files\IVI Foundation\VISA\VisaCom\Primary Interop…`
`…Assemblies\Thorlabs.PAX1000.Interop.dll`

Example for C

Source file:

`C:\Program Files\IVI Foundation\VISA\WinNT\TLPAX\Examples\CSample\…`
`…sample.c`

(C program how to communicate with a PAX1000 Series polarimeter)

Example for C#

Solution file:

`C:\Program Files\IVI Foundation\VISA\WinNT\TLPAX\Examples…`
`…\CSharp\Thorlabs.PAX_CSharp_Demo.sln`

© 2022 Thorlabs GmbH

Project file:

```
C:\Program Files\IVI Foundation\VISA\WinNT\TLPAX\Examples…
…\CSharp\Thorlabs.PAX_CSharp_Demo\Thorlabs.PAX_CSharp_Demo.csproj
```

Executable sample demo:

```
C:\Program Files\IVI Foundation\VISA\WinNT\TLPAX\Examples…
…\CSharp\Thorlabs.PAX_CSharp_Demo\bin\Thorlabs.PAX_CSharp_Demo.exe
```

Example for LabView

Included in LabVIEW llb container.

© 2022 Thorlabs GmbH

1.2 64 Bit Operating Systems

Note

According to the VPP6 (Rev6.1) Standard the installation of the 64 bit VXIpnp driver includes the WINNT, WIN64, GWINNT and GWIN64 frameworks. That means, that the 64 bit driver includes the 32 bit driver as well.

In case of a 64 bit operating system, 64bit drivers and applications are installed to

“C:\Program Files”

while the 32 bit files - to

“C:\Program Files (x86)”

Below are listed both installation locations, so far applicable.

VXIpnp Instrument driver:

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\Bin\TLPAX_32.dll`

`C:\Program Files\IVI Foundation\VISA\Win64\Bin\TLPAX_64.dll`

Note

This instrument driver is required for all development environments!

Header file

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\include\TLPAX.h`

`C:\Program Files\IVI Foundation\VISA\Win64\include\TLPAX.h`

Static Library

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\lib\msc…`
`…\TLPAX_32.lib`

`C:\Program Files\IVI Foundation\VISA\Win64\Lib_x64\msc\TLPAX_64.lib`

Function Panel

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\TLPAX\TLPAX.fp`

Online Help for VXIpnp Instrument driver:

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\TLPAX\Manual…`
`…\TLPAX.html`

NI LabVIEW driver (including an example VI)

The LabVIEW Driver supports 32bit and 64bit NI-LabVIEW2009 and higher.

32 bit NI-Labview version

`C:\Program Files (x86)\National Instruments\LabVIEW xxxx\instr.lib…`
`…\TLPAX\TLPAX.llb`

64 bit NI-Labview version

`C:\Program Files\National Instruments\LabVIEW xxxx\instr.lib…`
`…\TLPAX\TLPAX.llb`

(LabVIEW container file with driver vi's and an example. "LabVIEW xxxx" stands for actual LabVIEW installation folder.)

© 2022 Thorlabs GmbH

.net wrapper dll

`C:\Program Files (x86)\Microsoft.NET\Primary Interop Assemblies…`
`…\Thorlabs.PAX1000.interop.dll`

`C:\Program Files (x86)\IVI Foundation\VISA\VisaCom\…`
`…Primary InteropAssemblies\Thorlabs.PAX1000.Interop.dll`

`C:\Program Files\IVI Foundation\VISA\VisaCom64\…`
`…\Primary Interop Assemblies\Thorlabs.PAX1000_64.Interop.dll`

Example for C

Source file:

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\TLPAX\Examples\…`

`…CSample\sample.c`

(C program how to communicate with a PAX1000 Series polarimeter)

Example for C#

Solution file:

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\TLPAX\Examples…`
`…\CSharp\Thorlabs.PAX_CSharp_Demo.sln`

Project file:

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\TLPAX\Examples…`
`…\CSharp\Thorlabs.PAX_CSharp_Demo\Thorlabs.PAX_CSharp_Demo.csproj`

Executable sample demo:

`C:\Program Files (x86)\IVI Foundation\VISA\WinNT\TLPAX\Examples…`
`…\CSharp\Thorlabs.PAX_CSharp_Demo\bin\Thorlabs.PAX_CSharp_Demo.exe`

Example for LabView

Included in LabVIEW llb container.

© 2022 Thorlabs GmbH

2 Programmers Reference

2.1 PAX1x Remote Control

IEEE488.2 Common Commands

Common commands are device commands that are common to all devices according to the IEEE488.2 standard. These commands are designed and defined by this standard. Most of the commands are described in detail in this section. The following common commands associated with the status structure are covered in the “Status Structure” section: *CLS, *ESE, *ESE?, *ESR?, *SRE, *SRE?, *STB?

2.1.1 Command Summary

Mnemonic	Name	Description
***CLS**	Clear status	Clears all event registers and Error Queue
***ESE <NRf>**	Event enable command	Sets the Standard Event Enable Register
***ESE?**	Event enable query	Returns the Standard Event Enable Register
***ESR?**	Event status register query	Returns and clear the Standard Event Register
***IDN?**	Identification query	Returns the unit’s identification string
***OPC**	Operation complete command	Sets the Operation Complete bit in the Standard Event Register
***OPC?**	Operation complete query	Places a “1” into the output queue when all device operations have been completed
***RST**	Reset command	Returns the unit to the *RST default condition
***SRE <NRf>**	Service request enable command	Sets the Service Request Enable Register
***SRE?**	Service request enable query	Returns the Service Request Enable Register
***STB?**	Status byte query	Returns the Status Byte Register
***TST?**	Self-test query	Performs the unit’s self-test and returns the result.
***WAI**	Wait-to-continue command	Waits until all previous commands are executed

2.1.2 Command Reference

1. *IDN? – identification query - read identification code

The identification code includes the manufacturer, model code, serial number, and firmware revision levels and is sent in the following format: `THORLABS,MMM,SSS,X.X.X`

Where: `MMM` is the model code

`SSS` is the serial number

`X.X.X` is the instrument firmware revision level

2. *OPC – operation complete - set OPC bit

3. *OPC? – operation complete query – places a “1” in output queue

When *OPC is sent, the OPC bit in the Standard Event Register will set after all pending command operations are complete. When *OPC? is sent, an ASCII “1” is placed in the Output Queue after all pending command operations are complete.

Typically, either one of these commands is sent after the INITiate command. The INITiate command is used to take the instrument out of idle in order to perform measurements. While operating within the trigger model layers, many sent commands will not execute. After all programmed operations are completed, the instrument returns to the idle state at which time all pending commands (including *OPC and/or *OPC?) are executed. After the last pending command is executed, the OPC bit and/or an ASCII “1” is placed in the Output Queue.

4. *RST – reset – return instrument to defaults

When the *RST command is sent, the instrument performs the following operations:

- Returns the instrument to the default conditions
- Cancels all pending commands.
- Cancels response to any previously received *OPC and *OPC? commands.

5. *TST? – self-test query – run self test and read result

Use this query command to perform the instrument self-test routine. The command places the coded result in the Output Queue. A returned value of zero (0) indicates that the test passed, other values indicate that the test failed.

6. *WAI – wait-to-continue – wait until previous commands are completed

The *WAI command is a no operation command for the instrument and thus, does not need to be used. It is there for conformance to IEEE488.2.

2.1.3 PAX1x specific SCPI Command Reference

Note: underlined parameters specify the default values after *RST.

All commands with a ’SCPI’ checkmark are described in detail in the SCPI specifications.

1. SYSTem subsystem commands

Command	Description	SCPI
SYSTem	Path to SYSTem subsystem.	Vol.2 §21
:ERRor		Vol.2 §21.8
[:NEXT]?	Returns the latest error code and message	Vol.2 §21.8.8
:VERSion?	Returns level of SCPI standard (1999.0)	Vol.2 §21.21

© 2022 Thorlabs GmbH

2. STATus subsystem commands

Command	Description	SCPI
STATus		Vol.2 §20
:OPERation	Path to control operation event registers	Vol.2 §20.1
[:EVENt]?	Returns the event register	Vol.2 §20.1.4
:CONDition?	Returns the condition register	Vol.2 §20.1.2
:ENABle <value>	Sets the enable register	Vol.2 §20.1.3
:ENABle?	Returns the enable register	Vol.2 §20.1.3
:QUEStionable	Path to control questionable event registers	Vol.2 §20.3
[:EVENt]?	Returns the event register	Vol.2 §20.3.4
:CONDition?	Returns the condition register	Vol.2 §20.3.2
:ENABle <value>	Sets the enable register	Vol.2 §20.3.3
:ENABle?	Returns the enable register	Vol.2 §20.3.3
:PRESet	Configures SCPI and device dependet status data structures.	Vol.2 §20.2
:AUXiliary	Path to control auxiliary event registers For the bits in the registers see description in chapters 3.6.4.3 and 3.6.4.3.6	
[:EVENt]?	Returns the event register	
:CONDition?	Returns the condition register	
:ENABle <value>	Sets the enable register	
:ENABle?	Returns the enable register	

3. SENSe1 Polarization sensing subsystem commands

Command	Description	SCPI
SENSe[1]	**Path to polarization sensing**	
:CALCulate		
[:MODe] {<num_val>|IDLE| H512|H1024|H2048| F512|F1024|F2048| D512|D1024|D2048}	Set averaging mode: 1 = H512 (half waveplate rotation with 512 point FFT) 2 = H1024 3 = H2048 4 = F512 5 = F1024 (one full waveplate rotation with 1024 point FFT) 6 = F2048 7 = D512 8 = D1024 9 = D2048 (two waveplate rotations with 2048 point FFT)	
[:MODe]?	Returns currently set averaging mode	
:CORRection		
:WAVelength {<num_val>|MIN| MAX|DEF}	Set wavelength in [m]	
:WAVelength? [{MIN|MAX|DEF}]	Returns wavelength in [m]	
:POWer		
:RANGe		
[:UPPer] {<num_val>|MIN|MAX}	Specifies the most positive signal level expected for the sensor input.	
[:UPPer]? [{MIN|MAX}]	Returns the most positive signal level in Watt the sensor input can handle in the active transimpedance amplifier configuration with any polarization state.	
:AUTO {0|OFF|1|ON(|2|ONCE)}	Sets the RANGe to the value determined to give the most dynamic range without overloading.	

© 2022 Thorlabs GmbH

:AUTO?	Returns the auto ranging	
:INDex {<num_val>|MIN|MAX}	Set the power range with the specified index. Range = 1 … 16 where 1 ist the least and 16 the most sensitive range	
:INDex? [{MIN|MAX}]	Returns the currently active power range index. Range = 1 … 16 where 1 ist the least and 16 the most sensitive range	
:NOMinal? [{<num_val>|MIN|MAX}]	Returns the most positive signal level in Watt the specified power range index can handle with any polarization state. If no index is specified the parameter defaults to the currently active index and the command is identical to SENS:POW:RANG:UPP?	
:DATA		
[:PRIMary]		
:LATest?	Returns latest completed primary measurement data set: revs, timestamp, paxOpMode, paxFlags (s. Kap. 3.6.4.2.3.1), paxTIARange, adcMin, adcMax, revTime, misAdj, theta, eta, DOP, Ptotal	

3.1 PAX Flags

The flags (= single bits of information) are stored alongside with each scan. This allows to recronstruct later the conditions under which the scan was taken. The C-language defines for the bits below are in file pax_status.h.

Bit Nr. Hex mask Dez.	Description
0 0x00000001 1	Motor speed is too low. I.e. below set speed – tolerance speed. The tolerance speed can be set during adjustment with the **`:DIAGnostic:MOTor:LIMits`** command. See chapter 3.6.4.2.5 7. Same as bit 0, MOTLO, in Questionable Data Registers, chap. 3.6.4.3.4 7
1 0x00000002 2	Motor speed is too high. I.e. above set speed + tolerance speed. The tolerance speed can be set during adjustment with the **`:DIAGnostic:MOTor:LIMits`** command. See chapter 3.6.4.2.5 7. Same as bit 1, MOTHI, in Questionable Data Registers, chap. 3.6.4.3.4 7
2 0x00000004 4	ADC underrun during last measurement. I.e. The lowest value acquired during the last scan was below the ADC_{MIN} threshold. The ADC_{MIN} threshold can be set during adjustment with the **`:DIAGnostic:LIMits`** command. See chapter 3.6.4.2.5 7. Same as bit 2, ADCLO, in Questionable Data Registers, chap. 3.6.4.3.4 7
3 0x00000008 8	ADC overflow during last measurement. I.e. The highest value acquired during the last scan was above the ADC_{MAX} threshold. The ADC_{MAX} threshold can be set during adjustment with the **`:DIAGnostic:LIMits`** command. See chapter 3.6.4.2.5 7. Same as bit 3, ADCHI, in Questionable Data Registers, chap. 3.6.4.3.4 7
4 0x00000010 16	PAX operating temperature is below limit. I.e. The temperature measured inside the PAX is below the low limit temperature threshold. The low limit temperature threshold can be set during adjustment with the **`:DIAGnostic:LIMits`** command. See chapter 3.6.4.2.5 7. Same as bit 11, UTEMP, in Standard Operation Registers, chap. 3.6.4.3.3 7
5 0x00000020 32	PAX operating temperature is above limit. I.e. The temperature measured inside the PAX is above the high limit temperature threshold. The high limit temperature threshold can be set during adjustment with the **`:DIAGnostic:LIMits`** command. See chapter 3.6.4.2.5 7. Same as bit 12, OTEMP, in Standard Operation Registers, chap. 3.6.4.3.3 7

© 2022 Thorlabs GmbH

6 0x00000040 64	USB power supply voltage is too low. I.e. The internally measured USB voltage is below the USB voltage threshold. USB voltage threshold can be set during adjustment with the **:DIAGnostic:LIMits** command. See chapter 3.6.4.2.5 7. Same as bit 10, UVOLT, in Standard Operation Registers, chap. 3.6.4.3.3 7
7 0x00000080 128	USB power supply current is too high. I.e. The internally measured USB current is above the USB current threshold. USB current threshold can be set during adjustment with the **:DIAGnostic:LIMits** command. See chapter 3.6.4.2.5 7. Same as bit 9, OCUR, in Standard Operation Registers, chap. 3.6.4.3.3 7
8 0x00000100 256	TIA gain switching is in progres. The CPU started a TIA gain switching, i.e. started transmitting a command via the I²C bus to the analog switch and the transmission did not finish yet. Same as bit 4, RNG, in Questionable Data Registers, chap. 3.6.4.3.4 7
9 0x00000200 512	TIA autoranging is enabled. Autoranging can be enabled or disabled with the **:SENSe[1]:POWer:RANGe:AUTO** command. See chapter 3.6.4.2.3 7. Same as bit 5, ENAAUTO, in Questionable Data Registers, chap. 3.6.4.3.4 7
10 0x00000400 1 024	A one-time TIA autoranging is in progres. A one-time autoranging can be executed with the **:SENSe[1]:POWer:RANGe:AUTO** command. See chapter 3.6.4.2.3 7. Same as bit 6, OTAUTO, in Questionable Data Registers, chap. 3.6.4.3.4 7
11 0x00000800 2 048	A out-of-sync condition occurred during the last measurement. Out-of-sync conditions ar realized by the firmware when (e.g.) the ADC sample counter and the encoder index pulse do not match. The results taken during this scan are invalid. Same as bit 7, SYNC, in Questionable Data Registers, chap. 3.6.4.3.4 7
12 0x00001000 4 096	External power supply present. The firmware has realized that an external power supply 8for the motor) is connected. When the external power supply is connected higher motor speeds are possible. Same as bit 8, VEXTPRES, in Standard Operation Registers, chap. 3.6.4.3.3 7
16 0x00010000 65 536	Motor is trimmed. This bit is a mirror of bit 0 (mask 0x01) of the status register of the motor controller chip DRV10983. Same as bit 0 in Auxiliary Status Registers, chap. 3.6.4.3.6 7
20 0x00100000 1 048 576	Motor is in lock condition. This bit is a mirror of bit 4 (mask 0x10) of the status register of the motor controller chip DRV10983. Same as bit 4 in Auxiliary Status Registers, chap. 3.6.4.3.6 7
21 0x00200000 2 097 152	Motor controller indicates over current condition. This bit is a mirror of bit 5 (mask 0x20) of the status register of the motor

	controller chip DRV10983. Same as bit 5 in Auxiliary Status Registers, chap. 3.6.4.3.6
23 0x00800000 8 388 608	Motor controller indicates over temperature. This bit is a mirror of bit 7 (mask 0x80) of the status register of the motor controller chip DRV10983. Same as bit 7 in Auxiliary Status Registers, chap. 3.6.4.3.6
24 0x01000000 16 777 216	Motor controller indicates hardware current limit. This bit is a mirror of bit 0 (mask 0x01) of the fault code register of the motor controller chip DRV10983. Same as bit 8 in Auxiliary Status Registers, chap. 3.6.4.3.6
25 0x02000000 33 554 432	Motor controller indicates abnormal motor speed. This bit is a mirror of bit 1 (mask 0x02) of the fault code register of the motor controller chip DRV10983. Same as bit 9 in Auxiliary Status Registers, chap. 3.6.4.3.6
26 0x04000000 67 108 864	Motor controller indicates abnormal K_T. This bit is a mirror of bit 2 (mask 0x04) of the fault code register of the motor controller chip DRV10983. Same as bit 10 in Auxiliary Status Registers, chap. 3.6.4.3.6
27 0x08000000 134 217 728	Motor controller indicates that the motor is missing. This bit is a mirror of bit 3 (mask 0x08) of the fault code register of the motor controller chip DRV10983. Same as bit 11 in Auxiliary Status Registers, chap. 3.6.4.3.6
28 0x10000000 268 435 456	Motor controller indicates that the motor is stuck in open loop. This bit is a mirror of bit 4 (mask 0x10) of the fault code register of the motor controller chip DRV10983. Same as bit 12 in Auxiliary Status Registers, chap. 3.6.4.3.6
29 0x20000000 536 870 912	Motor controller indicates that the motor is stuck in closed loop. This bit is a mirror of bit 5 (mask 0x20) of the fault code register of the motor controller chip DRV10983. Same as bit 13 in Auxiliary Status Registers, chap. 3.6.4.3.6

4. CALibration subsystem commands

Command	Description	SCPI
CALibration		
:STRing?	Returns the device's calibration string that was written to the device at the latest calibration procedure (usually the calibration date).	

© 2022 Thorlabs GmbH

	Example: “15-Jul-2015”	

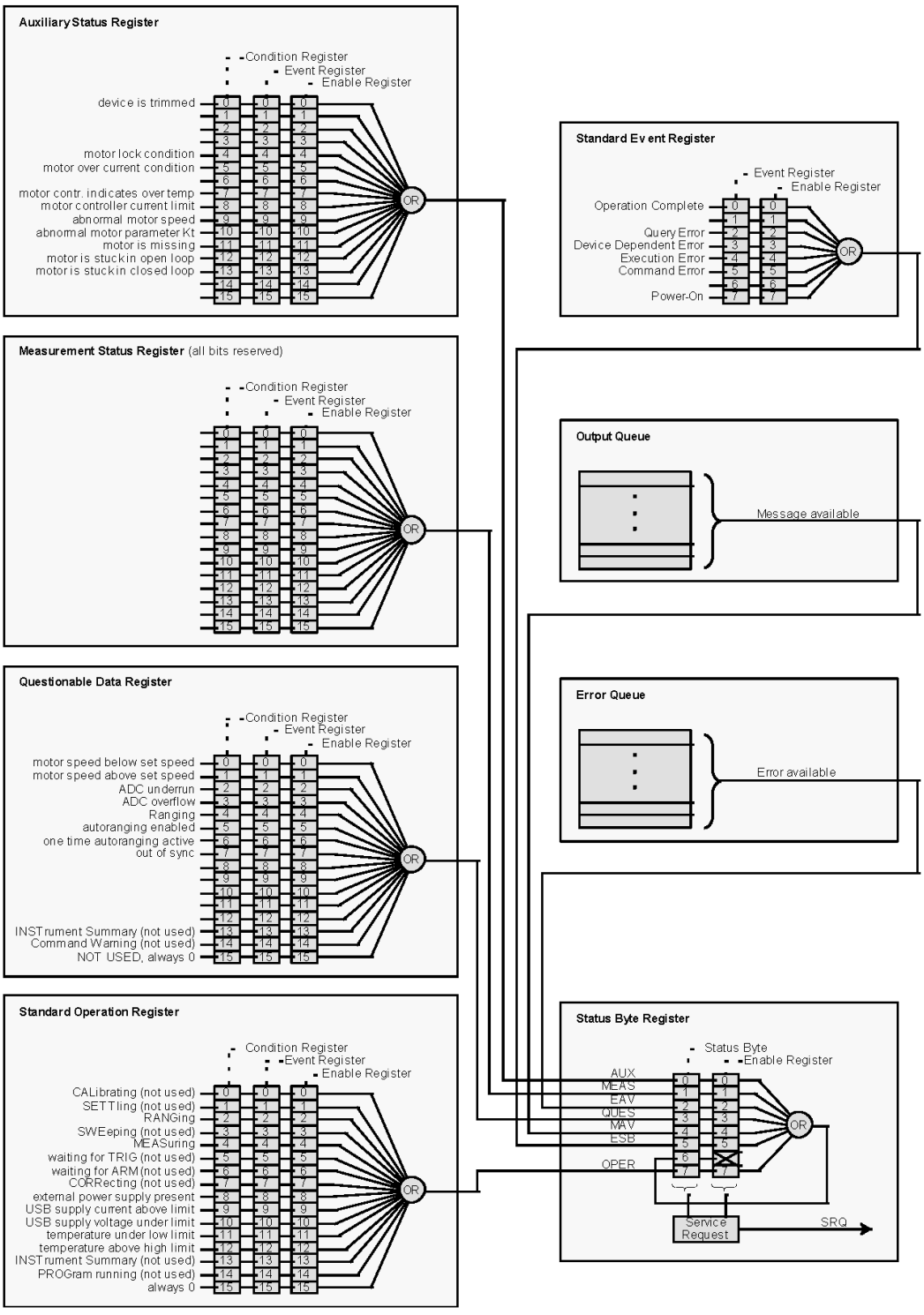
5. INPut subsystem commands

Command	Description	SCPI
INPut		
:ROTation		
[:STATe] {0|OFF|1|ON}	Enable/disable waveplate rotation.	
[:STATe]?	Returns waveplate motor state	
:VELocity {<num_val>|MIN|MAX|DEF}	Set the waveplates rotation velocity in 1/s. Note: The value range depends on the selected measurement mode and the power supply state. Changing these conditions will coerce the set value to the new limits.	
:VELocity? [{MIN|MAX|DEF}]	Returns the waveplate rotation velocity in 1/s	
:VELocity:LIMits?	Returns the maximum waveplate rotation velocity in 1/s for operation with and without external power supply	

© 2022 Thorlabs GmbH

2.1.4 Status Reporting

The figure gives an overview of the device's status reporting structure. See also chapter *STATus subsystem commands* for a detailed description of the related commands and their syntax.



1. Status Byte Register

The Status Byte Register gives a summary of all underlying status structures. See also IEEE488.2-1992-§11.2.

© 2022 Thorlabs GmbH

Bit #	Mnemonic	Description
7	**OPER**	Standard Operation Status Structure Summary Bit
6	**RQS/MSS**	Request Service / Master Summary Status
5	**ESB**	Standard Event Status Bit
4	**MAV**	Message Available. There is response data available for readout
3	**QUES**	Questionable Status Structure Summary Bit
2	**EAV**	Error Available. There is at least one error in the error queue.
1	**MEAS**	Measurement Status Structure Summary Bit
0	**AUX**	Auxiliary Status Structure Summary Bit

2. Standard Event Status Structure

The Standard Event Status Structure is described in IEEE488.2-1992-§11.5.

3. Standard Operation Register

The Standard Operation Status Structure is described in SCPI1999.0-Vol1-§9.3. In addition bit 8..12 are used as output state/on indicators.

Bit #	Mnemonic	Description
15..13		See SCPI1999.0-Vol1-§9.3
12	**OTEMP**	read as 1 when temperature is above high limit
11	**UTEMP**	read as 1 when temperature is below low limit
10	**UVOLT**	read as 1 when USB supply volatge is below limit
9	**OCUR**	read as 1 when USB supply current is above limit
8	**VEXTPRES**	read as 1 when external power supply is connected to PAX1000
7..0		See SCPI1999.0-Vol1-§9.3

4. Questionable Data Register

The Questionable Data Status Structure is described in SCPI1999.0 Vol1 §9.4.

Bit #	Mnemonic	Description

15..13		See SCPI1999.0-Vol1-§9.4 – flags are not implemented
12..8		reserved, read as 0
7	**SYNC**	the PAX measurement got out of synchronisation
6	**OTAUTO**	a one time autoranging is active
5	**ENAAUTO**	autoranging of photodiode amplifier (TIA) is enabled
4	**RNG**	the photodiode amplifier (TIA) is ranging
3	**ADCHI**	the ADC had an overflow condition during last measurement
2	**ADCLO**	the ADC had an underrun condition during last measurement
1	**MOTHI**	the motor is running above set speed
0	**MOTLO**	the motor is running below set speed

5. Measurement Status Register

The Measurement Status Register Status Byte Register reports device operation and measurement states. This register is currently not used for PAX1000 devices

Bit #	Description
0…15	not used

6. Auxiliary Status Register

The Auxiliary Status Register Status Byte Register reports auxiliary device states. For the PAX1000 this register mirrors the status byte (0x10) and the fault code register (0x1E) of the motorcontroller chip (DRV10983).

Bit #	Description
15...14	reserved, read as 0
13	read as 1 when the motor controller indicates that the motor is stuck in closed loop (bit 5 of reg. 0x1E of DRV10983 is set)

© 2022 Thorlabs GmbH

12	read as 1 when the motor controller indicates that the motor is stuck in open loop (bit 4 of reg. 0x1E of DRV10983 is set)
11	read as 1 when the motor controller indicates that the motor is missing (bit 3 of reg. 0x1E of DRV10983 is set)
10	read as 1 when the motor controller indicates an abnormal motor parameter Kt (bit 2 of reg. 0x1E of DRV10983 is set)
9	read as 1 when the motor controller indicates an abnormal motor speed (bit 1 of reg. 0x1E of DRV10983 is set)
8	read as 1 when the motor controller indicates hardware current limit (bit 0 of reg. 0x1E of DRV10983 is set)
7	read as 1 when the motor controller indicates over temperature (bit 7 of reg. 0x10 of DRV10983 is set)
6	reserved, read as 0
5	read as 1 when the motor controller indicates over current condition (bit 5 of reg. 0x10 of DRV10983 is set)
4	read as 1 when the motor controller indicates that the motor is locked (bit 4 of reg. 0x10 of DRV10983 is set)
1…3	reserved, read as 0
0	read as 1 when the motor controller indicates that the device is trimmed (bit 0 of reg. 0x10 of DRV10983 is set)

2.1.5 Error Reporting

The device stores errors in a queue containing up to 10 entries. The error queue may be read out by the `'SYSTem:ERRor[:NEXT]?'` command. The following table lists all error numbers and the according descriptive messages.

Note: Negative numbers are defined by SCPI while positive error numbers are device dependent.

Note: Not all error numbers provided in the list below may occur, especially most of the negative numbers defined by SCPI are noted for completeness only.

Error	Description
0	No error

101	Attempt to set or use a PAX operating mode that is not supported
107	An I^2C bus communication failed
108	Attempt to switch Motor direction while Motor is powered
109	Attempt to switch on Motor while Temperature is out of operating range
111	The wanted result was not found in the ringbuffer
112	Restore from NVMEM is not possible, checksum error
195	IAP error, please contact Thorlabs
196	Authentication required
197	This command is not yet implemented
199	Error code for not yet properly defined errors
-100	General command error
-101	Invalid character
-102	Syntax error
-103	Invalid separator
-104	Data type error
-105	GET not allowed
-108	Parameter not allowed
-109	Missing parameter
-110	Command header error
-111	Header separator error
-112	Program mnemonic too long
-113	Undefined header (Unknown command)

© 2022 Thorlabs GmbH

-114	Header suffix out of range
-115	Unexpected number of parameters
-120	Numeric data error
-121	Invalid character in number
-123	Exponent too large
-124	Too many digits
-128	Numeric data not allowed
-130	Suffix error
-131	Invalid suffix
-134	Suffix too long
-138	Suffix not allowed
-140	Character data error
-141	Invalid character data
-144	Character data too long
-148	Character data not allowed
-150	String data error
-151	Invalid string data
-158	String data not allowed
-160	Block data error
-161	Invalid block data
-168	Block data not allowed
-170	Expression error
-171	Invalid expression

© 2022 Thorlabs GmbH

-178	Expression data not allowed
-180	Macro error
-181	Invalid outside macro definition
-183	Invalid inside macro definition
-184	Macro parameter error
-200	General execution error
-201	Invalid while in local
-202	Settings lost due to rtl
-203	Command protected
-210	General trigger error
-211	Trigger ignored
-212	ARM ignored
-213	Init ignored
-214	Trigger deadlock
-215	Arm deadlock
-220	Parameter error
-221	Settings conflict
-222	Data out of range
-223	Too much data
-224	Illegal parameter value
-225	Out of memory
-226	Lists not same length
-230	Data corrupt or stale

© 2022 Thorlabs GmbH

-231	Data questionable
-233	Invalid version
-240	Hardware error
-241	Hardware missing
-250	Mass storage error
-251	Missing mass storage
-252	Missing media
-253	Corrupt media
-254	Media full
-255	Directory full
-256	File name not found
-257	File name error
-258	Media protected
-260	Expression error
-261	Math error in expression
-270	Macro error
-271	Macro syntax error
-272	Macro execution error
-273	Illegal macro label
-274	Macro parameter error
-275	Macro definition too long
-276	Macro recursion error
-277	Macro redefinition not allowed

© 2022 Thorlabs GmbH

-278	Macro header not found
-280	Program error
-281	Cannot create program
-282	Illegal program name
-283	Illegal variable name
-284	Program currently running
-285	Program syntax error
-286	Program runtime error
-290	Memory use error
-291	Out of memory
-292	Referenced name does not exist
-293	Referenced name already exists
-294	Incompatible type
-300	Device specific error
-310	System error
-311	Memory error
-312	PUD memory lost
-313	Calibration memory lost
-314	Save/recall memory lost
-315	Configuration memory lost
-320	Storage fault
-321	Out of memory
-330	Self-test failed

© 2022 Thorlabs GmbH

-340	Calibration failed
-350	Queue overflow
-360	Communication error
-361	Parity error in program message
-362	Framing error in program message
-363	Input buffer overrun
-365	Time out error
-400	Query error
-410	Query INTERRUPTED
-420	Query UNTERMINATED
-430	Query DEADLOCKED
-440	Query UNTERMINATED after indefinite response
-500	Power on
-600	User request
-700	Request control
-800	Operation complete

© 2022 Thorlabs GmbH



THORLABS

www.thorlabs.com