Engineering Report

Vapoterm Board Test Application

(VapothermTest.exe)

Prepared for

Vapotherm

By

Enercon Technologies

update: 03-02-2015

Prepared by:

Jim Peterson

Table of Contents

[2. Overview 2](#_Toc403065669)

[Comments 4](#_Toc403065670)

[test <test name> 4](#_Toc403065671)

[type <string> 4](#_Toc403065672)

[desc <string> 4](#_Toc403065673)

[sendline\_a <string> 4](#_Toc403065674)

[sendline\_b <string> 4](#_Toc403065675)

[readline\_a 4](#_Toc403065676)

[readline\_b 5](#_Toc403065677)

[flush\_a 5](#_Toc403065678)

[flush\_b 5](#_Toc403065679)

[expect <field> <min> <max> 5](#_Toc403065680)

[expect\_char <field> <char number> <char> 5](#_Toc403065681)

[expect\_str <field> <string> 5](#_Toc403065682)

[sleep <int - milliseconds> 5](#_Toc403065683)

[prompt <string - question> 5](#_Toc403065684)

[pause <string - comment> 6](#_Toc403065685)

[waitfor <a|b> <int-ms> <string> 6](#_Toc403065686)

[end\_script 6](#_Toc403065687)

[end\_on\_error 6](#_Toc403065688)

# Overview

The Vapotherm board test program (Vapotherm.exe) is a script driven test program designed to test boards mounted in the custom test fixture. The program communicates to both the Vapotherm board as well as a processor built into the test fixture. The communication takes place via serial connections over USB. A single USB connection from the host PC connects to a hub in the fixture that intern provides communication to both devices.

The communication with both devices uses the following serial communication settings:

* Baud Rate: 38400
* Data: 8-bit
* Parity: none
* Stop: 1 bit
* Flow Control: none

Basic program settings are stored in the initialization file VapothermTest.ini which is located in the same directory as the executable. See the section below for details. The tests are driven by a script file. Details of the possible commands in the script file are detailed below.

Initialization File (VapothermTest.ini)

Basic settings are stored from run to run in the initialization file VapothermTest.ini which is located in the same directory as the executable. Some of the settings in this file can be modified via the application’s user interface, while others can only be modified by editing this file. If this file is missing when the application starts, default values will be used and on termination of the application a new file will be created.

The settings controlled by this file are:

* Script – Name of the script file to run
* ReportDir – Directory in which Aegis-style reports will be written
* TerminateOnError – If “true” the tests will be terminated on the first error, otherwise will continue with the next tests after a test fails.
* CheckConnections – When “true” the serial connections are checked before the script is run. The script will only be run if both ports can be opened.
* OutputDelayMS – delay (milliseconds) between characters written to the serial port. This affects I/O on both serial ports.
* TimeoutMS\_A – Timeout (milliseconds) on reads from the board under test.
* TimeoutMS\_B – Timeout (milliseconds) on reads from the fixture processor.
* PortA – Name of the port connected to the board under test.
* PortB – Name of the port connected to the fixture.
* ValidateSerialNumber – when “true” the serial number field validated against the corporate database.
* databaseServer – server on which the database resides
* databaseName – name of the database to check for the serial numbers
* databaseUser – user name to use for logging into the database
* databasePwd – password associated with the database user.
* databaseZNum – the Z-number associated with the product being tested.

There is a user interface in the application for setting the following of these:

* Script
* TerminateOnError
* CheckConnections
* PortA
* PortB

The following is an example of the .INI file:

[General]

Script=C:/Enercon/Vapotherm/work/VapothermTest/VapoThermTestScript.txt

ReportDir=//enxlnk1/Transfer/FunctionalTest/

TerminateOnError=false

[Serial]

CheckConnections=false

OutputDelayMS=120

TimeoutMS\_A=100

TimeoutMS\_B=100

PortA=not connected

PortB=not connected

[Database]

ValidateSerialNumber=false

databaseServer=ENFS3

databaseName=EnerconUtilities

databaseUser=eu\_ro

databasePwd=ET657&me

databaseZNum=Z4001-01

[UnsavedReports]

count=5

report\_0=C:/Enercon/Vapotherm/work/build/debug/Reports/03022015\_084453.txt

report\_1=C:/Enercon/Vapotherm/work/build/debug/Reports/03022015\_082716.txt

report\_2=C:/Enercon/Vapotherm/work/build/debug/Reports/02202015\_142039.txt

report\_3=C:/Enercon/Vapotherm/work/build/debug/Reports/03022015\_083348.txt

report\_4=C:/Enercon/Vapotherm/work/build/debug/Reports/03022015\_085559.txt

Note: Unix-style directory separators (‘/’) are required in any file or directory names, rather than the Microsoft-style (‘\’) separators. This is also true when specifying server names.

Script Format

The script file consists of a list of commands that are sequentially executed. The commands are grouped into “Tests”. The application allows the tests to be independently turned on and off for a given run.

The command parsing is very simplistic. Each line is considered a command. White space at the beginning and ending of the line is ignored. All commands are in lower case. The commands are parsed when the script file is loaded. A warning is generated for any syntactically incorrect or incomplete commands. While you can still run the script file, the flagged commands will be ignored.

## Comments

Any line beginning with “#” is treated as a blank line. Any text following “//” is treated as white space.

## test <test name>

This command signals the beginning of a new test named test. Each test consists of a list of commands. If any one of the commands fails then the test is flagged has having failed. The test name specified will appear in the test list

There are two special named tests that are used for cleanup and shutdown. A test with the name “OnAbort” is run when the operator aborts the script. Similarly, a test with the name “OnExit” will be run on the termination of the script. Neither of these tests will show up in the list of tests.

## type <string>

This command allows the test to be tagged with a descriptive type. This will be displayed in the feedback window as well as the generated reports. This pseudo-command will always succeed.

## desc <string>

Similar to the “type” command this pseudo-command allows a description of the test to be added to the feedback window as well as the generated reports. This command will not generate errors.

## sendline\_a <string>

This command sends the specified string to the serial port associated with port B. The transmission delay specified in the .ini file will be applied between characters. A line termination of carriage-return, line-feed (“\r\n”) is added to the end of the string. After each sendline command a readline command is issued to remove the echoed command that is assumed.

## sendline\_b <string>

Similar to the “sendline\_a” command, this command sends the specified string to the serial port associated with port B.

## readline\_a

This command reads the next line from serial port A. The line of input is terminated by any standard combination of new-line and/or carriage-return. The line read is held in an internal buffer for comparison test by subsequent script commands. The line is discarded when the next readline\_a command is issued.

## readline\_b

This command is the same as readline\_a, but reads from port B.

## flush\_a

This command flushes all I/O from serial port A.

## flush\_b

This command flushes all I/O from serial port B.

## expect <field> <min> <max>

This command tests the range of a numeric field from the last line read. For example, if the last line read was

x-values 123 45.5 101.2

The following commands could be issued:

expect 1 100 150

expect 2 50 100

expect 3 88.8 99.9

Of these the first would pass and the second and third would fail this range test

Note: the field index is 1-based.

## expect\_char <field> <char number> <char>

This command tests a character of a generic field from the last line read. For example, if the last line read was

ABC-flags aazbbcc 01110101 10000000

The following commands could be issued:

expect\_char 1 3 z

expect\_char 2 4 1

expect\_char 3 1 0

Of these the first and second would pass and the third would fail this test.

Note: both the field index and the character index are 1-based.

## expect\_str <field> <string>

This test compares the specified field against the specified string. The test only passes with an exact match. The comparison is case-sensitive.

Note: the field index is 1-based.

## sleep <int - milliseconds>

This command sleeps for the specified number of milliseconds.

## prompt <string - question>

This command presents a Message-Box to the operator with bothYES and NO buttons. This provides the script author with a means or asking a yes/no question of the operator. This command generates an error when the negative response is entered by the operator.

## pause <string - comment>

This command displays a Message-Box with the specified text and an OK button. This gives the script writer an opportunity to provide instructions to the user. This command does not generate errors.

## waitfor <a|b> <int-ms> <string>

This command reads lines from the specified port (A or B), until the specified string is received. The command waits at most for the specified timeout period before continuing. No errors are generated by this command, even if the timeout is exceeded.

## end\_script

This command will terminate the script immediately. This is primarily intended for debugging purposes.

## end\_on\_error

The script will terminate on this error if the previous command was a test that failed.