**Unit Test Plan**

**For**

**< PSAStHdlr\_CF13>**

**Prepared For:**

**Product Engineering**

**Nexteer Automotive**

**Saginaw, MI,**

**United States of America**

**Document Identifier: <Project\_ID>\_<Config Id>**

**VERSION: 3.0**

**DATE: 14-Mar-2016**

**Prepared By:**

**Location:** The official version of this document is stored in the Nexteer Configuration Management System.

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Description** | **Prepared/ Modified By** | **Version** | **Date** | **Approved By** |
| 1 | Initial version | KPIT | 1.0 | 1/30/2015 | Lonnie Newton |
| 2 | Updated for MIL approach | KPIT | 2.0 | 8/19/2015 |  |
| 3 | Updated for NonSimulating approach | KPIT | 3.0 | 12/17/2015 |  |

**Table of Contents**

[1 Abbreviations And Acronyms 5](#_Toc389223980)

[2 References 6](#_Toc389223981)

[3 Introduction 7](#_Toc389223982)

[3.1 Purpose 7](#_Toc389223983)

[3.2 Scope 7](#_Toc389223984)

[4 Test Automation 8](#_Toc389223985)

[5 Unit Test Entry / Exit Criteria 9](#_Toc389223986)

[5.1 Entry Criteria 9](#_Toc389223987)

[5.2 Exit Criteria 9](#_Toc389223988)

[6 Suspension And Resumption Criteria 10](#_Toc389223989)

[6.1 Suspension Criteria 10](#_Toc389223990)

[6.2 Resumption Criteria 10](#_Toc389223991)

[6.3 Approval Criteria 10](#_Toc389223992)

[7 Assumptions 11](#_Toc389223993)

[8 Unit Testing Procedure 12](#_Toc389223994)

[8.1 Testing Method 12](#_Toc389223995)

[8.2 Unit Test Preparation 12](#_Toc389223996)

[9 Regression Strategy 1](#_Toc389223997)7

[10 Test Deliverables 14](#_Toc389223998)

[11 Communication Approach 15](#_Toc389223999)

[12 Appendix 16](#_Toc389224000)

**List Of Tables**

No table of figures entries found.

**List Of Figures**

**No table of figures entries found.**

# Abbreviations And Acronyms

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| GCC | Gross Cross Check |
| MDD | Module Design Document |
| FR | Functional Requirement |
| DD | Data Dictionary |
| UTT | Unit Test Tracker |
| UT | Unit Test |
| CM | Configuration Management |
| SVN | Software version |
| CR | Change Request |
| MIL | Model In Loop |
| SIL | Software In Loop |
| PIL | Processor In Loop |
| EPS | Electrical Power Steering |
| MCC | Modified Condition Coverage |
| MC | Modified Condition |
| DC | Decision Coverage |
| ASIL | Automotive Software Integrity Level |

# References

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Version** |
| 1 | Unit test guidelines | 1.0 |
| 2 | Task readiness checklist | 1.1 |
| 3 | Task delivery checklist | 1.8 |
| 4 | Peer review checklist | 4.0 |
| 5 | Compiler and Linker settings document | 1.0 |
| 6 | Software environment setup document | 1.0 |
| 7 | Hardware environment setup document | 1.0 |
| 8 | Unit test workspace folder structure template | 1.0 |

# Introduction

This document describes the unit Test Plan for the module “**PSAStHdlr**”.

## Purpose

The purpose of this document is to capture various aspects involved in performing the unit testing for the component “**PSAStHdlr**”. This document captures the list of functions of the components that will be unit tested and strategy applied to create test vectors, execute test vectors, generate unit test report for assessment.

## Scope

The scope of this testing would be to Unit test the module “**PSAStHdlr**” with respect to the ISO 26262 compliance and generate the detailed report so that each and every aspect of Unit testing is covered.

### Functions To Be Tested

Following are the functions that are to be tested in the Module:

1. **ComputeApa**
2. **ComputeApaAllw**
3. **ComputeApaDrvrIntv**
4. **ComputeApaFltActv**
5. **ComputeApaHwAgCtrlErr**
6. **ComputeApaLimits**
7. **ComputeApaState**
8. **ComputeApaState\_ChildTransitions**
9. **ComputeApaState\_ParentTransitions**
10. **ComputeEPSState**
11. **ComputeEPSStateForLxa**
12. **ComputeEPSStateForLxa\_ChildTransitions**
13. **ComputeEPSStateForLxa\_ParentTransitions**
14. **ComputeLxaDrvrAbsntDetd**
15. **ComputeLxaDrvrBhvr**
16. **ComputeLxaHowDetnTime**
17. **ComputeLxaHwAgCmd**
18. **ComputeLxaHwAgThd**
19. **ComputeLxaHwVelThd**
20. **ComputeLxaInactivOnDrvrBhvr**
21. **ComputeLxaInactivRstTmr**
22. **ComputePosSrvoEnable**
23. **LxaHwAgMon**
24. **LxaHwTrqMon**
25. **LxaHwVelMon**
26. **ComputeLxaCorrnFacReqLimd**
27. **PSASH\_Init1**
28. **PSASH\_Per1**

### Functions Not To Be Tested

Following are the functions which are not tested as they are Inline Functions, and we do not Unit test them.

1. **Abslt\_f32\_f32**
2. **Abslt\_u08\_s08**
3. **Abslt\_u16\_s16**
4. **Abslt\_u32\_s32**
5. **Arctan2\_f32**
6. **Blnd\_f32**
7. **Rint**
8. **Cos\_f32**
9. **Exp\_f32**
10. **Lim\_f32**
11. **Lim\_s08**
12. **Lim\_s16**
13. **Lim\_s32**
14. **Lim\_u08**
15. **Lim\_u16**
16. **Lim\_u32**
17. **Max\_f32**
18. **Max\_s08**
19. **Max\_s16**
20. **Max\_s32**
21. **Max\_u08**
22. **Max\_u16**
23. **Max\_u32**
24. **Min\_f32**
25. **Min\_s08**
26. **Min\_s16**
27. **Min\_s32**
28. **Min\_u08**
29. **Min\_u16**
30. **Min\_u32**
31. **Mod\_f32**
32. **FixedToFlaoat\_f32\_s16**
33. **FixedToFlaoat\_f32\_s32**
34. **FixedToFlaoat\_f32\_u16**
35. **FixedToFlaoat\_f32\_u32**
36. **FixedToFlaoat\_s16\_f32**
37. **FixedToFlaoat\_s32\_f32**
38. **FixedToFlaoat\_u16\_f32**
39. **FixedToFlaoat\_u32\_f32**
40. **FloatToFixdWithRound\_s16\_f32**
41. **FloatToFixdWithRound\_s32\_f32**
42. **FloatToFixdWithRound\_u16\_f32**
43. **FloatToFixdWithRound\_u32\_f32**
44. **Sign\_s08\_f32**
45. **Sign\_s08\_s08**
46. **Sign\_s08\_s16**
47. **Sign\_s08\_s32**
48. **Sin\_f32**
49. **Sqrt\_f32**

### Unit test objective

1. Defining test vectors
2. Perform  unit testing as defined in the unit testing guideline document
3. Identify range issues
4. Identify overflow/underflow
5. Identify coverage issue
6. Uncover dead code or redundant code by trying to achieve 100% code coverage
7. Identify requirement conformance issue [ Dependency on well define FR and well documented Design]
8. Identifying the deviation (MDD, SRC, and DD) or any other bug/Anomaly in the code.
9. Generate Output of UT in the form of the report generated from TESSY
10. Compute the Memory and throughput metrics for the components submitted for unit testing
11. Generate test traceability document.

# Test Automation

Following are the softwares to be used for the unit testing of the Module “**PSAStHdlr**”:

1. Nexteer UTT:

The Nexteer Automotive Unit Test Tool (UTT) is an Excel based development tool that provides an automated interface for testing the C-code modules used in electronic control units (ECUs). The UTT allows the user to easily add input values and expected output values into auto-generated Excel worksheet templates. A C-code 'wrapper' is generated and merged with the C-code module under test.

Version of the Nexteer UTT: 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Name of the Sheet** | **Abbreviation** | **Description** |
| 1 | Unit Test Information | - | This is the front sheet of the unit test vector sheet consist of the details of the Tester, the module under test and version number of various files |
| 22 | Module Definitions | - | This sheet consist of the functions under test the input and Output Variables and the stub Functions used in the module |
| 23 | Unit Test Setup (PSASH\_Per1) | - | This sheet will have the details regarding the test types ,input and Output Variables and the stub Functions used for the function under test |
| 4 | PSASH\_Per1() R | Range sheet | This sheet will consist of the Range test Vectors for Pim variables |
| 5 | PSASH\_Per1 () P | Path Sheet | This sheet will consist of the Path test Vectors |
| 6 | PSASH\_Per1() PIM | PIM Range sheet | This sheet will consist of the Range test Vectors for Pim variables |
| 7 | PSASH\_Per1() PRM | PRM Range sheet | This sheet will consist of the Range test Vectors for Prm variables |
| 8 | Unit Test Setup (PSASH\_Init1) | - | This sheet will have the details regarding the test types ,input and Output Variables and the stub Functions used for the function under test |
| 9 | PSASH\_Init1() PRM | PRM Range sheet | This sheet will consist of the Range test Vectors for Prm variables |

# Unit Test Entry / Exit Criteria

## Entry Criteria

Onsite Entry Criteria:

* The code is frozen and contains complete functionality
* The code review is completed and code review defects are closed
* The source code is checked in SharePoint (Such as SVN)
* Unit test CR is created in the change manager
* Unit test CR version matches the CM versions
* All the required support files are present to initiate the unit testing.
* The design is baselined and available for UT
* The requirement is baselined and available for UT

Offshore Entry Criteria:

* Input folder uploaded in the common share point(Such as SVN):
* Unit testing initiation intimation E-mail with details of unit test CR number, the source version, component version etc. is received.
* The Input folder may consist of the following required files depending upon the approach:
* The Source code for which unit testing is to be initiated.
* The Data Dictionary and the Module Design document.
* The Contract folder, which will consist of all the Dependent files.
* The requirement document, the design document, the test traceability matrix etc.
* The Simulink Model and its dependent files.
* The MIL test data.
* No mismatch in source code and the MDD [ performed as part of Sanity Check, if any mismatches is found then clarification E-mail will be sent]
* The design is baselined and available for UT
* The requirement is baselined and available for UT
* The design and code traceability is correct.
* Task readiness E-Mail marks the Initiation of Unit testing of module, and it is sent if the following conditions are met:
* There is no mismatches in the Source code and MDD
* The requirements are properly captured in design and source.
* The ranges of all the Variables used in the module is available.
* The Module is open able in TESSY.
* The Simulink Model is executable.

## Exit Criteria

Onsite Exit Criteria:

* All known problems[failed vectors, low coverage,etc] posted to the defect-tracking system
* All unit test cases are executed
* The version number of the report files consistent with the submitted files.
* Results of executed tests are discussed with project team
* Requirement traceability is achived.
* There are no showstopper defect.
* The change manager CR moved to verification state

Offshore Exit Criteria:

* Any failing vector status is communicated to developer and required defect tracker is updated.
* Peer review is performed to identify the defects in delivery and required action is taken to address the identified issues.
* All the identified delivery items are properly placed on to the delivery packet
* Coverage analysis is performed if 100% coverage is not achievable and required tessy snap shots are provided.
* Memory and Throughput analysis is performed
* Clarification sheet is updated for all the clarifications obtained.
* Unit test Task details are captured and documented, this will be used to communicate as part of this CR what activities were performed.
* Traceability is verified to make sure all the implemented requirements are tested and clarification for all the non-testable requirements are documented.
* Naming convention of the delivery files are verified.
* All the related artifacts moved to strong store (CM) for the CR.

# Suspension And Resumption Criteria

## Suspension Criteria

The unit test shall be suspended in case if there are any dependencies due to which unit test can’t be proceed further. For ex: any missing variable range will require developer to provide the clarification on the range before unit testing is proceeded further.

If there is any missing documents unit testing shall be suspended till the dependent documents are obtained.

The unit testing shall be suspended if there are any hardware/tool issue is observed.

The unit testing shall be suspended if higher version of the same component is submitted for the unit testing and prior version unit testing is rejected. [Confirmation shall be obtained from the developer]

The unit testing shall be suspended if high priority task overrides the current unit testing activity.

## Resumption Criteria

The unit test shall be resumed in case if missing variable ranges are provided.

The unit test shall be resumed if missing documents for unit testing are provided.

The unit testing shall be resumed if any observed hardware/tool issue is resolved.

The unit testing shall be resumed if higher version of the same component is submitted for the unit testing and prior version unit testing is asked to continue. [Confirmation shall be obtained from the developer]

The unit testing shall be suspended if high priority task overrides the current unit testing activity.

## Approval Criteria

Communication mail shall be retained for all the suspension and resumption cases. The suspension communication shall detail the reason for suspension with possible resumption date.

Any approval shall be in written communication and no verbal communication shall be considered as approval.

# Assumptions

1. 100% coverage can’t be achieved because of Macros
2. An anomaly shall be raised by the developer when 100% code coverage is not achievable because of the code issues
3. Requirement and design shall be baselined to performing the requirement based unit testing.
4. The developer shall verify the test report and take the necessary actions based on the test report submitted.
5. All the requirements can’t be unit tested at unit level and deviations will be recorded.

# Unit Testing Procedure

## Testing Method

Following are the Unit testing method which will be followed:

1. **Processor in Loop (PIL)**,

The processor in loop involves the code to be run on the target processor RH850 and unit test to be executed to evaluate the expected behavior. Unit testing of the module “**PSAStHdlr**” shall be executed on the EA4.0 architecture built on RH850 processor.

1. **Software in Loop (SIL)**

The software in loop testing of the module “**PSAStHdlr**” will be performed by ensuring the compiler and linker setting similar to that of development environment and using the default GCC compiler option present in Tessy.

## Unit Test Preparation

### Pre-Test Procedures

The following are the list of documents/files required to initiate the unit testing.

Functional requirement document [FR].

Module design document [MDD].

Functional Design[Simulink Model]

Data Dictionary.

Design traceability document.

Code traceability document.

Specification document

Dependent header files [Contract folder].

Nexteer UTT file.

EPS library file.

MIL Data [csv and .m]

### Preparation for Testing

The software and hardware setup document shall be followed to setup the hardware and software environment to begin with the unit testing.

Software Environment:

Test Specification document required to be updated.

The Nexteer UTT file shall be used to create the unit test vectors used for unit testing the component. The settings of Nexteer UTT is explained in the document:

‘UT\_Nexteer\_UTT\_Settings’

**Path:**

[**https://kchjsvni01.kpit.com/svn/R0006913/Nexteer/EA\_3.0/Testing/Unit\_Testing/05\_Tools\_Management/Software\_Tools/Nexteer UTT/ UT\_Nexteer\_UTT\_Settings.doc**](https://kchjsvni01.kpit.com/svn/R0006913/Nexteer/EA_3.0/Testing/Unit_Testing/05_Tools_Management/Software_Tools/Nexteer%20UTT/%20UT_Nexteer_UTT_Settings.doc)

NOTE: Documents common to EA3.0 & EA4.0 are stored in EA3.0 path.

The Nexteer UTT is used to form the UTP, where in the Test Vectors will be created, this phenomenon is well explained in the Unit test Guidelines document under the section 3.2.2

The Matlab module shall be used to create the expected output for the identified test vectors

The VBA module shall be used to create the expected output for the identified test vectors

The Provided EPS library shall be used to generate the expected output

If MIL approach is followed then MIL Simulink model will be used to create the expected output for the identified test vectors. No VBA will be created for this task nor the EPS library shall be used.

For NonSimulating modules the values generated by tessy will be evaluated against the relevant ranges mentioned in DD.

Following are the Sheets which will be created in UTP:

1. Range Sheet:

This sheet consist of the vectors which will be checking the Range of all the Input

variables.

The Vectors Description would involve the details of the input variables base on their

ranges, which would be taken from the Data Dictionary.

1. Path Sheet:

This sheet will consist of the vectors required to cover the paths or the condition involved

in the source code under test.

The vector description involves the details about the path or the logical conditions to be

covered by the vectors

The Path sheet will ensure to have the MC/DC, MCC, DC, C0 and C1 coverage as 100%.

If 100% coverage is not achieved by MIL data then extra test cases will be added to

achieve 100% coverage.

1. Metrics Sheet:

This sheet will consist of only two vectors, one will cover the longest possible path in the

function under test and other will cover the shortest path.

1. Interface Test:

This Sheet will consist of vectors required to test the Interface functions or the Stubs

which is used in the Functions under test.

As MIL Data available Interface test is covered in MIL testing, so sepreate test cases

will not be required.

1. Fault injection Test:

This sheet will consist of the vectors which are required to test the behavior of the

function if some fault is injected in it.

In the fault injection test we will give out of range values to the variable, out of bound

arrays and the initialization of the pointers etc.

As MIL Data available Fault injection test is covered in MIL testing, so sepreate test cases

will not be required.

1. Requirement coverage test:

This sheet will contain the test Cases to test each and every requirement mentioned in the

requirement specification document.

This sheet will contain the test vectors mapped to the test cases written to test the

requirements mentioned in the requirement document.

As MIL Data available Requirement coverage test is covered in MIL testing. so sepreate

test cases will not be required. If all requirements are not covered and test cases for those requirement required to be written then it will be added in this sheet.

1. MIL Data Test:

As MIL Data (csv & m file) & Simulink model is available then expected output will be calculated using Simulink model for extra created vector and Boundray vectors.

Include the Make File Templates, with correct Compiler and Linker settings, for the exact settings kindly refer the document - ‘EA\_4.0\_UT\_Compiler\_Settings\_Document.docx’

### Test Classification & Prioritization

Assignment of each module/sub-function with one of the following four classifications:

ASIL A

ASIL B

ASIL C

ASIL D

|  |  |  |
| --- | --- | --- |
| Module/Function | Classification | Priority for testing |
| Init | ASIL D | High |
| Period1 | ASIL D | High |

### Pass/Fail Criteria For Evaluating Results

Explain what the pass is and fail criteria, detail it for different type of test vectors for ex: in a Range test if the output or any internal variable is found to go outside the defined variable range (DD/MDD) then the vectors is considered fail.

The Common PASS/FAIL criteria depends on the values of the output variables, i.e. if the Actual values of the output vectors matches with that of the expected, then those vectors are considered as PASS, however if the Actual values of the output vectors does not matches with that of the expected values , then those vectors are considered as Fail.

Action:

1. Complete Analysis of the failure is to be done, Debug the failed vectors, and arrive to reason of failure, which could be due to wrong ranges mentioned in the DD/MDD, or it could be because of some functionality mismatch.
2. After the detailed analysis of the failure and its reason, a detailed E-Mail comprising of full functionality and the explanation is sent to the developer , by the Onsite Coordinator, who will seek the resolution of the Issue reported to him
3. If the developer agrees to the suggestions made, then these suggestions is added in the Unit Test report.

Apart From the common failure, following are the PASS/FAIL criteria’s with respect to the Kind of the Vectors:

#### Pass/Fail Crteria for vectors in Range Vectors:

Pass Criteria:

If in the output of the vector in Range test, any Module internal variable is remained inside the defined variable range (DD/MDD) then the vectors is considered Pass.

**Note:** As of now we do not test the Range of the Output variables used in the Rte\_write functionality.

Fail Criteria:

If in the output of the vector in Range test, any Module internal variable is found to go outside the defined variable range (DD/MDD) then the vectors is considered Fail.

Action:

1. Complete Analysis of the failure is to be done, Debug the vectors for which the output variables have got the out of range values and arrive to reason of failure, which could be due to wrong ranges mentioned in the DD/MDD, or it could be because of some functionality mismatch.
2. After the detailed analysis of the failure and its reason, a detailed E-Mail comprising of the full suggested ranges of the output variables and the explanation is sent to the developer , by the Onsite Coordinator, who will seek the resolution of the Issue reported to him

#### Pass/Fail Crteria for vectors in Path Test:

Pass Criteria:

If after executing all the vectors in the path sheet gives the MC/DC(Modified Condition/Decision Coverage), MCC(Multiple Condition coverage), DC(Decision Coverage), C0(Statement Coverage) and C1(Branch Coverage) coverage as 100%. Then the Path Test is considered as PASS.

Fail Criteria:

If after executing all the vectors in the path sheet gives the MC/DC, MCC, DC, C0 or C1 coverage less than 100%. Then the Path Test is considered as FAIL.

Action:

1. Complete Analysis of the failure i.e. the path which is uncovered is to be done, by giving the different possible values to the Input variables.
2. There could be scenario that, a certain path could be covered by giving out of range values for some of the variables.
3. In Case of above scenario, a detailed E-Mail comprising of full explanation of uncovered path and the suggested ranges of the input variables and the explanation is sent to the developer , by the Onsite Coordinator, who will seek the resolution of the Issue reported to him
4. If the developer agrees to the suggested ranges, then UT will be done with these ranges, and the DD/MDD is updated accordingly and the same is mentioned in the clarification sheet in Unit Test Results.
5. If after the complete analysis, the tester comes up with the conclusion that a particular path cannot be covered, a detailed Email is sent to the developer regarding the Uncovered path
6. If the developer agrees to the conclusion, the uncovered paths is mentioned in the first sheet of the Unit test Results and its clarification should be updated in the clarification sheet in the Unit test results.

#### Pass/Fail Crteria for vectors in Metrics Test:

The execution time of the function under test is mentioned in the header of the function under test, generally for the periodic functions the execution times is 2ms(Mili seconds).

However for some of the Non-Periodic functions , the execution time is not mentioned in the header, but these Non-Periodic functions would be called inside the Periodic functions in the source code file under test, so the execution time for these functions would be less than 2ms(Mili Seconds).

In TESSY Execution environment, the unit of the throughput data calculated is in CPU Cycles.

The conversion formula of the CPU Cycles to the Time in Milliseconds is given below:

msec = (cycles/clockrate)/1000

For example, on a 2.993GHz machine, a function with 36040 cycles would convert as (36040/2993)/1000, which is 12.04143 usec or .01204 milliseconds.

Pass Criteria:

After the calculation of the CPU cycles, and then converting it to ms, if the calculated time for both the Shortest path and the longest path is less than the mentioned execution rate in the function header(eg. 2ms) then the vectors is considered to be PASS.

Fail Criteria:

After the calculation of the CPU cycles, and then converting it to ms, if the calculated time for both the Shortest path and the longest path is greater than the mentioned execution rate in the function header (eg. 2ms) then the vectors is considered to be FAIL.

Action:

1. Calculate the throughput multiple times, if then also the time greater than the mentioned execution rate in the function header(eg. 2ms) the Complete Analysis of the failure is done.
2. The analysis of the failed test vectors also include the calculation of the Throughput data through multiple methods like the GPIO Pin toggling method or the RTI method.
3. If still In Case of the above scenario, a detailed E-Mail comprising of full explanation and the throughput data calculated from the different methods in Tabular form is sent to the developer , by the Onsite Coordinator, who will seek the resolution of the Issue reported to him
4. The Clarification received from the Developer is to be updated in the clarification sheet in the Unit test Results.

#### Pass/Fail Crteria for vectors in FaulT Injection Test:

NOTE: This is not applicable for Unit test with MIL approach.

The Pass criteria for injecting the faults would differ according to the faults injected explained below.

Pass Criteria:

1. Inputs exceeds legal range If a particular variable is given a value beyond its range, and the system output remains within its range as mentioned in DD, then those test cases would be considered as PASS.This applies to the logic variables also.
2. Inputs exceeds datatype range: If a particular variable is given a value beyond its data type range, and the system output remains within its range as mentioned in DD, then those test cases would be considered as PASS.

Fail Criteria:

1. Outputs exceeds legal range:

If a particular variable is given a value beyond its range, and the system output is not within its range as mentioned in the DD, then those test cases would be considered as FAIL. This applies to the logic variables also.

Action:

1. Analyze the fault injection vectors and a detailed E-Mail is sent to the Developer through the Onsite coordinator.
2. The Clarification received from the Developer is to be updated in the clarification sheet in the Unit test Results.

# Regression Strategy

Definition: Regression testing is always done to verify that modified code does not break the existing functionality of the application and works within the requirements of the system

If the current module is received with version update, we follow the following steps:

1. If the version of the source code under test is other than 1, then the UT results for the previous version is searched, and the previous workspace is restored.
2. Compare the current and the previous version of the source file received in the input folder of the task under test.
3. If there are minimal changes i.e. one out of all the functions is changed, then VBA only for that function is updated as per the changes. For the MIL approach, updated MIL test vectors will be used for regression.
4. If the changes made does not affect other functions in the source code, then the Vectors for that particular change is updated in the individual test sheets.
5. However TESSY does not support the this kind of partial regression testing,i.e if there is even a single line of source code change, the vectors of all the other functions along with the one which is changed to be executed.

**Note:** For point 5 kindly refer the embedded E-Mail below:



# Test Deliverables

List of all the unit test deliverables

## Technical

1. Unit test report consisting of [with production setting and with out production setting]
   1. Range condition check, Range check, underflow/overflow detection
   2. Module memory usage
   3. Module average throughput
   4. Module longest path throughput
   5. Coverage report
2. Coverage analysis if not 100% coverage
   1. Support documents
3. Updated DD/MDD documents
4. Test plan, Test case (Not applicable for MIL based approach) and test vectors.
5. Traceability document (Not applicable for MIL based approach).

## Process

1. Unit test task readiness checklist
2. Unit test delivery checklist
3. Unit test peer review checklist

# Communication Approach

The below are the list of communications done via mail to exchange the information between unit tester and component developer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SL Num | Initiated By | Sent to | Regarding | Remarks |
| 1 | Developer | UT team | Component submitted for unit testing | To initiate unit testing |
| 2 | UT team | Developer | Component unit test readiness status | To inform the readiness fo the component |
| 3 | UT team | Developer | Range issue |  |
| 4 | UT team | Developer | Missing header |  |
| 5 | UT team | Developer | MDD mismatch |  |
| 6 | UT team | Developer | Out of range issue |  |
| 7 | UT team | Developer | Coverage issue |  |
| 8 | UT team | Developer | Software/hardware issue |  |
| 9 | UT team | Developer | SER/SCIR dependecy |  |
| 10 | UT team | Developer | Unit test delivery |  |
| 11 | Developer | UT Team | Re execution | In case of any descripancies. |

# Appendix

<This section is for appendix>