ST-ERICSSON

ITE NMF 8500

Release Documentation

A basic user guide for developing and releasing test cases for ITE_NMF_8500. This document provides a basic overview on how to write test cases for effective test report generation in ITE_NMF_8500.

Legal Information

Disclaimer

The contents of this document are subject to change without prior notice. ST-Ericsson makes no representation or warranty of any nature whatsoever (neither expressed nor implied) with respect to the matters addressed in this document, including but not limited to warranties of merchantability or fitness for a particular purpose, interpretability or interoperability or, against infringement of third party intellectual property rights, and in no event shall ST-Ericsson be liable to any party for any direct, indirect, incidental and or consequential damages and or loss whatsoever (including but not limited to monetary losses or loss of data), that might arise from the use of this document or the information in it.

ST-Ericsson and the ST-Ericsson logo is trademarks of the ST-Ericsson group of companies or used under a license from STMicroelectronics NV or Telefonaktiebolaget LM Ericsson.

All other names are the property of their respective owners.

Trademark List

All trademarks and registered trademarks are the property of their respective owners.

In addition to the generic statement above, please fill in all trademarks and registered trademarks that could be identified.

See examples below.

Microsoft® Microsoft is a registered trademark of Microsoft Corporation in the United States and/or other countries.

OBEX™ OBEX is a trademark of Infrared Data Association.

Table of Contents

Purpose of this document

This document provides a basic overview on how to write test cases for effective test report generation in ITE_NMF_8500.

Scope of this document

The scope of this document is limited to the engineers, who will work on imaging sensor development.

Revision History

Please keep the latest version on top

Ver	Change Description	Sections	Date	Author	Reviewer
0.1	All	All	09-02-2011	Sudeep KS	Atul Gupta, Partha Mitra

References

Reader should read this document in conjunction with the following documents

No	Document Name	Ver	Location
1.	TestCase_description_ite_nmf_8500	0.2	STE Camera SDK
2.	TestCase_description_ite_nmf_8500	0.5	ITE_NMF_8500
3.	ISP8500_FW_User_Manual	FW version	ISP FW Documentation

1 Overview

This document provides a basic overview on how to write test cases for effective test report generation in ITE_NMF_8500. The tests are divided into two broad categories

- Non Regression
- Functional Tests

These test scripts are executed using ite_nmf_8500's **'do'** command and generate **'*.out'** log files with Pass / Fail results. These logs are subsequently used to generate reports. ITE_NMF_8500 requires MMTE support.

2 MMTE TEST API's And MACROS

MMTE API's are used for generating .out files which are used for test report generation. They are explained in brief.

MMTE TEST API 's :

1. mmte_testStart(filename, description, g_out_path)

- This is used for starting the test case
- Filename : Name of the .out file to be generated
- Description : Here we mention a brief description of the test case
- g_out_path : This is the path where the .out file generated will be stored

2. mmte_testNext(comment)

- Comment on the subtest case will be reflected in the report

3. mmte_testComment(comment)

- Comment this will be reflected in the report

4. **mmte_testResult(TEST_PASSED/TEST_FAILED/TEST_SKIPPED)**

- This gives the result of the test

5. **mmte_testEnd()**

- This is used for ending the test case

MMTE TEST MACROS:

1. MMTE_TEST_START(name, path, description)

- Name :Filename
- Path : Path where .out file is to generated.
- Description: Here we mention a brief description of the test case

2. MMTE_TEST_NEXT(comment)

- Comment this will be reflected in the report and printed on the window

3. MMTE_TEST_COMMENT(comment)

- Comment this will be reflected in the report and printed on the window

4. MMTE_RESULT_PASSED/ MMTE_RESULT_FAILED/ MMTE_RESULT_SKIPPED ()

- This is used for results of subtests

5. MMTE_TEST_PASSED/ MMTE_TEST_FAILED/ MMTE_TEST_SKIPPED ()

- This is used for final test result

NOTE: For all functional tests we make use of mmte test macros and for non regression testing we make use of mmte test api's .All comments will be recorded but on the HTML report we can see comments only if they are preceded by **MMTE_TEST_NEXT/mmte_testNext.**

We also must ensure that the path we are giving is valid and it has the valid folder in the test environment.

3 MMTE TEST API USAGE IN NON REGRESSION TESTS

The API's explained in the Section 2 has to be used in a particular way to get precise results and comments on the test report.

We must prefix sensorName as "PrimaryCam" or "SecondaryCam" to the filename and then we use **mmte_testStart()**.

All test case must start with **mmte_testStart().** If there are no subtests involved we can directly use **mmte_testResult() and mmte_testEnd().**

If we have one or more subtests involved then after with **mmte_testStart()** we need to use . **mmte_testNext() and then use mmte_testResult()** once we reach the end of all subtest cases we have to use **mmte_testEnd().**

mmte_testComment()can be used anywhere without any restriction, but it should always be used after the first **mmte_testNext()** in a test case.

Once we use **mmte_testEnd()** we must set the g_out_path to 0 as shown below

memset (g_out_path, 0, KlogDirectoryLentgh*sizeof (char))

4 MMTE TEST MACROS USAGE IN FUNCTIONAL TESTS

The MACROS explained in Section 2 has to be used in a particular way to get precise results and comments on the test report. These macros have built in checks and will display test state errors if they are used in a wrong order.

All functional test cases must start with **MMTE_TEST_START.** If there are no subtests involved we can directly use **MMTE_TEST_PASSED**, **MMTE_TEST_FAILED** or **MMTE_TEST_SKIPPED** as the need arises.

We must ensure that **MMTE_TEST_COMMENT** is used wherever there is an information we need to see in the report ,also we need to use this macro to convey the reason test fails i.e., we must use this before **MMTE_TEST_PASSED/ MMTE_ TEST _FAILED/ MMTE_ TEST _SKIPPED** is used.

If we have one or more subtests involved then after **MMTE_TEST_START we have to use MMTE_TEST_NEXT** which should be followed by use **MMTE_RESULT_PASSED**, **MMTE_RESULT_FAILED** or **MMTE_RESULT_SKIPPED. MMTE_TEST_COMMENT** can be used anywhere without any restriction, but it should always be used after the first **MMTE_TEST_NEXT** in a test case.

We must ensure that every intermediate and final result should be preceded by a **MMTE_TEST_COMMENT.** This is done to ensure that we get clear cut comment s on the HTML report.

5 Adding New Sensor Support

We have to do the following changes to add support to new sensor in ITE NMF 8500

1. Insert sensor configurations with the switch as sensor id in ite_sia_bootcmd.cpp
2. Insert the new .ini file for the sensor in sensorinformation folder
3. Insert the new sensor id with the .ini file name in ite_sensorinfo.c
4. If sensor does not support auto detection add the following code in ite_sensorinfo.c in function ITE_autoDetectSensors

```
If(0 == strcmp(g_sensor_select,"NEW_SENSOR"))
 {
   discoveredID = XXX;
   discoveredRevision = XXX;
  }
```

5. Once everything is set up as soon as ite_nmf_8500 is loaded we must give the command

```
sensorname <sensor_name>
```

6 ITE NMF 8500 BRANCHES

We have one main TRUNK and branching is done as and when a new multimedia release is made in entirety .As of now we have 3 branches namely

- 11.12_7 : This branch has a WK12 base and supports all 7 series FW.It supports Symbian, Linux and Android environment
- 11.12_3: This branch has a WK12 base and supports all 3 series FW. It supports Android and Linux environments
- 11.30:This branch is based on WK30 base and supports all 7series FW. It supports Symian, Linux and Android environments.

When we modify ITE NMF 8500 ideally we have to update the latest branch (as of now wk 30 11.30)and then deliver the code to trunk. In other words the trunk and the latest branch should always be in sync. We will maintain WK12 FW 7 series (11.12_7) branch till we get XTI traces runnin on WK30 branch.

7 ITE NMF 8500 RELEASE PROCEDURE

Step by step procedure for releasing ITE NMF 8500 into trunk.

1. **Download latest ite_nmf_8500 from working branch**

   An example of Component to be used is shown below

ite_nmf_8500
[sb::svnroot=http://codexstn.cro.st.com/svnroot/siavalid/ite_nmf_8500,path=imaging/ite_nmf_8500,branch=releases/11.12_7/trunk parentbranch=trunk]

Then we do a rodos uninstall to remove any old component

rodos uninstall –c ite_nmf_8500

then

rodos install –c ite_nmf_8500

2. **Do all changes required**

Once all the changes are updated update about.txt with the version number of ite_nmf_8500in repo folder along with the required baseline and extension.h files and component files. Attach report.html , It can be a functional test report if functional test added else non regression test report. Update Test case description document and release_notes in docs folder and release_notes in component folder.

3. **Commit these changes to branch**

svn status

if we have any files with "?" we have to add these files using *"svn add path from ite_nmf_8500/file/foldername"*

svn ci –m "message /reason for commit"

svn update This is done to bring the latest status from branch to your local copy befor publishing

4. **Publish The code**

Inside folder ite_nmf_8500 we give the following commands

rodos publish -p none -c ite_nmf_8500

5. **Deliver it to trunk**

sb deliver ite_nmf_8500 : Deliver's the code to main trunk.