ELTEK
A Delta Group Company

HTOB Crontab Manager GUI - User Guide

Edition 01
17.8.2020

PETER ANTOSKA (petantsk)

Document Revisions

Date	Version Number	Document Changes
17/08/2020	01	Initial release

Table of Contents

1 Introduction

1.1 Scope and Purpose

HTOB Crontab Manager GUI (HTOB Manager) provides the basic functionalities to control HTOB test. Besides that it provides a capability

- to run manually a predefined scripts/commands,
- to inform the user about new errors during the execution of Python scripts,
- to track an errors based on user exception code inserted into Python scripts,
- to handle logging data.

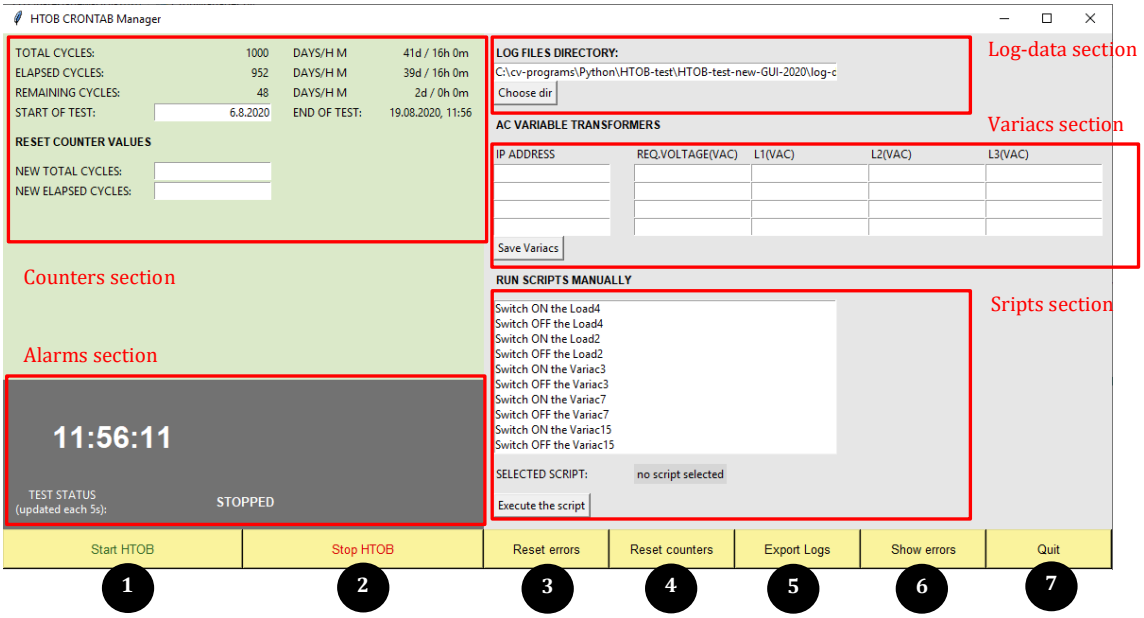


Fig.1: HTOB Manager GUI.

1.2 Overview

LFManager has Graphical User Interface (GUI) with 1 basic panel and several sections (see Fig.1):

1. **Counters section:**

- number of test cycles/date and time information for running test.

2. **Alarm section:**

- new alarm event results to change of background of this section to red color,

- actual test status (running/stopped) is represented in text form and background color (green/gray),

- actual CET in HH:MM:SS format.

3. **Log-data section:**

- any local folder can be selected by user to store the log-files.

4. **Variable transformers (Variacs) section (not functional in this release):**

- requested target AC voltage for 4 selected variacs can be defined by user,

- actual output AC line-neutral voltages are shown for 4 selected variacs.

2. **Scripts section:**

- selected script from list commands can be triggered manually by user,

- list of scripts can be updated by user in local configuration text file.

Main menu is located at the bottom of GUI panel:

❶ - Start HTOB test (/etc/crontab-final file is copied to /etc/crontab file).

❷ - Stop HTOB test (/etc/crontab-stop file with empty content is copied to /etc/crontab file).

❸ - Reset errors (delete all records) in *error-detection.txt* file. When HTOB test is running in errorless mode, no data is inside *error-detection.txt* file (file is empty).

❹ - Update the counters "TOTAL CYCLES" and "ELAPSED CYCLES" with values from counters fields "NEW TOTAL CYCLES" and "NEW ELAPSED CYCLES".

❺ - Copy the log-files *htob-test.txt* and htob-test-scripts.txt into the archive folder. Archive folder can be selected by GUI user.

❻ - Show all error events written in *error-detection.txt* file.

❼ - Quit GUI application.

Additional configuration data for HTOB Manager is stored in separate text file *CronManager-config.ini* located in root-application folder. When needed, update this file with appropriate text-editor e.g. Notepad.

You can use a comments in configuration file to create more comprehensible records, but comments cannot be combined with configuration valid values - use them only on otherwise empty lines! All comments must have the prefix '#' or ';' at the beginning of line.

SYSTEM REQUIREMENTS

HTOB Manager requires the following preconditions must be fulfilled:

1. Linux OS with Cron service running.
2. Python2/Python3 environment is installed.
3. Tkniter/tkinter library is installed (based on Python env.version)
4. Linux Cron service is running and /etc/crontab file consists of relevant data.

2 Installation

HTOB Manager GUI application can be hosted in any local folder in Linux filesystem. To install the application copy all needed files into a folder and set permissions to files and nested folders to 755 with *chmod* command.

Default installation folder in Raspberry Pi minicomputer is:

/home/pi/Documents/HTOB

If other root folder is used for HTOB Manager, update the configuration text file *CronManager-config.ini*, section **[APPLICATION]/root_path** with desired path (without last trailing slash), for example:

```
[APPLICATION]
# rooth_path: root folder of CronManager GUI application (in Linux OS)
root_path = /home/pi/Documents/newHTOB
```

Fig.2: Root path for HTOB Manager in configuration file.

2.1 File system description

HTOB Manager files are stored in following file system (default configuration):

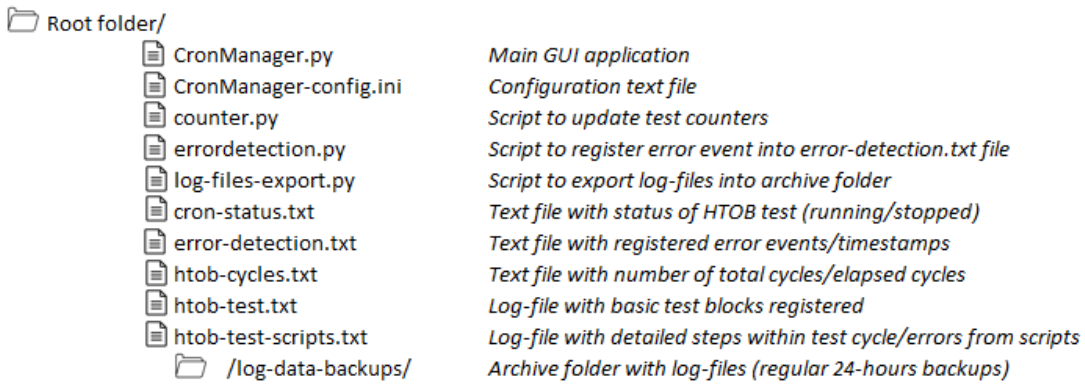


Fig.3: HTOB Manager application filesystem.

3 Counters section

Counters section in HTOB Manager consists of test cycles in number and date/time formats (Fig.4).

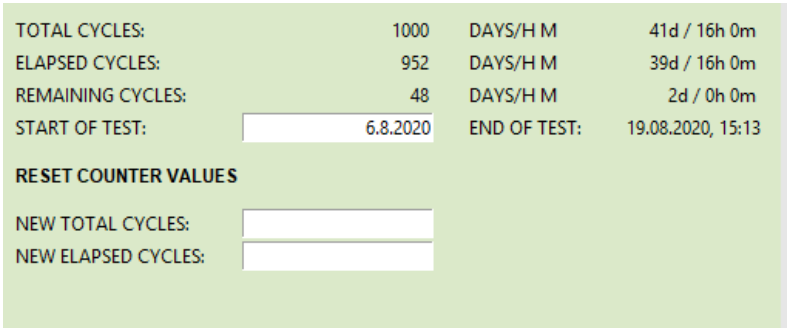


Fig.4: HTOB Manager – Counters section.

START OF TEST (DD.MM.YYYY) – informative field to store a date when HTOB test was started for the first time. It is not used in automated HTOB Manager system for a calculations, it can be updated to any date without an impact on other counter fields.

NEW TOTAL CYCLES – new value for TOTAL CYCLES.

NEW ELAPSED CYCLES – new value for TOTAL CYCLES. This value will be used for automated calculation of END OF TEST date (calculated since actual date & time).

Enter a demanded values into both NEW TOTAL CYCLES and NEW ELAPSED CYCLES fields and click on **<Reset counters>** button in main menu. Fields TOTAL CYCLES and ELAPSED CYCLES will be overwritten with new values and new date in END OF TEST filed will be shown to user.

4 Alarms section

Alarms section consists of several logical parts:

- Actual time (CET), which is synchronized with Eltek NTP servers ❶,
- Status of HTOB test (RUNNING/STOPPED) ❷,
- Alarm indicator based on background colors (see Fig.6). If any error occurs in a script, the red background color appears in section block regardless of actual state of HTOB test (running/stopped).

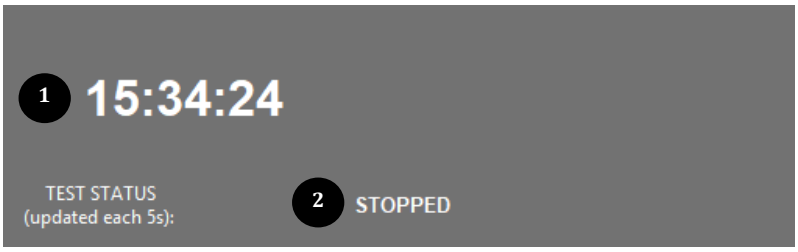


Fig.5: HTOB Manager – Alarms section.

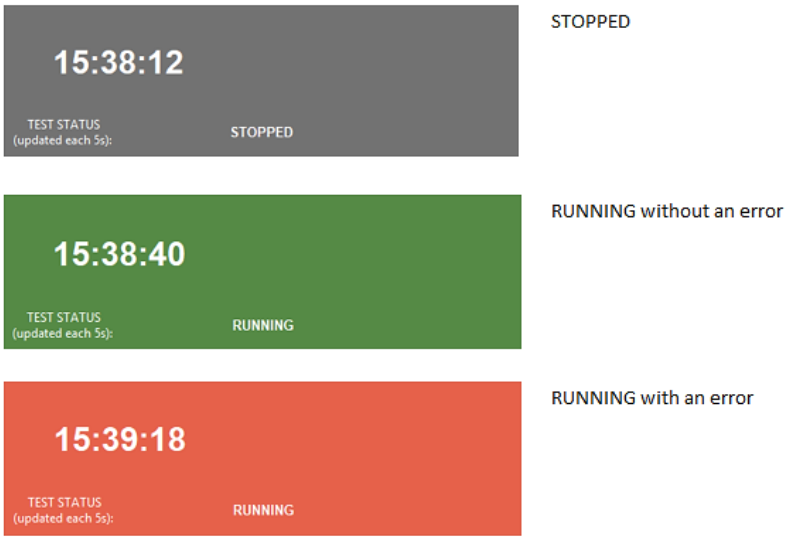


Fig.6: HTOB test states.

Alarms status is refreshed in GUI every 5 seconds and is controlled by text file *error-detection.txt* (Alarm counter), where all alarm events are registered in following format (example):

17.08.2020_15:39:13 - error detected

Based on timestamp at the beginning of line the further analysis should be done with help of log-files *htob-test.txt* and *htob-test-scripts.txt*. To reset Alarm counter (delete all records in *error-detection.txt* file, click on **<Reset errors>** button in main menu.

4.1 Setup Alarm counter to register custom alarms

HTOB Manager user can create own alarm event to register custom alarm and store it in *error-detection.txt* file when alarm is activated. This way any Python code block inside scripts can be chosen to be supervised by HTOB alarm system. Best solution for custom alarms is to use **try-except** Python block for catching alarm events (the try block lets you test a block of code for errors - the except block lets you handle the error). But using this block is not a condition, any Python code block can host this alarm system.

New custom alarm can be inserted into Alarm counter with following proceeding:

1. Add this command to the import section of Python script:

   import errordetection

2. Insert following command to the place inside Python code, where new custom alarm should be generated:

   errordetection.writeError()

Example (code extraction):

```
from socket import *
import time
import errordetection                                    (1)

def sendSocketData(server_ip, server_sock, command):
    global BUFFER_SIZE
    global TCP_PORT


    try:
        server_sock.send(command.encode('utf-8'))
        time.sleep(0.1)
        data = server_sock.recv(BUFFER_SIZE)

    # Error occured when connecting to server. Reconnect and send command again.
    except:
        print("[Socket re-connect due to an uknown exception]")
        errordetection.writeError()                      (2)
        server_sock = socket(AF_INET, SOCK_STREAM)
        server_sock.connect((server_ip, TCP_PORT))
        server_sock.send(command.encode('utf-8'))
```

(1) - external Python module errordetection.py is imported into the script.

❷ - function writeError() in errordetection.py modul is called. It writes down new alarm record into error-detection.txt (Alarm counter) file and consequently new alarm in HTOB Manager is generated.

5 Log-data section

This section consists of control items for handling with log-data files *htob-test.txt* and *htob-test-scripts.txt:*

htob-test.txt - main headlines for all program blocks sorted ascending by the time. User can verify that respective block was triggered in demanded time, e.g.program block to control output contactors - see more in /etc/crontab file e.g.:

```
root echo HTOB_OUTPUT_LOAD_CONTACTOR_ON__$(date +"\%Y-\%m-\%d_\%H:\%M:\%S") >>
/home/pi/Documents/HTOB/htob-test.txt
```

htob-test-scripts.txt - outputs from program routines/scripts sorted ascending by the time. Each record in log-file is generated by Python print() command inside the script code. In addition this file consists of runtime errors from Python scripts thanks to the redirecting the error output to standard output ("2>&1" at the end of commands) - see more in /etc/crontab file e.g.:

```
sudo python /home/pi/Documents/HTOB/HTOB-Output-Load-Contactor-ON.py >> /home/pi/Documents/HTOB/htob-
test-scripts.txt 2>&1
```

All log-data is archived into backup folder in 24 hours interval at 23:59. Backup procedure can be triggered manually as well by clicking on **<Export Logs>** button in main menu. Deafult "log-data-backups" archive folder can be changed with LOG FILES DIRECTORY text field by clicking on **<Choose dir>** button and selecting new folder in File selection modal window (Fig.7).

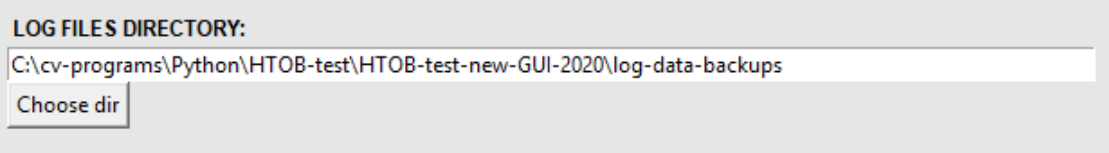


Fig.7: Control fields for selecting new log-files archive folder.

6 Variacs section

Variable AC transformers (Variacs) section provides basic setup for max.4 variacs – requested target AC voltage, output AV voltage for all 3 phases L1, L2, L3.

Under preparation at the time of writing.

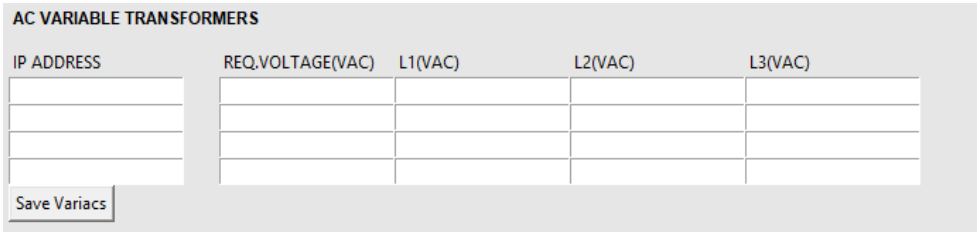


Fig.8: HTOB Manager – Variacs section.

7 Scripts section

In addition of running a scripts automatically by Cron service daemon, any script can be executed manually by the GUI user in Script section.

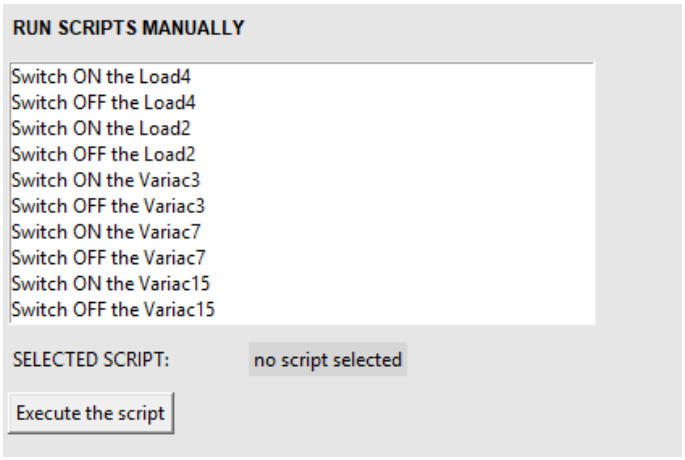


Fig.9: HTOB Manager – Scripts section (after the GUI is started).

List of scripts in Script section is fully under user control. Any script can be added into this list through main configuration file *CronManager-config.ini*, section [MANUAL SCRIPTS]. To add a new script, the following format must be applied for new record in .ini file:

Label to show the script in GUI list = Terminal line command to execute the script

Each script record must be placed in separate line in CronManager-config.ini. Number of scripts is not limited.

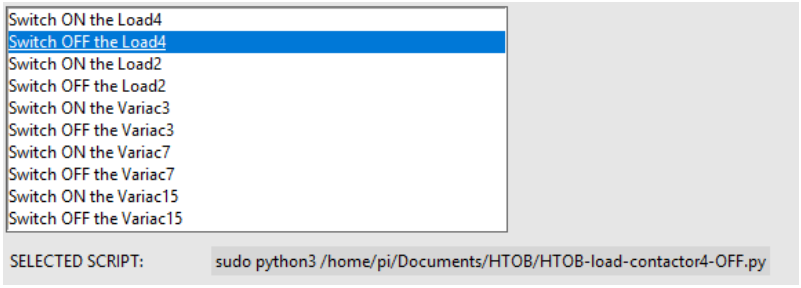
```
[MANUAL SCRIPTS]
# Label to show in GUI interface = command to run the script
Switch ON the Load4 = sudo python3 /home/pi/Documents/HTOB/HTOB-load-contactor4-ON.py
Switch OFF the Load4 = sudo python3 /home/pi/Documents/HTOB/HTOB-load-contactor4-OFF.py
Switch ON the Load2 = sudo python3 /home/pi/Documents/HTOB/HTOB-load-contactor2-ON.py
Switch OFF the Load2 = sudo python3 /home/pi/Documents/HTOB/HTOB-load-contactor2-OFF.py
Switch ON the Variac3 = sudo python3 /home/pi/Documents/HTOB/HTOB-variac3-ON.py
Switch OFF the Variac3 = sudo python3 /home/pi/Documents/HTOB/HTOB-variac3-OFF.py
Switch ON the Variac7 = sudo python3 /home/pi/Documents/HTOB/HTOB-variac7-ON.py
Switch OFF the Variac7 = sudo python3 /home/pi/Documents/HTOB/HTOB-variac7-OFF.py
Switch ON the Variac15 = sudo python3 /home/pi/Documents/HTOB/HTOB-variac15-ON.py
Switch OFF the Variac15 = sudo python3 /home/pi/Documents/HTOB/HTOB-variac15-OFF.py
```

Fig.10: CronManager-config.ini section to add a script for manual execution (example).

To execute a script manually, use the following proceeding:

1. Select desired script in listbox by mouse click.

2. Check in textbox next to SELECTED SCRIPT that expected command is shown:



3. Run the script by clicking on **<Execute the script>** button.

8 Main menu - details

Main menu is located at the bottom of GUI. Main functionalities are described in chapter 1.2 Overview.

8.1 Stop HTOB

This menu item performs immediate stop of HTOB test. The following operations are done:

1. It copies */etc/crontab-stop* file to */etc/crontab* file.
   crontab-stop file consist of no commands to be performed – empty file. This ensures no scripts are executed since current moment.
2. All running scripts related to HTOB test are stopped/killed by command: "sudo pkill -9 -f '\/HTOB-.+\.py'".
   Kill command affects all running processes with filename format "HTOB- ..... .py". **It is strictly required to name all HTOB test scripts to be in line with this convention!**

HTOB SCRIPTS – FILE NAME CONVENTION

It is necessary to name all HTOB scripts with format "HTOB- ..... .py"! Otherwise these scripts cannot be stopped during HTOB test with menu button <Stop HTOB>!

Be aware that scripts without this file name convention will be still running to the end of algorithm and can cause unexpected behavior inside test station/tested samples.

8.2 Show errors

This menu item opens new modal window with content of *error-detection.txt* file (Alarm counter). Alarm counter content provides basic overview about all alarms which arose during execution of HTOB scripts controlled by Cron service (/etc/crontab). For deeper analysis see further log-files *htob-test.txt* and *htob-test-scripts.txt* – look for timestamp close to the Alarm counter record.

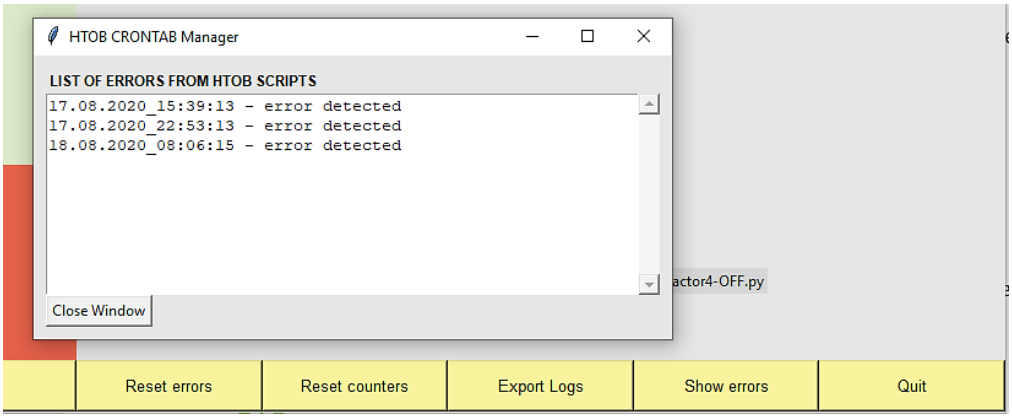


Fig.10: Modal window with Alarm counter content.