Test Automation 0.1.1 dev User Guide

Version: 1.1

Revision History

S.N	Change Description	Author(s)	Date
1.	Draft	Chetan Sharma, Harish Bansal	8th September, 2016
2.	Draft	Chetan Sharma	5th June, 2017

Table of Contents

Introduction....4
Archive Contents....4
automation....4
dist....4
docs....4
config....4
requirements.txt....4
timesys-test-project....5
Test Config File....5
Sample Configuration file....6
Timesys Test Runner (ts_test_runner)....7
How to....8
View Test Runner’s supported arguments....8
View Supported Tests....8
Detailed TestCase Information....9
Create new config file....10
Run Tests....10
Display verbose information....10
Provided in Configuration file....10
As Command line argument to Test Runner....12
See Execution logs of old runs....15
Appendix....16
Supported Tests....16
Useful Links....18
How to submit Bugs / improvement ideas....18
Configuration File Format....18
Section....18
Keys....18
Comments....18

Introduction

This release provides a driver testing framework and tool for running Driver tests on remote Boards. This document guide you with procedure to use this tool assuming that your test environment is already setup. If not, than refer to our TestAutomation 0.1.1.dev Setup Guide (Link https://drive.google.com/open?id=0B2zIL4BqlAedVEZtU2ZFUnZhbGs) .

Archive Contents

After downloading Test Automation 0.1.1.dev archive from (Link: https://drive.google.com/open?id=0B2zIL4BqlAedeU5VcHF4VDN3X1U) and uncompressing it on filesystem it would show two top level directories “timesys-test-project” and “automation”.

Here is a brief overview of these folders and their contents

automation

This top level directory contains framework installer and supported documents.

dist

This sub-directory under automation contains Test Automation 0.1.1.dev installer i.e. TestAutomationFramework-0.1.1.dev0-py2.7.egg. It could be installed like any other standard python package using easy_install or pip python package manager.

docs

This sub-directory contains Test Automation 0.1.1.dev setup guide and user guide documents.

config

This sub-directory contains sample configuration files for three boards which could be used for running tests as well as used as reference for writing new configuration files.

Configuration file is a mandatory argument for command link test execution tool (ts_test_runner). These config file contain information about the board on which test will be executed, test server details and a list of test cases to be executed on that board.

requirements.txt

This file contains list of python packages required for running driver testing tool. During the tool setup python’s pip utility reads this file for download and installation of packages. As a user, there is no need to make any changes in this file.

timesys-test-project

This top level directory contains Timesys's driver tests. We have cloned this repository form Timesys git server and using it as it is.

This directory is not required on the Test Server machine. After copying it to board you could delete it from Test Server machine.

Test Config File

Every test execution requires a configuration file*.

Description of configuration file sections and keys.

Section	Key	Key Description
BOARD	name	Board name
	host	IP Address of Board
	port	SSH port of Board
	user	Login username
	password	Login password
	test_suite_path	Directory path of board where compiled timesys-test project is kept. Usually it is "/usr/tstp"
	log_dir	Directory path on board where test execution logs are kept.
TEST_SERVER	ssh_log_file	Filename which contains ssh connection logs.
	test_info_file	**[OPTIONAL]** This file is in json format. It contains information about all testcases which can be executed with TestAutomation. Reference format of file is present in docs/info.json.
	log_dir	Base test execution log directory path on Test Server machine. For each job test execution individual test case logs are generated under **{log_dir}/TimeSys-TestSuite/{BoardName}_{timestamp}/{test_name}** folder.

TESTS	test_list	Array of test cases to be executed. Format: **[** **(test_case1, arguments1),** **(test_case2, arguments2),** **.....** **(test_caseN, argumentsN),** **]** Each array element is a tuple "()" having two strings. First string is the name of test case. For example "CPU_FREQUENCY", second string is test case argument(s). For Example " 0"
	timeout	OPTIONAL key. Number of seconds test runner should wait for individual test case execution to finish before marking its status as TIMED_OUT and start execution of next test case. If not provided in the configuration file, its value is default to 60 seconds.

* Refer to "Configuration File Format" section of Appendix to learn more about the standard configuration file format

Sample Configuration file

[BOARD]

name: BBB

host : 10.42.0.86

port : 22

user : root

password : root

test_suite_path : /usr/tstp/

log_dir : /tmp/logs/

[TEST_SERVER]]

```
ssh_log_file : paramiko_log.txt

test_info_file : automation/docs/info.json

log_dir : /tmp/logs/


[TESTS]

# list tests with ( test script key, args )

test_list : [

        ('LED_TOGGLE',' -c 10 -a'),

        ('GPIO_TOGGLE' , ' -v 1'),

        ('CPU_FREQUENCY', ' 0'),

        ('ADC', '')

    ]

timeout : 60
```

Timesys Test Runner (ts_test_runner)

Its a command line tool provided by framework for test execution and listing out available tests.

Once framework is installed this tool could be run from anywhere in the filesystem.

Overview of arguments supported by test runner tool.

Argument	**Description**
-h, –help	Prints ts_test_runner usage help text
-l –list	Prints list of supported tests on console
-c CONFIG, --config_file=CONFIG	Path of configuration file. One could use his/her own file or one of the sample configuration file provided alogwith test framework.
-t TEST_LIST, --test_list=TEST_LIST	**[OPTIONAL]** Comma “,” separated list of test cases to be executed. **Format**: “test case 1 with arguments, test case 2 with arguments,....., test case N with arguments” Test case name should exactly match with name printed in “**$ts_test_runner -l**” output. Test cases provided by this option overrides test cases supplied in the configuration file. Configuration file test cases wont run.

-i TEST_NAME, --info=TEST_NAME	Prints details information of Test case as Test Name, Script Path, Category, Arguments details, example, preconditions – postconditions, Description, Manual Testing Step if any, Logs information and Notes.
-v, --verbose	**[OPTIONAL]** This option is being used with -t or -c option to display verbose information of Test case before executing test case.

More details for its usage is available in this How to Section.

How to

View Test Runner's supported arguments

From terminal run the following command

$ts_test_runner -h

Usage: test_runner.py [options] arg

Options:
-h, --help show this help message and exit
-c CONFIG, --config_file=CONFIG
Path of config file which contains Tests information
-l, --list List all test cases which are supported
-i INFO, --info=INFO Display information of Test.
-v, --verbose Display verbose information of executing Test.
-t TEST_LIST, --test_list=TEST_LIST
Comma(',') separated list of test cases with arguments
Example :-
For single Test:- "LED_TOGGLE -c 10 -a"
For more than one test:- "LED_TOGGLE -c 10 -a, CPU_FREQUENCY 0"

View Supported Tests

From terminal run the following command

$ts_test_runner -l

List Test Cases with description...

==========

MIC_TEST *: Test mic function on Device.*

I2C_TEST *: Script to execute simple I2C test.*

USBG_M *: Testing the r/w operation on connected SD card by performing MD5 checksum on it.*

WLAN *: Script to test wlan network functionality.*

.....

See Supported Tests section to view list of tests that could be run with current release

Detailed TestCase Information

From terminal run the following command

$ts_test_runner -i <TEST_CASE_NAME>

$ts_test_runner -i MIC_TEST

```
==========================================================================
```

Test Name : MIC_TEST

Test Script Path : bin/audio/test_mic.sh

Test Category : Audio

Test Arguments : hw:0,0 target/audio/500Hz_48000.wav

Arguments Detail:

Needs 2 inputs as:

1. Playback Device

(check with aplay -l)

2. Audio file name with location

Example test entry in configuration file

test_list : [('MIC_TEST', '')]

Test Precondition : Connect the front panel OR connect a speaker to headphone/lineout

Listen for a 500Hz tone on the speaker

Test Description : Tests mic function on Device.

Test Postcondition : 500Hz tone is heard on the speaker

Manual Testing Steps(if any) :

Logs Interpretation :

Notes :

```
==========================================================================
```

Create new config file

<extraction_directory>/automation/config directory contains few sample configuration files. Simply copy the one of choice and save it with a different name.

- **$ cd TestAutomation0.1.1.dev/automation/config**
- **$ cp bbb.cfg my_config.cfg**

Open up your new config file(my_config.cfg) in any editor of choice and modify Board, and Tests sections as per your test environment and test execution needs.

Run Tests

Test runner supports two different modes of supplying test cases to be executed.

Display verbose information

This option is optional and is used in-addition of execution of testsuite to display detailed information about testcase which is going to execute.

*$*ts_test_runner -v -c ~/bbb_1.cfg

Follow this link for more Detailed information about this option

Provided in Configuration file

- Edit test_list key value in the configuration file to specify what test cases you would like to execute and their test parameters.
- *Run Command with verbose: **$ts_test_runner -v -c <path of your configuration file>**
  - For example *$*ts_test_runner -v -c ~/bbb_1.cfg

Test Runner Started ...

Executing Tests..

==========

Test Name : CPU_FREQUENCY

Test Script Path : bin/cpu/test_cpufreq.sh

Test Category : CPU

Test Arguments : 0

Arguments Detail:

CPU number on which frequency scaling needs to be performed

Example test entry in configuration file

test_list : [('CPU_FREQUENCY', '')]

Test Precondition :

Test Description : Tests CPU frequency scaling for CPU with a range of available CPUFreq Governors and frequencies

Test Postcondition : CPU core has booted and has CPU frequency scaling enabled for CPU0

Manual Testing Steps(if any) :

Logs Interpretation : Results CPU frequency scaling for CPU core with a range of available CPUFreq Governors and frequencies

Notes :

```
======================================================================

********** TEST RESULTS **********
    Test Name  |       Argument(s)  |  Result Status
==========================================================
CPU_FREQUENCY    |              0  |  SUCCESS
==========================================================
```

Detailed Logs for all test cases are present in directory /tmp/logs/TimeSys-TestSuite/BBB_19700101001231

* Before running tests make sure log directories(provided in the configuration file) are present on both board and test server. Also Linux user running ts_test_runner should be having WRITE permissions for log directory , else run this command with “sudo” prefix.

- *Run Command without verbose: **$ts_test_runner -c <path of your configuration file>**
  - For example *$*ts_test_runner -c ~/bbb_1.cfg

```
Test Runner Started ...
Executing Tests..

********** TEST RESULTS **********
    Test Name   |        Argument(s)   |   Result Status
============================================================
CPU_FREQUENCY    |              0   |   SUCCESS
============================================================
Detailed Logs for all test cases are present in directory /tmp/logs/TimeSys-TestSuite/BBB_19700101001231
```

* Before running tests make sure log directories(provided in the configuration file) are present on both board and test server. Also Linux user running ts_test_runner should be having WRITE permissions for log directory , else run this command with “sudo” prefix.

As Command line argument to Test Runner

Test Runner also supports passing test cases from command line. These tests overrides test cases mentioned in the config file under **TESTS >>test_list** value.

- *Run Command: **$ts_test_runner -v -c <path of your configuration file> -t “comma(,) separated list of test cases and their arguments”**
    - For example
        - For Single test case with verbose: *$ts_test_runner -v -c ~/my_config/bbb.cfg -t “LED_TOGGLE -c 10 -a”*
        - For more than 1 test cases with verbose: *$ts_test_runner -v -c ~/my_config/bbb.cfg -t “GPIO_TOGGLE -v 32, CPU_FREQUENCY 0”*

```
 Test Runner Started ...
Executing Tests..
======================================================================
Test Name                : CPU_FREQUENCY

Test Script Path         : bin/cpu/test_cpufreq.sh

Test Category            : CPU

Test Arguments           :  0
```

Arguments Detail:

CPU number on which frequency scaling needs to be performed

Example test entry in configuration file

test_list : [('CPU_FREQUENCY', ' 0')]

Test Precondition :

Test Description : Tests CPU frequency scaling for CPU with a range of available CPUFreq Governors and frequencies

Test Postcondition : CPU core has booted and has CPU frequency scaling enabled for CPU0

Manual Testing Steps(if any) :

Logs Interpretation : Results CPU frequency scaling for CPU core with a range of available CPUFreq Governors and frequencies

Notes :

==========

==========

Test Name : GPIO_TOGGLE

Test Script Path : bin/gpiolib/gpio-toggle.sh

Test Category : GPIO

Test Arguments : -v 32

Arguments Detail:

2 input:

1. Verbose on (shows commands being run)

2. GPIO pin number to toggle

Example test entry in configuration file

test_list : [('GPIO_TOGGLE', ' -v 32')]

Test Precondition :

Test Description : Tests GPIO on board by toggle one-another pin

Test Postcondition :

Manual Testing Steps(if any) :

Logs Interpretation : Output logs shows the toggling of backlight values for gpio pin provided as input.

Notes :

```
======================================================================
********** TEST RESULTS **********
 Test Name          |      Argument(s)  |  Result Status
 ============================================================
GPIO_TOGGLE        |           -v 32   |  SUCCESS
CPU_FREQUENCY  |                 0  |  SUCCESS
============================================================
```

Detailed Logs for all test cases are present in directory /tmp/logs/TimeSys-TestSuite/BBB_19700101001231

** Before running tests make sure log directories(provided in the configuration file) are present on both board and test server. Also Linux user running ts_test_runner should be having WRITE permissions for log directory , else run this command with "sudo" prefix.*

- For more than 1 test cases without verbose: *$ts_test_runner -c ~/my_config/bbb.cfg -t "GPIO_TOGGLE -v 32, CPU_FREQUENCY 0"*

```
 Test Runner Started ...
Executing Tests..
********** TEST RESULTS **********
 Test Name          |      Argument(s)  |  Result Status
```

```
 ============================================================
GPIO_TOGGLE         |            -v 32    |  SUCCESS
CPU_FREQUENCY   |                   0   |  SUCCESS
============================================================
```

Detailed Logs for all test cases are present in directory /tmp/logs/TimeSys-TestSuite/BBB_19700101001231

** Before running tests make sure log directories(provided in the configuration file) are present on both board and test server. Also Linux user running ts_test_runner should be having WRITE permissions for log directory , else run this command with "sudo" prefix.*

See Execution logs of old runs

For each test job execution, framework create a new unique sub-directory ({Board_name}_{timestamp}) under {log_dir}/TimeSys-TestSuite.

Where log_path is the base Test Server log path given in "TEST_SERVER" section and "log_path" key value in the test configuration file.

This {Board_name}_{timestamp} directory contains

- test_summary.txt -> Test execution summary report
- configuration file used for test execution
- Sub-directories for each executed test case. These sub-directories stores each corresponding test case execution logs.

Appendix

Supported Tests

S.N.	Test Case	Test case description
1	MIC_TEST	Test mic function on Device.
2	I2C_TEST	Script to execute simple I2C test.
3	USBG_M	Testing the r/w operation on connected SD card by performing MD5 checksum on it.
4	WLAN	Script to test wlan network functionality.
5	USBH_M	Testing the r/w operation on mounted USB2.0 by performing MD5 checksum on it.
6	ACCELEROMETER	Script to test accelerometer function of IO connected to device
7	IMX_ADC	Script to test ADC function on hardware by monitoring.
8	SERIAL_TEST	Script to execute simple SERIAL test on device
9	USB_TEST	Test r/w operation of inserted USB device
10	FS_TEST	Testing the r/w operation on connected SD card by performing MD5 checksum on it.
11	BUTTON_TEST	Script to test button function
12	TEST_USB	Script to test USB device
13	LINE_TEST	Test Line functionality of speaker
14	TEST_NAND	Test for erasing and read write operation over a partition of MTD flash.
15	SPEAKER	Testing ALSA soundcard driver by playing the input audio file.
16	ADC	Test simple ADC function on board
17	GPIO_TOGGLE	Script to test GPIO on board by toggle one-another pin
18	RTC_TEST	Script to test RTC on device
19	ETH	Test script to test etho functionality
20	I2C_READ	Testing for read registers visible through the I2C bus (or SMBus).
21	TEST_LOOPBACK	Script to test Loopback function on device.
22	WESTON	Test script which starts timesys-test-project

23	PIPELINE	Script to test pipe lines on Board
24	BLK_READ	Script to test bulk read using i2c interface.
25	PCIE_ENUM	Script to test PCIE bus device connected to device
26	BACKUP_FILE	Script to test BACKUP file copy in USB.
27	SPIDEV_TEST	Script to execute simple spidev test.
28	WATCHDOG_TEST	Script to test watchdog function on device
29	MDIO	Script to test mdio function on device
30	WATCHDOG_SIMPLE	Script to test Simple function of Watchdog
31	UART_LOOPBACK	Script to test UART Loopback on device
32	NOR_FLASH	Test for erasing and read write operation over a partition of NOR flash.
33	CPU_FREQUENCY	Testing CPU frequency scaling for CPU with a range of available CPUFreq Governors and frequencies
34	USBG_M_S	Script to setup usb m storage
35	EEPROM	Testing the r/w operation on input device with page size by performing MD5 checksum on it.
36	LED_TOGGLE	Script to test LED function by toggle one-another
37	ALTIMETER	Script to test altimeter functions on board
38	I2C_SEND	Script to test send command on i2c devices
39	BACKLIGHT	Script to test backlight function on Board
40	X11_TEST	Testing Display of input image file on x11 system.
41	I2C_MCP	Testing i2c read and write operations for mcp23008 module.
42	SPIDEV_MCP	Testing MCP device testing with spidev.
43	FB_TEST	Testing Framebuffer using fb-demo utility.
44	UBI_TEST	Testing jffs filesystem of storage device.
45	ACCELEROMETER_ST	Testing ST LIS3LV02D accelerometer function of IO connected to device.

Useful Links

Framework Download Link: https://drive.google.com/open?id=0B2zIL4BqlAedeU5VcHF4VDN3X1U

Linux Link: http://linuxlink.timesys.com/

How to submit Bugs / improvement ideas

We are using Jira for defect management. You could file your identified Bugs/improvement suggestions/new feature requests under "**Test Automation**" project, "**Driver Testing Framework**" epic.

Epic Link: https://timesys.atlassian.net/browse/TA-114

Configuration File Format

Configuration files are written in INI/CFG format. It consist of 3 different elements as sections, keys and comments.

Section

The section name appears on a line by itself, in square brackets "[ ]" . All keys after the section declaration are associated with that section. There is no explicit "end of section" delimiter, sections end at the next section declaration or end of file. Sections may not be nested.

Keys

The basic element contained in an config file is the *key* or *property*. Every key has a *name* and a *value*, delimited by an equal sign (=) or colon (:).

Comments

Semicolons (;) and hash (#) at the beginning of the line indicate a comment. Comment lines are ignored.