TEL/FAX.: (514) 328-7499
TOLL FREE: (844) 488-7499 (USA/CANADA)
EMAIL: support@vpixx.com
www.vpixx.com

VPixx Release Guide

Version 3.12

Table of Contents

Table of Contents 1
Overview 2
Document Icons 2
New firmware 3
New general behavior for 3D 3
Button Listener Class in pypixxlib 3
TRACKPixx3 pypixxlib Expansion 3
Pixel mode helper functions 4
Support for ADC and DAC Simple Scheduling in pypixxlib 5
Support for PROPixx TSCOPE in pypixxlib 5
M16 pypixxlib Shader Update 6
Miscellaneous pypixxlib changes 6
VPixx API changes (libdpx/libtpx) 7
Matlab Datapixx Toolbox 7
New 3D Demo in PyPixx 7
New TRACPixx3 modes in PyPixx and MATLAB 8
Simulator Update to 1.3 9

Overview

This release guide provides installation and usage information relating to the latest VPixx Software Tool release.

VPixx Technologies Inc. reserves the right to modify or otherwise update this document without notice as required by a constantly evolving marketplace, client requests or to adapt to new progress or constraints in engineering or manufacturing technology. The information contained in this document may change without prior notice.

For more information about our company and products, visit our Web site at ***www.vpixx.com***

For information, comments or suggestions, please contact us by e-mail at ***support@vpixx.com***

Our offices are located at:

630 Clairevue West suite 301
Saint-Bruno, Qc
Canada, J3V 6B4

Document Icons

The use of icons emphasizes helpful, caution or warning notes. Below is a list of the available icons.

TABLE 1 – DOCUMENT ICONS

Icon	Description	Description
	Helpful Hint	Information to help out during assembly, installation or usage
	Caution Notice	Important Information to prevent misuse and/or damage to equipment
	Warning	**Critical information to prevent damage to equipment and/or injury to personnel or participants**

New firmware

DATAPixx 3 Revision 24 (was 22):

- New TRACKPixx tracking mode for NHP
- Console resolution can now be selected

DATAPixx2 / VIEWPixx / PROPixx Controller Revision 56 (was 55):

- HSPLIT function on DATAPixx2 has been removed. If you need this to work, contact us at support@vpixx.com and we will send you a compatible firmware.

PROPixx Revision 53 (was 50):

- Stability fix in case of rare issue of no image on power-up.

New general behavior for 3D

The new default behavior for 3D test patterns is to assume Volfoni glasses are connected to a VIEWPixx /3D. If you have Nvidia glasses, you will need to adjust the software settings.

Button Listener Class in pypixxlib

You can find more information on how to use this class with PsychoPy in our newest Vocal here: https://vpixx.com/vocal/psychopy/

A new button listener can be found in the pypixxlib package. This allows you to create an object that you can use to start and stop logging button presses, and then get an organized summary of button activity. As well, the class offers additional tools to control the LEDs on the RESPONSEPixx, if applicable, and wait for a specific button event. This class replicates much of the behavior of the RESPONSEPixx functions from PsychToolBox in MATLAB. This new pypixxlib class is located in the responsepixx.py file.

TRACKPixx3 pypixxlib Expansion

pypixxlib now offers a Python equivalent to all of the TRACKPixx functionality offered in our MATLAB API. The TRACKPixx3 class defined in tracker.py now has additional functionality:

- Start and stop data recording, and get the current status of an ongoing data collection schedule.
- Obtain all newly available data in one line of code, without the need to manually access the RAM.
- Check if a participant is currently fixating or doing a saccade.

As well, our _libdpx.py API has numerous other TRACKPixx3 additions:

- You can now get the eye positions estimates utilized by the calibration routine, which you can use to independently validate the calibration.
- Pupil size (in units of pixels) is also now available in Python.
- You can get the Cartesian coordinates of pupil center and corneal-reflection center in units of pixels.
- A TRACKPixx schedule can now be implemented in a user-friendly way with simple setup/start/stop functions.
- Obtaining eye tracking data has been coded in a user-friendly way, allowing you to simply get a list of all the eye tracking data.
- You can now request a software copy of the TRACKPixx's current camera image.
- You can now get/set/clear eye tracker search limits. Search limits restrict the eye tracking algorithm to operate on only a certain portion of the camera image, which can improve performance.

To facilitate the use of all of these changes, new demo files have been created: TPxCalibrationTesting.py, TPxUtils.py, TPxValidateCalibration.py. TPxCalibrationTesting.py can be run on its own. It performs a calibration and a subsequent validation, mimicking the behavior of the MATLAB demo of the same name. TPxUtils.py contains helper functions which you may need in your experiment. TPxValidateCalibration.py is a simple script that can present gaze targets on the screen and assesses the quality of your calibration. These demos use PsychoPy to present visual stimuli.

Pixel mode helper functions

Two new functions have been added to pypixxlib which help you convert between pixel mode trigger values and the digital output value. In pixel mode, the values of the digital output are determined by the color value of the top left pixel. Most often, that color value is encoded by your video rendering hardware/software as an RGB triplet: a sequence of 3 values between 0 and 255. The digital output value, however, is a single number between 0 and 2^{24}-1. (The binary representation of this number requires 24 digits, i.e., 24 bits, and the value of each bit corresponds to the state of a specific digital out pin.) These functions will allow you to easily convert between an RGB color triplet and a digital output value.

Pixel Mode to Trigger (_libdpx.py or DigitalOut object)

```
DPxRGBToTrigger(color) or dout.RGBToTrigger(color
```

Trigger to Pixel Mode (_libdpx.py or DigitalOut object)

```
DPxTriggerToRGB(trigger) or dout.TriggerToRGB(color)
```

Support for ADC and DAC Simple Scheduling in pypixxlib

pypixxlib now includes scheduling functions that provide identical functionality as in MATLAB. The following functions are now available:

DAC

```
ADCdictionary =  DPxSetAdcSchedule(onSet, rateValue, rateUnits, maxScheduleFrames, channelList = None,
bufferBaseAddr = 4e6, numberBufferFrames=None)

data, timetags = DPxReadAdcBuffer(inDict, numFrames = 0, customReadAddr = None)

DPxGetAdcStatus(inDict)
```

DAC

```
nextWriteAddr = DPxWriteDacBuffer(bufferData, bufferAddress=0, channelList=None)

DACdictionary = DPxSetDacSchedule(scheduleOnset, scheduleRate, rateUnits,
maxScheduleFrames, channelList=None, bufferBaseAddress=0, numBufferFrames=None)

DPxGetDacStatus(inDict)
```

In contrast to MATLAB, scheduling functions in Python require an extra step: you must propagate the dictionary created it by the Set function. That dictionary must be passed to all further scheduling functions. These functions are also available in the digitalOut and analogOut classes.

Support for PROPixx TSCOPE in pypixxlib

pypixxlib now includes TSCOPE functions that provide identical functionality as in MATLAB. The following functions are now available:

Matlab	Python
`Datapixx('SetPropixxTScopeMode' [, mode=0]);`	`def DPxSetPPxTScopeMode(mode = 0):`
`Datapixx('SetPropixxTScopeProgramAddress' [, addr=0]);`	`def DPxSetPPxTScopeProgAddr(addr = 0):`
`Datapixx('SetPropixxTScopeProgramOffsetPage', mode);`	`def DPxSetPPxTScopeProgOffsetPage(offset):`
`Datapixx('SetPropixxTScopeProgram', program);`	`def DPxSetPPxTScopeProg(program):`
`Datapixx('EnablePropixxSoftwareTestPatternLoad');`	`def DPxEnablePPxSwtpLoad():`
`Datapixx('DisablePropixxSoftwareTestPatternLoad');`	`def DPxDisablePPxSwtpLoad():`
`Datapixx('SetPropixxSoftwareTestPatternLoadPage', page);`	`def DPxSetPPxSwtpLoadPage(page):`
`Datapixx('EnablePropixxTScope');`	`def DPxEnablePPxTScope():`
`Datapixx('DisablePropixxTScope');`	`def DPxDisablePPxTScope():`
`Datapixx('EnablePropixxTScopePrepRequest');`	`def DPxEnablePPxTScopePrepReq():`
`Datapixx('DisablePropixxTScopePrepRequest');`	`def DPxDisablePPxTScopePrepReq():`
`Datapixx('WritePropixxTScopePages', pageData, pageIndex [, nPages=1]);`	`def DPxWritePPxTScopePages(index, data):`
`Datapixx('SetPropixxTScopeSchedule', scheduleOnset, scheduleRate, maxScheduleFrames [, startPage=0] [, nPages=maxScheduleFrames]);`	`def SetPropixxTScopeSchedule(onset, rate, maxScheduleFrame, startPage = 1, nPages = None):`
`Datapixx('StartPropixxTScopeSchedule');`	`def DPxStartPPxTScopeSched():`
`Datapixx('StopPropixxTScopeSchedule');`	`def DPxStopPPxTScopeSched():`

We have also added one demo to match the MATLAB files, found in the example folder of pypixxlib (named *Propixx_10kHzTScopeDemo.py*).

M16 pypixxlib Shader Update

The M16 shader code has been updated to work with the new GLSL standard. This should not affect any code currently written.

Miscellaneous pypixxlib changes

- The installer on Windows has changed from an executable (.exe) to the same format as other operating system. Use pip to install the software.
- The names of certain audio scheduling functions referred to “Left” and “Right”, which unintuitively did not refer to any stereo features have been renamed. To make these more intuitive, they have been changed: “Left” has been removed from function names altogether (it referred to the default audio schedule), and “Right” has been renamed “Aux” since it refers to the auxiliary audio port.
- Some problems were identified in the functions that synchronize register updates with the video signal (*vsync*) and pixel sequence (*psync*). These have been resolved.

  As an overview, here is how to use the revised register update functions synchronized to the video signal:

```
In dpxDevice.py
device.updateRegCacheAfterPixelSync(pixelData, timeout)
device.writeRegCacheAfterPixelSync(pixelData, timeout)

device.writeRegCacheAfterVideoSync():
device.updateRegCacheAfterVideoSync():


In _libdpx.py
DPxUpdateRegCacheAfterPixelSync(pixelData, timeout)
DPxWriteRegCacheAfterPixelSync(pixelData, timeout)

DPxUpdateRegCacheAfterVideoSync()
DPxWriteRegCacheAfterVideoSync ()
```

VPixx API changes (libdpx/libtpx)

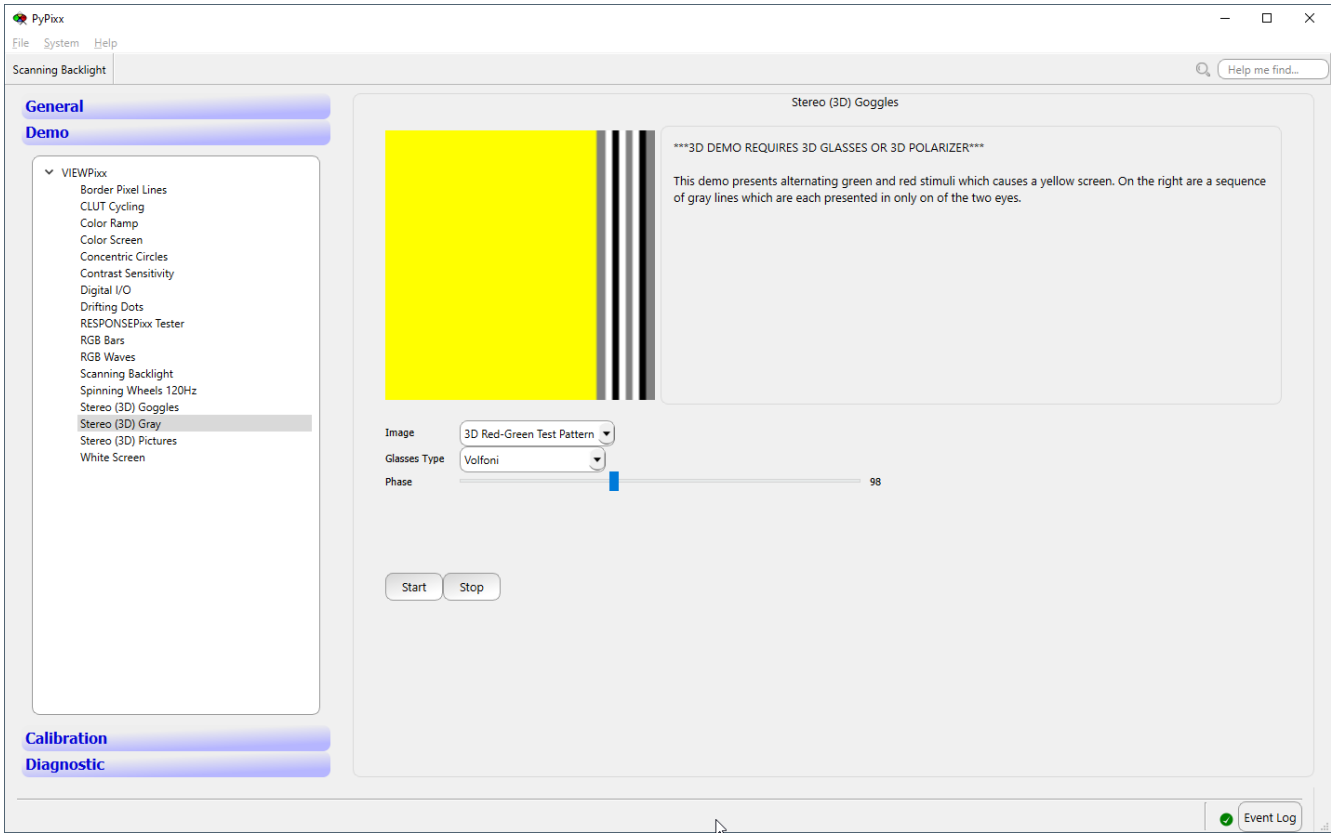
- Removed the need for users to manage constants when using the API (e.g., no more memory addresses for retrieving TRACKPixx camera images).
- Halted the creation of debug files by certain eye tracking functions.
- Fixed an issue in which TRACKPixx functions did not read and return distance in the same units.
- Fixed an issue in which the function `TPxSetIrisExpectedSize` did not work on macOS.

Matlab Datapixx Toolbox

- Support for Octave 5 has been added for Linux. In the software tools, you can now find Datapixx_octave4.mex and Datapixx_octave5.mex. Copy these files into the appropriate directory of your PsychToolBox installation and rename them to “Datapixx.mex”.
- Errors in some demos have been fixed.
- Pixel mode disable function now disables both pixel modes.

New 3D Demo in PyPixx

All 3D demos (test patterns) that used to be found in PyPixx have been unified into a single demo. This demo allows you to select from a list of test patterns, select the type of 3D glasses, and adjust the video signal phase to find the best value for your setup.



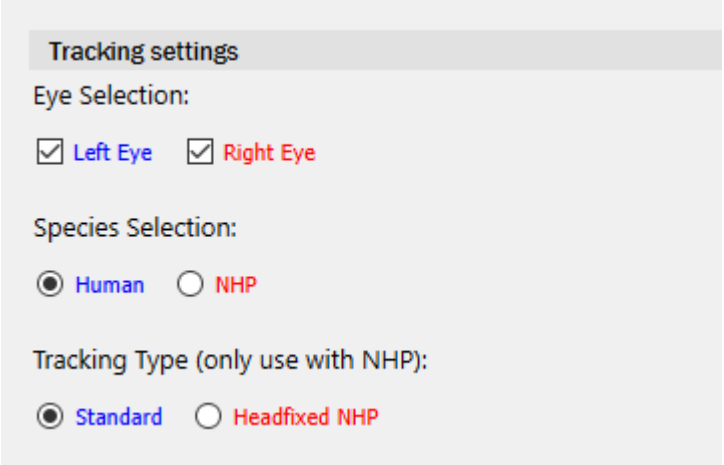
New TRACPixx3 modes in PyPixx and MATLAB

Two new modes have been added to help with NHP tracking. This requires firmware 23 on the DATAPixx3.

You can now select the species in either software. The choices are Human or NHP. When tracking NHP and selecting that option, you can set the IR LED intensity to higher values, if needed. In theory this should only be needed in setup where the TRACKPixx is further than the recommended distance.

The tracking type should only be changed when using the TRACKPixx3 with NHP. If you are using head fixed NHP you can select this new mode which will greatly improve the tracking for head fixed subjects.

In PyPixx, you can access these options by navigating to Demo → TRACKPixx → Settings:



In Matlab, there are four new functions available to you:

```
Datapixx('SetTrackingMode', mode);
mode = Datapixx('GetTrackingMode');
Datapixx('SetTrackingSpecies', Species);
species = Datapixx('GetTrackingSpecies');
```

PyPixx Stability Upgrade

Stability update

- RESPONSEPixx demo now updates much more often to prevent missing button presses.
- Fixed a bug whereby the TRACKPixx camera image did not update on systems with certain graphics card/drivers.

Simulator Update to 1.3

Stability update

Many bugs causing older versions of the Simulator to crash have been resolved. However, testing shows it might still be necessary for you to restart both the VPixx Device Server and the Simulator in some instances, should you encounter a crash.

Low-rate schedules

We corrected a bug whereby schedules running at less than 2 Hz only reported data from the beginning of the schedule. While the signal viewer showed the signal propagation correctly, when using a loopback, for example, the values were not registered in the simulated device RAM and therefore not usable in our other software. This has been solved and schedules of all rates up to the maximum possible value (~100 kHz) now work.

Crash when no VPixx Device Server is detected

The simulator will now keep working even when the server is not detected on launch. You will get the same warning as when the VPixx device server stops responding during an experiment to indicate that you should restart it.

VPixx Device Server

Fixed an issue which caused the server to log superfluous information on macOS and Linux. If you are using an old server, you might have very large log files in your directory that can be deleted.