WHAND USER MANUAL (V2)

Table des matières

[1. Introduction 2](#_Toc23171171)

[2. Package content 2](#_Toc23171172)

[2.1. Extracting the files 2](#_Toc23171173)

[2.2. List of sub-directories 3](#_Toc23171174)

[2.3. Content of directory “doc” 3](#_Toc23171175)

[2.4. Content of directory “Install” 3](#_Toc23171176)

[2.5. Content of directory “whand\_modules” 3](#_Toc23171177)

[3. Installing the components 4](#_Toc23171178)

[4. Options and parameters 4](#_Toc23171179)

[4.1. whand\_parameters.py 4](#_Toc23171180)

[4.1.1. Compilation options: 4](#_Toc23171181)

[4.1.2. Debugging options: 5](#_Toc23171182)

[4.1.3. I/O options 5](#_Toc23171183)

[4.1.4. Display options 6](#_Toc23171184)

[4.1.5. Global constants 6](#_Toc23171185)

[4.1.6. Debugging options 7](#_Toc23171186)

[4.2. config.txt 7](#_Toc23171187)

[4.3. Connection file(s) 8](#_Toc23171188)

[5. Getting started 8](#_Toc23171189)

[5.1. Tests 8](#_Toc23171190)

[5.2. Running hardware 8](#_Toc23171191)

[5.3. The interactive panel 9](#_Toc23171192)

[5.3.1. Sample interactive panel 9](#_Toc23171193)

[5.3.2. Active buttons 9](#_Toc23171194)

[5.3.3. Information cells 10](#_Toc23171195)

[6. Adapting Whand to new hardware 10](#_Toc23171196)

[6.1. Description of essential functions 11](#_Toc23171197)

WHAND USER MANUAL (V2)

*By Alain R. Marchand, oct. 2019*

This user manual aims is intended as a guide to install and use a particular implementation of Whand (Whand\_V2\_601\_Elvendust). For an informal introduction to the language, see WHAND TUTORIAL. For a complete description of the syntax of Whand, see WHAND REFERENCE MANUAL. For a deeper understanding of the mechanics of Whand, see WHAND DEVELOPER MANUAL.

Whand is a language with simplified syntax and limited operations. It is intended for the control of simple automatons and emphasizes the readability of programs.

Current Whand version is V2. There is no Whand V1.

# Introduction

This user manual is a practical guide, not a language tutorial. It focuses on installation package Whand\_V2\_601\_Elvendust.zip, but may be largely applicable to later versions. With this manual, you should be able to [install](#_Installing_the_components) the Whand program and the Whand language module for editor [Notepad++](#_Installing_the_components), to parametrize Whand to your needs, and to edit and run a Whand script.

A program running under Microsoft Excel® (TraiteSK\_V10j.xlsm) is also provided to analyze the data.

This package is primarily intended for use under Microsoft Windows® (Windows Seven and above) typically with an ASi interface (Biehl-Wiedemann) and the hardware provided by [Imetronic](http://www.imetronic.com) (Pessac, France) which includes eight polymodal cages.

Whand may run under Python 2.7, but Python 3.6 is recommended. Whand makes use of the Python tkinter module for display.

Adapting the program to other hardware systems requires a modification of the Whand\_driver.py module, and possibly a modification of the Whand\_io.py module, see WHAND DEVELOPER MANUAL.

# Package content

Installation package Whand\_V2\_601\_Elvendust.zip contains all the necessary files to install and run Whand, as well as examples and documentation.

The installation process only consists of extracting files, and does not modify the Windows® register. Python and other programs such as Microsoft Excel® or Notepad++ need to be installed separately in Windows®.

## Extracting the files

To extract the files, copy the package file Whand\_V2\_601\_Elvendust.zip to a new directory, and use right mouse click to select “Extract all”. By default, files will be extracted in a subdirectory called Whand\_V2\_601\_Elvendust, thereafter called the Whand directory. Another Whand directory may be specified before pressing the “Extract” button.

Main program whand\_V2\_6 is copied in the extract directory, along with several accessory files (needed):

* ASIDRV32.dll dll for ASi hardware interface
* AsiDrv32\_c1.ini needed for ASi hardware interface
* asipci.dll dll for ASi hardware interface
* [config.txt](#_config.txt) data codes for inputs and outputs (used by whand\_io.py)
* RUN WHAND.bat batch file to directly run Whand in Python (may need to be adjusted for Python path)

Several sub-directories are created during the extraction.

## List of sub-directories

autotests: contains a series of 230 automated tests

data: default directory to record data

[doc](#_Content_of_directory): information files, manuals, tutorials

[Install](#_Content_of_directory_1): installers for Python-3.6.1, npp.7.5.9 and accessories

scripts: various examples of Whand scripts and run files

[whand\_modules](#_Content_of_directory_2): Python files required by Whand

## Content of directory “doc”

The following files may be useful for documentation, in particular those mentioned in **bold**.

Installing and running Whand: brief reminder of the present chapter

**commented script:** A complete Whand script with detailed comments

Inside Whand: some notes about the internal workings of Whand.

**Introduction to Whand programming**: an informal introduction to the language

race detection: some private notes about detecting race conditions

Tips for Whand: some examples of how to program simple operations

**Whand Reference Manual**: A complete description of the language

**Whand User Manual**: The present document

**Learn Whand**: Basic Whand syntax and operations

**Whand step-by-step**: A systematic introduction to basic Whand syntax and operations

Whand\_language: An informal description of the language

Whand for the lab: An informal presentation of Whand

**Analyzing data with traiteSK**: TraiteSK\_V12c user manual

## Content of directory “Install”

The following files may be needed to install Whand.

File “Installing and running Whand” is a brief reminder of the present chapter.

**npp.7.5.9.Installer.exe**: Auto-installer for Notepad++.

**python-3.6.1.exe**: Auto-installer for Python 3.6.1

**pywin32-224.win32-py3.6.exe**: Auto-installer for Python extension Pywin for Windows®

**TraiteSK\_V12c.xlsm**: A program running under Microsoft Excel® to analyze the data.

**Whand.xml**: Whand language module for Notepad++.

## Content of directory “whand\_modules”

The files in directory whand\_modules are essential building blocks of Whand.

Module whand\_driver.py must be [adapted to the hardware](#_Adapting_Whand_to).

Module whand\_io.py may be adapted to the file system (see WHAND DEVELOPER MANUAL).

Module whand\_operators.py may be adapted to the user’s language (see WHAND DEVELOPER MANUAL).

Other modules should not be modified unless absolutely necessary.

six.py: compatibility module for Python 2.7

whand\_compile.py: parses the script and creates objects

whand\_controlpanel.py: controls interactive display

whand\_critic.py: attempts to identify race conditions

whand\_driver.py: links with a specific hardware

whand\_io.py: controls file and input/output operations

whand\_nodes.py: basic class and methods of objects

whand\_operators.py: language-specific constants and messages

whand\_parameters.py: options and parameter values

whand\_precompile.py: prepares script text for parsing

whand\_sharedvars.py: class and methods of execution instances

whand\_tools.py: various functions for internal use

# Installing the components

The following component may need to be installed if they are not already in the system.

**Python:** Whand is implemented using the Python language (free software). Versions of Python 2.7 and above should be compatible. Version 3.6 is recommended. If no version of Python is installed, run **python-3.6.1.exe** to auto-install Python 3.6.1.

**Pywin for Windows®:** if not present, run **pywin32-224.win32-py3.6.exe** to auto-install Python extension Pywin for Windows® (free software).

**Notepad++ editor:** Whand scripts are text files that can be edited with any text editor. Notepad++ (free software) is a general purpose language editor that can be customized to display Whand scripts in user-friendly format, with different colors for keywords and comments.

Run **npp.7.5.9.Installer.exe** to auto-install Notepad++ if it is not already present. Then customize it with file **Whand.xml** as follows:

In Notepad++, select “Language/define your language” then click “Import”. Browse to directory “Install” to find Whand.xml, then click “Open”. A message “Import Successful” should appear. Close it.

Again select “Language” then click on “Whand” to finally customize the editor then close the window.

If you then open a Whand script file using the “File” menu, it should appear in color, with keywords in orange and comments in green.

**Data analysis:** To analyze the data, you may need to create a button for **TraiteSK\_V12c.xlsm** under Microsoft Excel® (see **Analyzing data with traiteSK**).

# Options and parameters

## whand\_parameters.py

File **whand\_parameters.py** is a Python file where options and parameter values for Whand can be set. This file should be edited with care. It is recommended to keep a backup before any modification.

Whand may be directly used with the standard options in simulation mode where parameter Hardware in I/O [options](#_I/O_options) is set to None.

If the setup involves eight [polymodal cages](#_Introduction) (Imetronic), it is sufficient to set parameter Hardware to ASi.

### Compilation options:

These options are used to check for errors in the script.

Warnings # display warnings at compile time

Dumps # in case of error or abort

Race\_test # looks for collisions at compile

Fatal\_race # abort if a collision is found

Fatal\_use\_before # abort at runtime if a value is used before being updated

Random\_update\_order # update order varies randomly at each compilation

Warnings=True displays non-fatal error messages at compile time. May be disabled once the script has been debugged, by setting this parameter to False (i.e. replace the line with Warnings=False ).

Dumps=True displays the script structure if compilation fails or program is aborted. May be disabled once the script has been debugged by setting this parameter to False.

Race\_test=True Tries at compile time to identify potential race conditions. This analysis may raise false alarms. A warning message does not mean the script is unstable, but it should prompt to user to verify the order of updating of the various expressions and sub-expressions. May be disabled by setting this parameter to False.

Fatal\_race=True Aborts compilation if Race\_test is set to True and a potential race condition has been identified. May be disabled by setting this parameter to False.

Fatal\_use\_before=True Aborts execution if an object is found to be used before being updated (potential race condition). May be disabled by setting this parameter to False.

Random\_update\_order=False If this parameter is set to True, the order of updating of the various objects is randomized. This may reveal instabilities in the script structure. Setting this parameter to False ensures a deterministic behavior of the script.

### Debugging options:

These options are used to manually test inputs.

Allow\_manual # interactive buttons

Simulatepin # simulation of inputs

Simulatepinlist # simulated inputs

Simulatepinfreq # simulated input frequency

Debug

Allow\_manual=True Displays interactive buttons for inputs on the controlpanel. The inputs can then be manually activated with the mouse. May be useful to test the script without the hardware. If Parameter is set to False, inputs are only activated by the hardware.

Simulatepin=True Automatically activates inputs on and off.

Simulatepinlist=[3,10] A list of inputs to be simulated.

Simulatepinfreq=4 A frequency (cycles/s) to activate the simulated inputs.

Debug=True Displays error messages for the ASi hardware.

### I/O options

These options specify the hardware and output file process. These parameters are used by **whand\_io.py** and **whand\_driver.py**.

ASi="ASi"

Hardware= None # ASi # ASi for Imetronic or None

Boxes=8 # number of parallel setups

Boxinputs=12 # for input with multiple parallel scripts

Online\_save= False # True # record data online or when finished

Autoname=False # automatic naming of result files

Use\_interrupts=False # allow interruptions on pin inputs

Split\_second=10 # time unit for results (1/s)

Nosave= False # True # do not record data at all

Printout= False # True # display message when i/o change

ASi is the AS interface by Bihl+Wiedemann GmbH (Germany). It is typically used in the hardware designed by Imetronic (Pessac, France). This interface can control several parallel setups (Boxes, typically 8), each with a number of inputs and outputs (Boxinputs, typically 12).

Hardware=ASi To use Whand in simulation mode, set Hardware to None.

Online\_save= False Data are kept in memory and written to file only at the end of the session. If this parameter is True, data are saved continuously during the experiment, but the files are not closed and will still be lost if the program is aborted.

Autoname=False Result files are named by user. If this parameter is True, a standard name is given instead by Whand.

Use\_interrupts=False Inputs are scanned on a regular basis (parameter Timestep). If this parameter is True, inputs may trigger an interrupt and be more reliably detected and timed. This function needs an appropriate callback procedure to be included in whand\_driver.py.

Split\_second=10 This defines the time unit with which the results are output to file (10 for 0.1 s, 20 for 0.05 s…).

### Display options

These options determine the position, size, structure and colors of the controlpanel.

### Global constants

These parameters are necessary for the correct functioning of Whand. They should not usually be modified.

Scriptdir="scripts" # Subdirectory for scripts and run files

Datadir="data" # Subdirectory for results

Max\_occur\_list=6 # memory span for event occurrence

Max\_iter=100

Max\_depth=20

Timestep=.05 # display refresh period (seconds)

Clock\_cycle=0.016 # used for synchro (16 ms)

Change\_time= 0.000001 # used to create transients (change)

Glitch\_time= 3\*Change\_time # used to create transients (begin, end)

Epsilon\_time= 8\*Change\_time # used to prevent event synchrony

FloatPrecision=0.00000005 # absolute limit for equality

BigNumber=1048576 # used to extend code range in conditions

It is possible to specify different directories for scripts (Scriptdir) or data (Datadir).

It is possible to increase the temporal accuracy by changing parameter Timestep to a smaller value. However, this comes at a cost of a greater processing time. Careful real-time tests of the script are recommended when using a Timestep lower than 0.01 (10 ms).

Max\_occur\_list determines the length of the list of the last occurrences of each event (see function '*occur*'). Reducing parameter Max\_occur\_list may provide a very small improvement in processing time if many events are monitored, but it may be problematic when using functions '*inter*', '*order*' or '*sequence*'. Increasing parameter Max\_occur\_list is necessary only if function '*occur*' is used to analyze many previous occurrences of an event. This may slow down processing.

Max\_iter and Max\_depth are used during initialization and updating to detect infinite loops. Processing exceeding these values will raise a warning. It might be necessary to change these values in the case of a very complex script.

The other parameters should not be changed.

### Debugging options

These options are for developers to debug Whand program itself.

## config.txt

Sample config.txt file:

-1 , 11 , 51 , # magasine visit 1

-2 , 10 , 50 , # lick

-7 , 12 , 52 , # magasine visit 2

-10 , 46 , 6 , # lever press 1

-11 , 47 , 7 , # lever press 2

-5 , 48 , 8 , # lever press 3

1 , 16, 56 , # food pellet 1

3 , 23, 63 , # clicker

4 , 25, 65 , # houselight

5 , 18, 58 , # LED 1

6 , 19, 59 , # LED 2

7 , 17, 57 , # food pellet 2

9 , 24, 64 , # tone

10 , 1, 41 , # present lever 1

11 , 2, 42 , # present lever 2

8 , 3, 43 , # present lever 3

12 , 27, 67 , # foot shock

101 , 6, 46 , # simulated press 1 (same as real press)

102 , 7, 47 , # simulated press 2

103 , 8, 48 , # simulated press 3

File **config.txt** (text file) defines input and output events are given codes used by Whand to generate a result file after each session. This file is hardware-dependent. Parallel setups are assumed to have the same configuration. The codes are expected by the analysis program **TraiteSK\_V10j.xlsm.**

The codes are related to hardware events (logical True or False, i.e. 1 or 0) as follows:

Each line is divided into four fields separated by commas: Hardware ID, ON code, OFF code and comment.

Hardware ID is positive for outputs and negative for inputs. Its absolute value indicates the input or output port number in the setup.

ON code is the value that will be output in the result file when the hardware port becomes True (transition to 1).

OFF code is the value that will be output in the result file when the hardware port becomes False (transition to 0).

The comment indicates the meaning of the event, in any format.

A text block with the meaning of all event codes is appended to each data file (text file). This text block is programmed under “global text constants” in **whand\_io.py**.

## Connection file(s)

Sample connection file:

# Wiring for Imetronic conditioning box (hardwired) + event codes for script

MV1: pin 1 # magazine 1 visit

MV2: pin 7 # magazine 2 visit

noP1: pin 10 # inverted lever 1 input

noP2: pin 11 # inverted lever 2 input

noP3: pin 5 # inverted lever 3 input

P1: when end noP1 until noP1 or start # non inverted lever 1 input

P2: when end noP2 until noP2 or start # non inverted lever 2 input

P3: when end noP3 until noP3 or start # non inverted lever 3 input

output 1: D1 # food dispenser 1

output 4: HL # houselight

output 7: D2 # food dispenser 2

output 9: Ta # tone (5000 Hz)

output 10: SP1 # present lever 1

output 11: SP2 # present lever 2

output 8: SP3 # present lever 3

unused pin(7), output(7), output(9)

For each hardware configuration, it is recommended to name each numbered input (*pin*) and output (*output*) in Whand so that its meaning is explicit. This may be done in the script itself, but more appropriately in an *include* file, the connection file (e.g. **connections\_b.txt**, see Whand Reference Manual). In practice, the same connection file may be called by *include* in multiple scripts. The input and output names in the connection file should match those used in the script. Keyword *unused* may help ensure the compatibility between the script and the connection file.

# Getting started

## Tests

The following steps can be followed to test Whand in simulation mode.

* [Install and configure the components](#_Installing_the_components)
* In the Whand directory, run file RUN WHAND.bat. (If it does not run, add the Whand directory to the Python path or edit the file to provide a path to the python.exe program).
* To the prompt “Select file containing script list ?”, just type the [Enter] key.
* This launches a series of 230 short tests in a window. It should take around 20 s and end with messages \*\*\* AUTOTEST COMPLETE \*\*\* then \*\*\* End of program \*\*\*, etc. If the tests halt before the end, it probably means that the installation has somehow failed.
* Close the window.

## Running hardware

Whand can now be configured to run an Imetronic setup:

* In file **whand\_parameters.py**, change line Hardware=None to Hardware=ASi
* Verify the presence of file **config.txt** in the Whand directory
* Verify the presence of file **connections\_b.txt** (or equivalent) in the scripts subdirectory.
* Use **Notepad++** to edit a Whand script in a text (.txt) file (see WHAND REFERENCE MANUAL).
* Use any text editor to create a text file (the run file) that contains the names of one or more of your script files (up to 8), separated by commas. The names may include a path if the files are located in subdirectories of the script subdirectory, e.g.

my\_script1.txt, my\_path\my\_script2.txt, my\_script3.txt

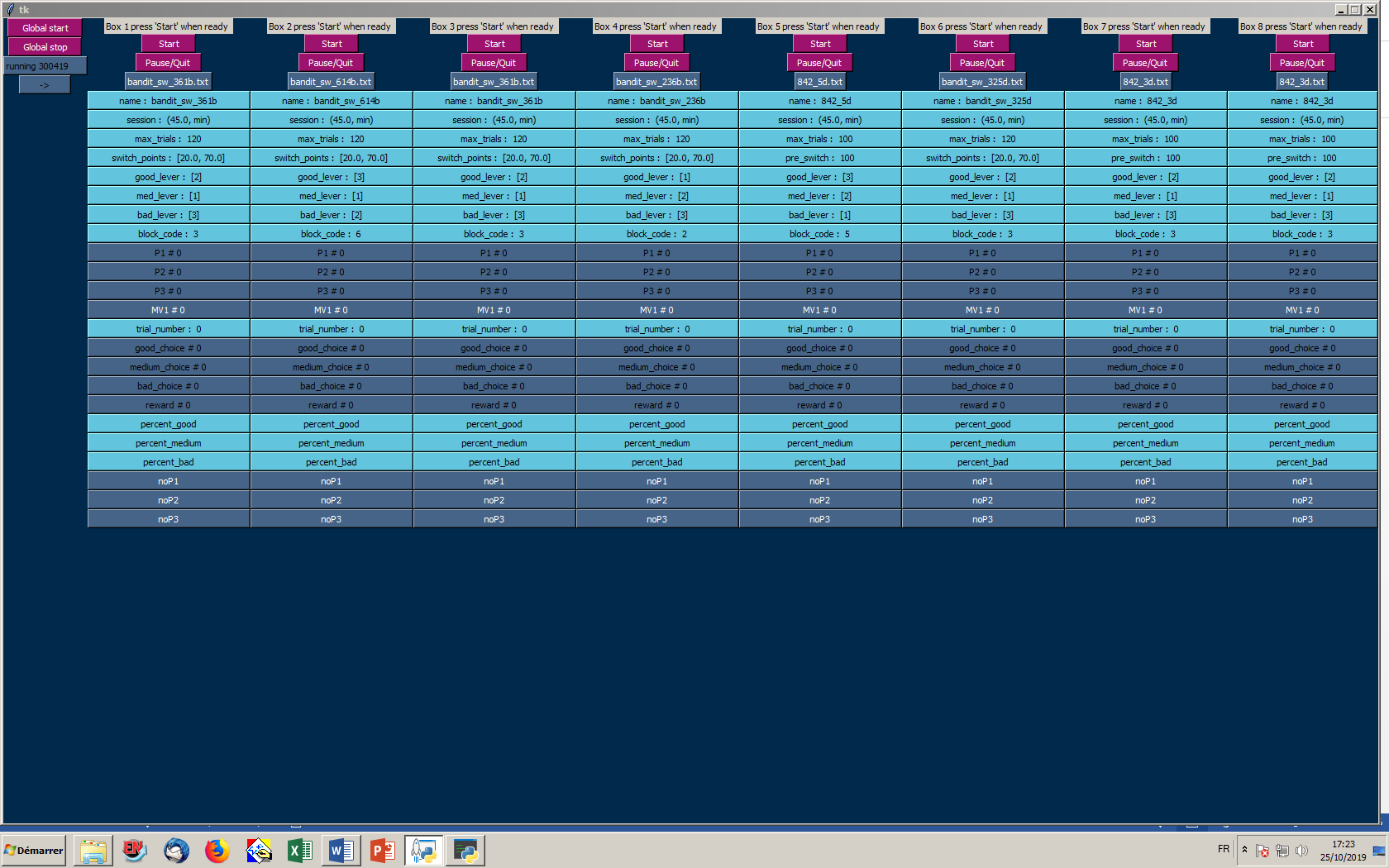
Whand will run each of these scripts independently in a different polymodal cage (here cages 1, 2, 3). Script “empty.txt” may be used to fill in unused cages (script contains only keyword *exit*).

* run file RUN WHAND.bat
* To the prompt “Select file containing script list ?”, type the name of your run file. If you want to run a single script, you can enter its name instead. The script will run in the small window if it does not contain a line with keyword *controlpanel*.
* If run file contains several script names or if one script contains a line with keyword *controlpanel*, the scripts will run on an interactive panel where the value of selected objects is displayed.
* You will then be prompted for a file to output results. If no results are needed, just type [Enter] key. If a file with the same name exists, Whand will ask if you want to overwrite it.

## The interactive panel

### Sample interactive panel

This panel shows the display for 8 cages before execution starts.



### Active buttons

The following buttons are available on the control panel (active buttons are shown in purple):

**Global start**: starts all cages at the same time when pressed. Cages that have already been started are not affected, i.e. continue execution. Once pressed, this button is disabled (shown in grey).

**Global stop**: stops all cages at the same time when pressed. Cages that have already been stopped are not affected. This button **must be used** to unlock the control panel so that it may be closed. Once pressed, this button is disabled.

**Start**: starts each cage separately. Once pressed, this button is disabled. In Pause mode, button changes to “Resume” and can be used to resume execution. When execution is terminated, button changes to “Ended”.

**Pause/Quit**: temporarily pauses execution on one cage. In Pause mode, clock timing is suspended and the “start” button for this cage changes to “Resume”. Pressing “Pause/Quit” a second time terminates execution and disables the button.

If option [Allow\_manual](#_Debugging_options) is true, additional buttons corresponding to inputs are presented for each cage.

### Information cells

running [name]: name of the run file giving script names for each cage.

🡪 [name]: name of the result file where data will be saved in subdirectory “data”. This includes one .xls file (actually a text file), e.g. “name.xls”, giving the contents of all visible information cells at the end of the experiment. In addition, for each cage, an individual text file (.e01) lists all events (inputs and outputs) that have occurred during execution. Each file is numbered with the cage number, e.g. “name3.e01” for cage 3.

Box [n] press “Start” when ready: This cell will continuously display time from the start of execution. The time in Pause mode is discounted. The cell also displays a “lag” value. If the lag value is greater than zero, it indicates that Whand is overloaded and cannot run at the desired speed.

The cell below the “Pause/Quit” button displays the name of the script file for this cage.

The other cells to be displayed must be specified in each script using *show* instructions.

# Adapting Whand to new hardware

Installation package Whand\_V2\_601\_Elvendust.zip is designed for hardware provided by [Imetronic](http://www.imetronic.com) (Pessac, France) with a maximum of eight polymodal cages and an ASi interface (Biehl-Wiedemann). In particular, it requires the following files

* ASIDRV32.dll dll for ASi hardware interface
* AsiDrv32\_c1.ini needed for ASi hardware interface
* asipci.dll dll for ASi hardware interface

as well as the provided module [whand\_driver.py](#_Content_of_directory_2).

The [result codes](#_config.txt) are also specific to this hardware setup.

The Python module [whand\_driver.py](#_Content_of_directory_2) may be modified by an experienced programmer to adapt it to a different hardware. The module contains a set of low-level functions used by Whand. Each of these functions must be declared and defined in the module, even if it is essentially empty (i.e. only contains instruction *pass*).

The essential functions are those used to initialize the hardware, to read or write a binary (event) port and to close the hardware. It may be sufficient to implement these functions.

init\_hardware() # open access to the hardware

open\_one\_box(box) # initializes access to one box

read\_all\_pins() # read all box bit inputs at once

read\_one\_pin(box, nb) # read a single bit input

write\_one\_bit(box, nb, bit) # write a single bit output (0 or 1)

close\_one\_box(box) # terminate one box

close\_hardware() # reset and close access to the hardware

Other nonessential functions are provided as an entry point into Whand but may or not be implemented in the current package. It is the responsibility of the programmer to determine whether and how to implement these functions.

init\_pin\_callback(box, nb) # needed even if no interrupt is used

activate\_interruption(box, nb) # needed even if no interrupt is used

simulate\_one\_pin(box, pinnumber, time) # simulate a single bit input

read\_one\_AD(box, ADnumber) # read a single analog port

write\_one\_DA(box, DAnumber, value) # write to a single analog port

read\_message(box, nb) # input a text from a port

write\_message(box, nb, text) # output a text to a port

display\_image(filename, X, Y) # display image from file

read\_touchscreen(filename, X, Y) # test touch on image

## Description of essential functions

init\_hardware() This function does not take any argument. It is assumed to initialize the whole setup, including the different cages/boxes that will function in parallel. If initialization is performed box by box, this function may be empty (i.e. only contain instruction *pass*).

open\_one\_box(box) This function takes as argument a box number (integer in range 1..8). It may be used to perform a specific initialization procedure on each cage/box. If initialization is performed globally, this function may be empty (i.e. only contain instruction *pass*).

read\_all\_pins() This function does not take any argument. It may be used to capture the input state of all cages/boxes simultaneously. In the current implementation, it uses a global variable in whand\_driver.py to pass information to function read\_one\_pin. If input capture is performed box by box, this function may be empty (i.e. only contain instruction *pass*).

read\_one\_pin(box, nb) This function takes as argument a box number (integer in range 1..8) and a pin number (in range 1..12) corresponding to one input port in the specified box/cage. It returns integer value 0 or 1.

write\_one\_bit(box, nb, bit) This function takes as argument a box number (integer in range 1..8) and a pin number (in range 1..12) corresponding to one output port in the specified box/cage and an integer value 0 or 1. It is assumed to set or clear only the specified output port and leave all other outputs unchanged.

close\_one\_box(box) This function takes as argument a box number (integer in range 1..8). It may be used to perform a specific closure procedure on each cage/box. If closure is performed globally, this function may be empty (i.e. only contain instruction *pass*).

close\_hardware() This function does not take any argument. It is assumed to close and reset the whole setup, including the different cages/boxes that will function in parallel. It is good practice to implement this function in order to leave the hardware in a standard state after use.