|  |  |  |  |
| --- | --- | --- | --- |
| Title | Software Subsystem Test Plan & Report < MVMeasurement Subsystem> | | |
| **Distribution** | Software Archive | | |
| **Author** | Jaden-JinDing Liu | **Date** | 2021-06-03 |
| **Reviewer** | See Test plan and report Review | | |
| **Approver** |  | | |
| **Remarks** |  | | |

Contents

[1 Introduction 3](#_Toc73632025)

[1.1 Definitions, acronyms, and abbreviations 3](#_Toc73632026)

[2 Strategy 3](#_Toc73632027)

[2.1 Static Check 3](#_Toc73632028)

[2.2 Module test 5](#_Toc73632029)

[2.3 Funcational test 5](#_Toc73632030)

[2.4 Code Reviews 5](#_Toc73632031)

[3 Test environments/Tools 5](#_Toc73632032)

[3.1 PC\_Lint 5](#_Toc73632033)

[3.2 IAR workbench 6](#_Toc73632034)

[4 Test Cases 6](#_Toc73632035)

[4.1 Module Test 6](#_Toc73632036)

[4.1.1 Module Test cases 6](#_Toc73632037)

[4.1.2 Module Test Coverage 102](#_Toc73632038)

[4.1.3 Module Test Results 104](#_Toc73632039)

[4.2 Function Test 115](#_Toc73632040)

[4.2.1 Function Test mapping table 115](#_Toc73632041)

[4.2.2 Function Test Cases 116](#_Toc73632042)

[4.2.3 Functional Test Results: 149](#_Toc73632043)

[5 Test plan and report Review 151](#_Toc73632044)

[5.1 Review-Participant: 152](#_Toc73632045)

[5.2 Decision of the Review: 152](#_Toc73632046)

[6 References 153](#_Toc73632047)

[7 Revision History 153](#_Toc73632048)

**List of Figures**

[Figure 1: Test Progress 3](#_Toc73632049)

**List of Tables**

[Table 1: Mapping between Test Case and Requirements from SSRS 115](#_Toc73632050)

# Introduction

This document defines the planned strategy for testing the MVM subsystem software.

The MVM subsystem is principally responsible for calculating standard volume flow and mass flow, Energy flow for Liquid, Gas, and Steam... (See MVM Subsystem Requirements Specification [1]).and

The objectives of the testing are

* Give confidence that the MVM subsystem requirements [1]have been met.
* There are no serious defects in its implementation.
* Document non-serious defects that are not going to be corrected.

## Definitions, acronyms, and abbreviations

| **Term** | **Definition** |
| --- | --- |
| Black Box Test | Test that requires no internal knowledge of the unit under test. |
| Common Framework | ABB BUI Common Framework [2] for subsystems and data object handling. |
| DUT | Device under test. Refers to the scope of the tests so could be a complete system, a subsystem or a class. |
| SCC | Source code control (E.g Subversion) |
| White Box Test | Test involving some internal knowledge of the unit under test. |

# Strategy



Figure 1: Test Progress

## Static Check

Lint [3] at level 3 will be used for static code analysis on all source .c modules with the following actions to be taken before code is checked-in for review:

1. Corrections will be applied to code constructs associated with error messages. Re-Lint.
2. All warnings and information messages will be assessed and either the associated construct will be changed or the message suppressed. Where a message is suppressed a reason will be provided in the source comment with the suppression. Re-Lint.

Commit all subsystem source code to latest version before code checks.

PC-lint result:

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\gasQm.c

C:\VT5\HART\cb\_application\MVMeasurement\Source\gasQm.c(124): Info 715: Symbol 'inNVF' (line 62) not referenced

C:\VT5\HART\cb\_application\MVMeasurement\Source\gasQm.c(62): Info 830: Location cited in prior message

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\gasQn.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\GasQnPartial.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\gasQp.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\GasQvPartial.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\IAPWS\_IF97.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\LiquidQm.c

C:\VT5\HART\cb\_application\MVMeasurement\Source\LiquidQm.c(121): Info 715: Symbol 'inTReal' (line 59) not referenced

C:\VT5\HART\cb\_application\MVMeasurement\Source\LiquidQm.c(59): Info 830: Location cited in prior message

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\LiquidQn.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\LiquidQp.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\MVMeasurement\_constructor.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\MVMeasurement\_execute.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\MVMeasurement\_overload.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\QCommonCal.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\splint\_SplineCubicInterpolation.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\SteamQm.c

--- Module: C:\VT5\HART\cb\_application\MVMeasurement\Source\SteamQp.c

--- Global Wrap-up

(): Note 900: Successful completion, 4 messages produced

PC-Lint Check passes

## Module test

* Test should be done automatically, not execute test code step by step thru breakpoint.
* Test focus on software logics, not include hardware related driver. – Low lever driver will be fully tested during hardware debug phase.
* All function branches will be covered in module test.
* Build test case first, especially build abnormal case first – boundary test.

## Funcational test

The subsystem test will aim to test various scenarios of the subsystem use cases as identified by the subsystem requirements [1][2] such that if the subsystem passes these exercises then it can be assumed to meet the subsystem requirements.

These tests will be principally Black Box test cases that will be defined for checking the public interfaces to the MVMeasurement subsystem. The test cases (with expected result where appropriate) and the actual test results are listed below.

The test cases are designed to be easily repeated and will cover all requirements. Boundary conditions will be tested where appropriate. See part 4.

## Code Reviews

Peer reviews of all C and assembly source modules (ie files with extensions .c, .h and .s34) will be conducted to check that coding standards and guidelines specified by the Software Code Review [4] have been followed.

The reviews will also aim to identify defects in the source code for the MVMeasurement subsystem that cannot be spotted through automatic checking tools such as not following the coding guidelines, having incorrect comments, implementation that does not follow the design and unnecessarily obscure or complex code.

The code reviews will also verify that the module test cases, located at the end of the module, adequately exercise the methods within it.

Code reviews will be documented and stored on the Subversion site with MVMeasurement subsystem code review [4].

# Test environments/Tools

Prior to reviews and tests, all CDT subsystem source code (and any modules that it depends on that were not part of the Common Framework) shall have been committed to SVN server.

One new branch will be created to perform the test and indicate the version of CDT subsystem being checked and this should be recorded with the test results.

## PC\_Lint

PC-Lint for C/C++ version 8.0.0

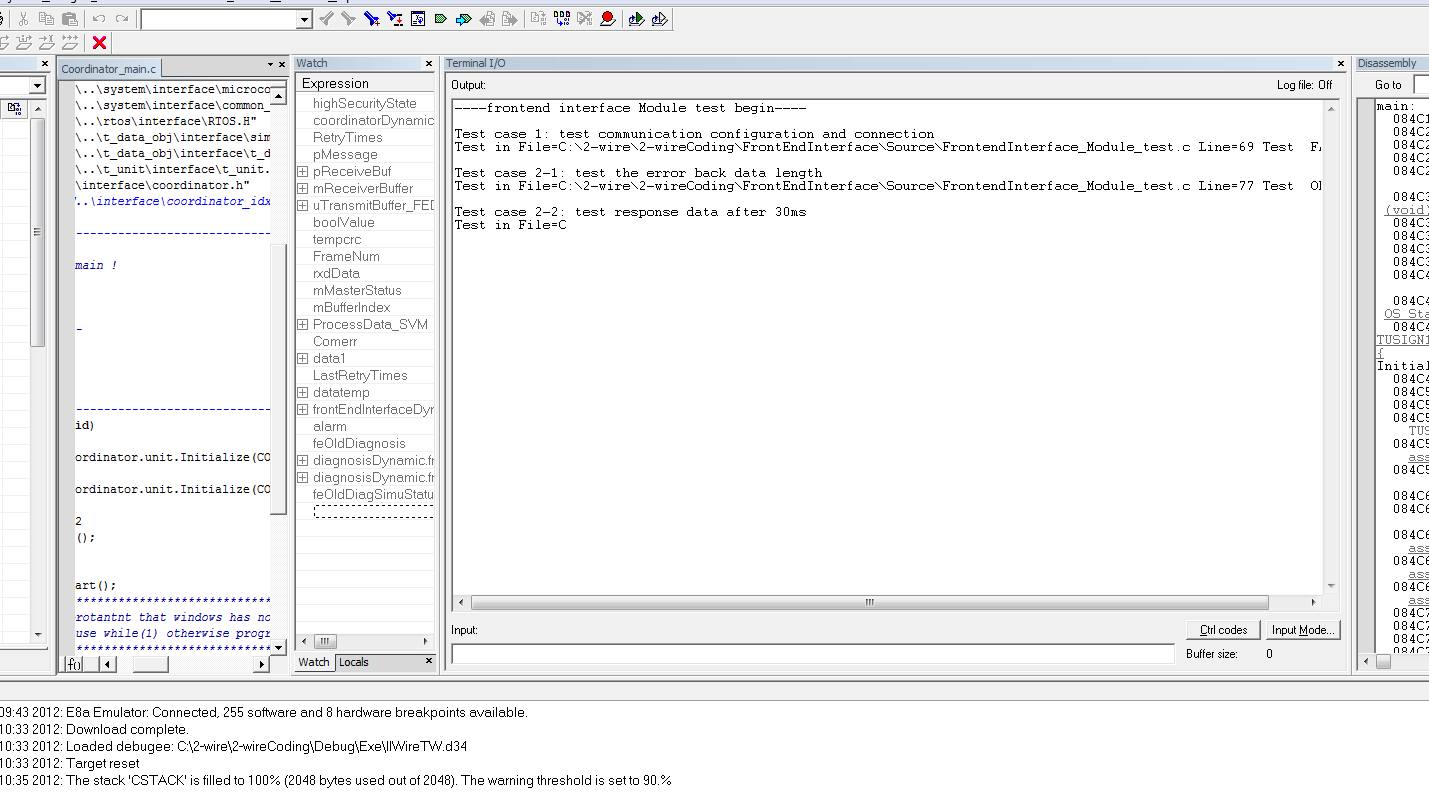
PC-lint for C/C++ (NT) Ver. 8.00o, Copyright Gimpel Software 1985-2004

## IAR workbench

IAR Embedded Workbench IDE 3.40

Integrated Development Environment: Compiler (v3.40.5) and Linker (v4.61R) for M16C.

Module tests and functional tests will be performed under emulation with external dependencies being handled by accessing data objects from external subsystems.



# Test Cases

## Module Test

### Module Test cases

#### Module LiquidQn.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateLiquidQn(TFLOAT inVF,TFLOAT inTReal) | |
| Test function name | void LiquidQnTest(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following:   range.hiLim = 90.00f;  range.loLim = 2.10f;  range.span = 87.90f;  QnSimEnable = SVM\_ENABLE  QnSim = 82.0f  VF = 87.58f  T Ref = 21.35f;  T Real = 31.25f;   1. Call CalculateLiquidQn(VF,tReal); |
| Expected result | 1. Range from 81.9999 to 82.0001 as the sim value 2. Cut off shall not be set. |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following:   VF = 87.58f;  QnSim = 82.00f;  tReal = 31.25f;  tRef = 21.35f;  NF = 0.00f;  vExpandBeta1 = 0.036;  range.hiLim = 90.00f;  range.loLim = 2.10f;  range.span = 87.90f;   1. Call CalculateLiquidQn(VF,tReal); |
| Expected result | 1. Range from Range from 64.5678f to 64.5680f 2. Cut off alarm shall not be set |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following:   VF = 87.58f;  Disable simulation;  QnSim = 82.00f;  tReal = 31.25f;  tRef = 21.35f;  NF = 0.00f;  vExpandBeta1 = 0.036;  range.hiLim = 90.00f;  range.loLim = 2.10f;  range.span = 87.90f;  Call CalculateLiquidQn(VF,tReal); |
| Expected result | 1. Shall return 0 2. Cut off alarm shall be set |
| Test result | Pass |

#### Module LiquidQm.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateLiquidQm(TFLOAT inVF,TFLOAT inDensityReal,TFLOAT inTReal) | |
| Test function name | void LiquidQmTest() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following:   SimulationMode = CDT\_SIMULATION\_CB\_QM\_VALUE  QmSimEnable = MVM\_DISABLE  LiquidCorrection = MVM\_WITHOUT\_CORRECTION  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;  QmSim = 78800kg/h  QmPerSim=70.00%  Tref =21.35 ¡æ  Treal =31.25 ¡æ  DensityRef =890.88 kg/m2  DensityReal =998.89 kg/m3  VolumeExpandBeta1= 0.036  densityExpandBeta2= 0.016  VolumeFlow= 77.580 m3/h   1. Call CalculateLiquidQm(VF,densityReal,tReal); |
| Expected result | 78800.0000f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following:   VF = 77.580f;  volumeExpandBeta1 = 0.036;  densityExpandBeta2 = 0.018;  densityRef =890.88 ;  densityReal =998.89 ;  tReal = 31.25f;  tRef = 21.35f;  liquidCorrection = MVM\_WITHOUT\_CORRECTION;  QmPerSim = 70.00f;  QmSim = 78800.00000f;  QmSimEnable = MVM\_DISABLE;  simulationMode = CDT\_SIMULATION\_CB\_QM\_PERCENTAGE;  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;   1. Call CalculateLiquidQm(VF,densityReal,tReal) |
| Expected result | 55860.0000f |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following:   VF = 77.580f;  volumeExpandBeta1 = 0.036;  densityExpandBeta2 = 0.018;  densityRef =890.88 ;  densityReal =998.89 ;  tReal = 31.25f;  tRef = 21.35f;  liquidCorrection = MVM\_WITHOUT\_CORRECTION;  QmPerSim = 70.00f;  QmSim = 78800.00000f;  QmSimEnable = MVM\_ENABLE;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  DValue = 78800.0000f;    T\_DEV\_RNGFLT range;  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;     1. CalculateLiquidQm(VF,densityReal,tReal) |
| Expected result | 78800.000000 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following:   VF = 77.580f;  volumeExpandBeta1 = 0.036;  densityExpandBeta2 = 0.018;  densityRef =890.88 ;  densityReal =998.89 ;  tReal = 31.25f;  tRef = 21.35f;  liquidCorrection = MVM\_WITHOUT\_CORRECTION;  QmPerSim = 70.00f;  QmSim = 78800.00000f;  QmSimEnable = MVM\_DISABLE;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;   1. CalculateLiquidQm(VF,densityReal,tReal) |
| Expected result | 77493.88620 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following:   VF = 77.580f;  volumeExpandBeta1 = 0.036;  densityExpandBeta2 = 0.018;  densityRef =890.88 ;  densityReal =998.89 ;  tReal = 31.25f;  tRef = 21.35f;  liquidCorrection = MVM\_DENSITY\_CORRECTION;  QmPerSim = 70.00f;  QmSim = 78800.00000f;  QmSimEnable = MVM\_DISABLE;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;   1. CalculateLiquidQm(VF,densityReal,tReal) |
| Expected result | 81430.66903f |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following:   VF = 77.580f;  volumeExpandBeta1 = 0.036;  densityExpandBeta2 = 0.018;  densityRef =890.88 ;  densityReal =998.89 ;  tReal = 31.25f;  tRef = 21.35f;  liquidCorrection = MVM\_VOLUME\_CORRECTION;  QmPerSim = 70.00f;  QmSim = 78800.00000f;  QmSimEnable = MVM\_DISABLE;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;   1. CalculateLiquidQm(VF,densityReal,tReal) |
| Expected result | 50954.34267f |
| Test result | Pass |

#### Module LiquidQp.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateLiquidQp(TFLOAT inMF,TFLOAT inTReal,TFLOAT inTExtReal) | |
| Test function name | void LiquidQpTest(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following:   MF = 8240.580f;  tReal = 121.25f;  tExtReal = 112.25f;  QpSim = 298493.99000;  QpSimEnable = MVM\_ENABLE;  range.hiLim = 320000.00f;  range.loLim = 100.0f;  range.span = 319900.0f;   1. Call CalculateLiquidQp(MF,tReal,tExtReal) |
| Expected result | 298493.99000f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following:   MF = 8240.580f;  tReal = 121.35f;  tExtReal = 112.25f;  heatCapacity = 4.20f;  QpSim = 298493.99000;  QpSimEnable = MVM\_DISABLE;  range.hiLim = 320000.00f;  range.loLim = 100.0f;  range.span = 319900.0f;   1. Call CalculateLiquidQp(MF,tReal,tExtReal) |
| Expected result | 314954.96760f |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following:   MF = 7500.0f;  tReal = 121.35f;  tExtReal = 112.25f;  heatCapacity = 4.20f;  QpSim = 298493.99000;  QpSimEnable = MVM\_DISABLE;  range.hiLim = 320000.00f;  range.loLim = 100.0f;  range.span = 319900.0f;   1. Call CalculateLiquidQp(MF,tReal,tExtReal) |
| Expected result | 0.0 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following:   MF = 7500.0f;  tReal = 121.35f;  tExtReal = 112.25f;  heatCapacity = 4.20f;  QpSim = 298493.99000;  QpSimEnable = MVM\_DISABLE;  range.hiLim = 320000.00f;  range.loLim = 100.0f;  range.span = 319900.0f;   1. Call CalculateLiquidQp(MF,tReal,tExtReal) |
| Expected result | Code coverage: Missing else statement execution |
| Test result | Pass |

#### Module GasQm.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQm(TFLOAT inVF,TFLOAT inNVF,TFLOAT densityReal) | |
| Test function name | void GasQmTest(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following:   VF = 89.580f;  NF = 1128.580f;  QmSim = 28862.00f;  QmPerSim = 75.00f;  dRef = 1.293f;  dReal = 13.987f;  gasDensitySelection = MVM\_REFERENCY\_DENSITY;  range.hiLim = 30000.00f;  range.loLim = 100.0f;  range.span = 29900.0f;   1. Call CalculateGasQm(VF,NF,dReal) |
| Expected result | 28862.000000f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following:   VF = 89.580f;  NF = 1128.580f;  QmSim = 27872.00f;  QmPerSim = 72.00f;  dRef = 1.293f;  dReal = 13.987f;  gasDensitySelection = MVM\_REFERENCY\_DENSITY;  range.hiLim = 29000.00f;  range.loLim = 100.0f;  range.span = 28900.0f;  simulationMode = CDT\_SIMULATION\_CB\_QM\_PERCENTAGE;  QmSimEnable = MVM\_DISABLE;     1. Call CalculateGasQm(VF,NF,dReal) |
| Expected result | 20880.000000f |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following:   VF = 89.580f;  NF = 1128.580f;    QmSim = 27872.00f;  QmPerSim = 72.00f;  dRef = 1.293f;  dReal = 13.987f;  gasDensitySelection = MVM\_REFERENCY\_DENSITY;  range.hiLim = 29000.00f;  range.loLim = 100.0f;  range.span = 28900.0f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  QmSimEnable = MVM\_ENABLE;  RV = CalculateGasQm(VF,NF,dReal);     1. Call CalculateGasQm(VF,NF,dReal) |
| Expected result | 27872.00000f |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following:   VF = 89.580f;  NF = 1128.580f;    QmSim = 27872.00f;  QmPerSim = 72.00f;  dRef = 1.293f;  dReal = 13.987f;  gasDensitySelection = MVM\_REFERENCY\_DENSITY;  range.hiLim = 29000.00f;  range.loLim = 100.0f;  range.span = 28900.0f;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  QmSimEnable = MVM\_DISABLE;   1. Call CalculateGasQm(VF,NF,dReal) |
| Expected result | 1459.25394f |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following:   VF = 89.580f;  NF = 1128.580f;  QmSim = 27872.00f;  QmPerSim = 72.00f;  dRef = 1.293f;  dReal = 13.987f;  gasDensitySelection = MVM\_ACTUAL\_DENSITY;  range.hiLim = 29000.00f;  range.loLim = 100.0f;  range.span = 28900.0f;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  QmSimEnable = MVM\_DISABLE;   1. Call CalculateGasQm(VF,NF,dReal) |
| Expected result | 1. 1252.955460f 2. Cut off shall be set |
| Test result | Pass |
| Test case 6 | Test procedure | To test the else branch of if (IsPVMatchWithQ(CDT\_GAS\_MASS))   1. Set the Precondition parameter as following:   VF = 89.580f;  NF = 1128.580f;  QmSim = 27872.00f;  QmPerSim = 72.00f;  dRef = 1.293f;  dReal = 13.987f;  gasDensitySelection = MVM\_ACTUAL\_DENSITY;  range.hiLim = 29000.00f;  range.loLim = 100.0f;  range.span = 28900.0f;  simulationMode = CDT\_SIMULATION\_CB\_OFF;  QmSimEnable = MVM\_DISABLE;  PV = MAPPER\_DEVICE\_VAR\_POWER;  Call CalculateGasQm(VF,NF,dReal) |
| Expected result | Cut off alarm shall remain unchanged. |
| Test result | Pass |

#### Module gasQn.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQn(TFLOAT inVF, TFLOAT tReal, TFLOAT pReal) | |
| Test function name | void LineariztionLiquidKF Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following:   VF = 1127.580f;  QnSim = 28862.00f;  gasRef = MVM\_UK;  tReal = 31.25f;  pReal = 235.23f;  range.hiLim = 30000.00f;  range.loLim = 10.0f;  range.span = 29990.0f;     1. Call CalculateGasQn(VF,tReal,pReal) |
| Expected result | 28862.0000 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following:   VolumeFlow = 1127.580 m3/h  gasComputeType = MVM\_IDEAL\_GAS  QnSimEnable MVM\_DISABLE  range.hiLim = 30000.00f;  range.loLim = 10.0f;  range.span = 29990.0f;  QnSim =28862 m4/h  T Real =31.35 ¡æ  P real =2535.23 Kpa  gasRef = MVM\_UK   1. Call CalculateGasQn(VF,tReal,pReal) |
| Expected result | 26750.925 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following:   QnSimEnable MVM\_DISABLE  range.hiLim = 30000.00f;  range.loLim = 10.0f;  range.span = 29990.0f;  QnSim =28862 m4/h  T Real =31.35  P real =2535.23  gasRef = MVM\_US  Z Standard =0.8956  Z Real =0.8896  VolumeFlow =1127.580 m3/h  gasComputeType = MVM\_ISO12213\_MOLAR  cut off = 0   1. Call CalculateGasQn(VF,tReal,pReal) |
| Expected result | 27449.593 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following:   QnSimEnable MVM\_DISABLE  range.hiLim = 30000.00f;  range.loLim = 10.0f;  range.span = 29990.0f;  QnSim =28862 m4/h  T Real =31.35  P real =2535.23  gasRef = MVM\_US  Z Standard =0.8956  Z Real =0.8896  VolumeFlow =1127.580 m3/h  gasComputeType = MVM\_ISO12213\_MOLAR  cut off = 20   1. Call CalculateGasQn(VF,tReal,pReal) |
| Expected result | 1. Value shall be 0 2. Cut off shall be set |
| Test result | Pass |

#### Module gasQp.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQp(TFLOAT inVF) | |
| Test function name | void GasQpTest(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QpSimEnable = MVM\_ENABLE;  QpSim = 2040000.00f;  calorificEnergy = 30.0f;  range.hiLim = 2370000.00f;  range.loLim = 1000.0f;  range.span = 2369000.0f;   1. Call CalculateGasQp(VF); |
| Expected result | 2040000.000f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QpSimEnable = MVM\_DISABLE;  QpSim = 2040000.00f;  calorificEnergy = 30.0f;  range.hiLim = 2370000.00f;  range.loLim = 1000.0f;  range.span = 2369000.0f;   1. Call CalculateGasQp(VF); |
| Expected result | 1727400 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QpSimEnable = MVM\_DISABLE;  QpSim = 2040000.00f;  calorificEnergy = 30.0f;  range.hiLim = 2370000.00f;  range.loLim = 1000.0f;  range.span = 2369000.0f;   1. Call CalculateGasQp(VF); |
| Expected result | 1727400 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QpSimEnable = MVM\_DISABLE;  calorificEnergy = 30.0f;  range.hiLim = 2370000.00f;  range.loLim = 1000.0f;  range.span = 2369000.0f;   1. Call CalculateGasQp(VF); |
| Expected result | Code coverage: Missing else statement execution |
| Test result | Pass |

#### Module GasQvPartial.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQvPartial(TFLOAT inVF,TFLOAT biogasPerc) | |
| Test function name |  | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QvPartialSim = 68.00f;  biaGas = 60.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  QvPartialSimEnable = MVM\_ENABLE;   1. Call CalculateGasQvPartial(VF,biaGas); |
| Expected result | 68.00000f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QvPartialSim = 68.00f;  biaGas = 60.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  QvPartialSimEnable = MVM\_DISABLE;   1. Call CalculateGasQvPartial(VF,biaGas); |
| Expected result | 34.54800 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   VF = 57.580f;  QvPartialSim = 68.00f;  biaGas = 60.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  QvPartialSimEnable = MVM\_DISABLE;  Cut off = 90   1. Call CalculateGasQvPartial(VF,biaGas); |
|  | Expected result | 1. The value shall be 0 2. Cut off shall be set |
|  | Test result | Pass |

#### Module GasQnPartial.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQvPartial(TFLOAT inVF,TFLOAT biogasPerc) | |
| Test function name |  | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   VF = 67.870;  QnPartialSim = 65.00f;  QnPartialPerSim = 78.00f  biaGas = 56.00f;  range.hiLim = 72.00f;  range.loLim = 2.0f;  range.span = 70.0f;  QnPartialSimEnable = MVM\_ENABLE;   1. Call CalculateGasQnPartial(VF,biaGas); |
| Expected result | 65.00000f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   VF = 67.870;  QnPartialSim = 65.00f;  QnPartialPerSim = 78.00f  biaGas = 56.00f;  range.hiLim = 72.00f;  range.loLim = 2.0f;  range.span = 70.0f;  QnPartialSimEnable = MVM\_ENABLE;   1. Call CalculateGasQvPartial(VF,biaGas); |
| Expected result | 38.007200 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   VF = 67.870;  QnPartialSim = 65.00f;  QnPartialPerSim = 78.00f  biaGas = 56.00f;  range.hiLim = 72.00f;  range.loLim = 2.0f;  range.span = 70.0f;  QnPartialSimEnable = MVM\_ENABLE;  Cut off = 90   1. Call CalculateGasQvPartial(VF,biaGas); |
|  | Expected result | 1. The value shall be 0 2. Cut off shall be set |
|  | Test result | Pass |

#### Module IAPWS\_IF97.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT vpt1n\_(const TFLOAT \*p,const TFLOAT \*t) | |
| Test function name | void Vpt1n\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 300.0;  p = 3.0 ;     1. Call vpt1n\_( &p,&t); |
| Expected result | 0.00100215168 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT hpt1n\_(const TFLOAT \*p,const TFLOAT \*t) | |
| Test function name | void Hpt1n\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 300.0;  p = 3.0 ;     1. Call hpt1n\_( &p,&t); |
| Expected result | 115.331273 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT vpt2n\_(const TFLOAT \*p,const TFLOAT \*t) | |
| Test function name | void Vpt2n\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 493.15;  p = 1.75;     1. Call vpt2n\_( &p,&t); |
| Expected result | 0.1187 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT hpt2n\_(const TFLOAT \*p,const TFLOAT \*t) | |
| Test function name | void hpt2n\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 493.15;  p = 1.75;     1. Call hpt2n\_( &p,&t); |
| Expected result | 2836.40 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT vpt3n\_(const TFLOAT \*p,const TFLOAT \*t) | |
| Test function name | void Vpt3n\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 633.15;  p = 21.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0.001789 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 21.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0.0072 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   t = 643.15;  p = 23.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0.001945 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 24.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 002612 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 23.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 004782 |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 28.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 001952 |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 42.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 001654 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   t = 663. 150;  p = 25. 00;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 004647 |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   t = 663. 1500001;  p = 25. 0000001;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 004647 |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set the Precondition parameter as following   t = 637.15;  p = 16.8 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 0103586 |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set the Precondition parameter as following   t = 723.15;  p = 43.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 003103 |
| Test result | Pass |
| Test case 12 | Test procedure | 1. Set the Precondition parameter as following   t = 823.15;  p = 72.0 ;     1. Call vpt3n\_( &p,&t); |
| Expected result | 0. 0031134 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT Hpt3n\_(const TFLOAT \*p,const TFLOAT \*t) | |
| Test function name | void Hpt3n\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 633.15;  p = 21.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 1728.76 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 21.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2591.72 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   t = 643.15;  p = 23.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 1818.84 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 24.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2025.16 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 23.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2364.90 |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 28.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 1862.42 |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   t = 653.15;  p = 42.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 1767.55 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   t = 663. 150;  p = 25. 00;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2395.53 |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   t = 663. 1500001;  p = 25. 0000001;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2395.53 |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set the Precondition parameter as following   t = 637.15;  p = 16.8 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2702.08376 |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set the Precondition parameter as following   t = 723.15;  p = 43.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2413.2913 |
| Test result | Pass |
| Test case 12 | Test procedure | 1. Set the Precondition parameter as following   t = 823.15;  p = 72.0 ;     1. Call v= vpt3n\_( &p,&t); 2. Call hvt3n\_( &v,&t); |
| Expected result | 2776.244 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | int wnpt3\_(const TFLOAT \*xa,const TFLOAT \*xb,TFLOAT (\*f) (),const TFLOAT \* p,const TFLOAT \*t,const TFLOAT \*eps,const TFLOAT \*px,const TINT16 \*pix) | |
| Test function name | void wnpt3\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   xa = 3.0;  xb = 3.0;  p = 0.03;  t = 100.0;  eps = 3.0;  x = 5.0;   1. Call wnpt3\_(&xa, &xb,&fwnpt3\_1,&p,&t, &eps, &x, &ix) |
| Expected result | ix == 1.0 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   xa = -3.09;  xb = 3.0;  p = 0.003;  t = 300.0;  eps = 3.0;  x = 5.0;   1. Call wnpt3\_(&xa, &xb,&fwnpt3\_1,&p,&t, &eps, &x, &ix) |
| Expected result | ix == 0.0 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   xa = -1.09;  xb = -5.00;  p = 0.013;  t = 280.0;  eps = 8.0;  x = 5.0;   1. Call wnpt3\_(&xa, &xb,&fwnpt3\_1,&p,&t, &eps, &x, &ix) |
| Expected result | ix == 1.0 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   xa = -9.0;  xb = 13.0;  p = 13.0;  t = 26.0;  eps = 0.31;  x = 5.0;   1. Call wnpt3\_(&xa, &xb,&fwnpt3\_1,&p,&t, &eps, &x, &ix) |
| Expected result | ix == 1.0 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT diter3\_( const TFLOAT \*p, const TFLOAT \*t, const TFLOAT \*pDest, const TFLOAT \*eps) | |
| Test function name | void diter3\_Tetst(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   rValue = 0.0f;  dest = 0.0f;  p = 0.03;  t = 100.0;  eps = 3.0;     1. Call diter3\_(&p, &t, &dest,&eps) |
| Expected result | 0.0 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | int regsopt\_(const TFLOAT \*p, const TFLOAT \*t, const TINT16 \*pIreg) | |
| Test function name | void Regsopt\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t= 300.0;  p= 3.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X01 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   t= 493.15;  p= 2.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X02 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   t= 993.15;  p= 2.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X02 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   t= 633.15;  p= 17.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X02 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   t= 633.1500001;  p= 17.0000001;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X02 |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following   t= 633.15;  p= 21.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X03 |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   t= 1093.15;  p= 8.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X05 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   t= 1093.15;  p= 12.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X00 |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   t= 193.15;  p= 2.0;   1. Call regsopt\_(&p,&t,&ireg); |
| Expected result | ireg == 0X00 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT pbt\_(const TFLOAT \*t) | |
| Test function name | void Pbt\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   t = 613.15;   1. Call pbt\_(&t); |
| Expected result | 14.6 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   t = 700.00;   1. Call pbt\_(&t); |
| Expected result | 0.0 |
| Test result | Pass |

#### Module splint\_SplineCubicInterpolation.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | void spline1d(TFLOAT \* x, TFLOAT\* y, TUSIGN8 n, TFLOAT\* y2) | |
| Test function name | void spline1d\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   x[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };  y[4] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463  };  y2[4] = {  1.0 ,  2.0,  3.0,  4.0  };   1. Call spline1d( x, y, 0x60,y2); |
| Expected result | y2[0] == 1.0  y2[2] == 3.0 |
| Test result | Pass |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   x[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };  y[4] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463  };  y2[4] = {  1.0 ,  2.0,  3.0,  4.0  };   1. Call spline1d( x, y, 4,y2); |
| Expected result | y2[0] == 0.0  y2[2] == 000026694 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void spline2d(TFLOAT \* x1a, TFLOAT \* x2a, TFLOAT \*ya, TUSIGN8 m, TUSIGN8 n, TFLOAT \*y2a) | |
| Test function name | void spline2d\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };    TFLOAT tArray[3] = {  6.85,  16.85,  26.85  };    TFLOAT cFArray[12] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.80573 ,    0.87712 ,  0.86025 ,  0.84512 ,  0.83201  };    TFLOAT y2[12] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.80573 ,    0.87712 ,  0.86025 ,  0.84512 ,  0.83201  };       1. Call spline2d(pArray,tArray,cFArray,4,3,y2); |
| Expected result | y2[0] == 0.0  y2[4] ==-0.00155715 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void splineInterpolate1d(TFLOAT\* xa, TFLOAT\* ya, TFLOAT\* y2a, TUSIGN8 n, TFLOAT x, TFLOAT \*y) | |
| Test function name | void splineInterpolate1d\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT x[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };  TFLOAT y[4] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463  };  TFLOAT y2[4] = {  0.0,  0.0,  0.0,  0.0  };    TFLOAT xValue = 84.0;  TFLOAT yValue = 0.0;     1. Call spline1d( x, y, 0x04,y2); 2. Call splineInterpolate1d(x,y,y2,0x04,xValue,&yValue); |
| Expected result | 0.81346 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   x[0]= 0.0f;  x[1]= 0.0f;  x[2]= 0.0f;  x[3]= 0.0f;  y[0] = 0.0;  y[1] = 0.0;  y[2] = 0.0;  y[3] = 0.0;  TFLOAT y2[4] = {  0.0,  0.0,  0.0,  0.0  };    yValue = 0.71346;   1. Call spline1d( x, y, 0x04,y2); 2. Call splineInterpolate1d(x,y,y2,0x04,xValue,&yValue); |
| Expected result | 71346 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void splineInterpolate2d(TFLOAT\* x1a, TFLOAT\* x2a, TFLOAT \*ya, TFLOAT \*y2a, TUSIGN8 m, TUSIGN8 n, TFLOAT x1, TFLOAT x2, TFLOAT \*y) | |
| Test function name | void splineInterpolate2d\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };    TFLOAT tArray[3] = {  6.85,  16.85,  26.85  };    TFLOAT cFArray[12] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.80573 ,    0.87712 ,  0.86025 ,  0.84512 ,  0.83201  };    TFLOAT y2[12] = {  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,    0.0 ,  0.0 ,  0.0 ,  0.0  };    TFLOAT tIn = 16.85;  TFLOAT pIn = 96.0;  TFLOAT cF = 58.0;   1. Call spline2d(pArray,tArray,cFArray,4,3,y2); 2. Call splineInterpolate2d((TFLOAT \*) tArray,(TFLOAT \*)pArray, (TFLOAT \*)cFArray,y2, 3, 0x60,tIn ,pIn, &cF); |
| Expected result | 58.0 |
| Test result | Pass |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };    TFLOAT tArray[3] = {  6.85,  16.85,  26.85  };    TFLOAT cFArray[12] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.80573 ,    0.87712 ,  0.86025 ,  0.84512 ,  0.83201  };    TFLOAT y2[12] = {  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,  0.0 ,    0.0 ,  0.0 ,  0.0 ,  0.0  };    TFLOAT tIn = 16.85;  TFLOAT pIn = 96.0;  TFLOAT cF = 58.0;   1. Call spline2d(pArray,tArray,cFArray,4,3,y2); 2. Call splineInterpolate2d((TFLOAT \*) tArray,(TFLOAT \*)pArray, (TFLOAT \*)cFArray,y2, 3, 0x60,tIn ,pIn, &cF); |
| Expected result | 0.82107 |
| Test result | Pass |

#### Module SteamQm.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateSteamQm(TFLOAT inVF,TFLOAT inDensityReal) | |
| Test function name | void SteamQmTest() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 0.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_QM;  TUSIGN8 lOpMode = CDT\_STEAM\_MASS;  T\_DEV\_RNGFLT range;  TFLOAT VF = 57.580f;  TFLOAT inputDensity = 30.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QmSim = 68.00f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_QM\_VALUE;   1. Call CalculateSteamQm(VF,densityReal); |
| Expected result | Qm shall be equal to QmSim  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be cleared |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 0.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_QM;  TUSIGN8 lOpMode = CDT\_STEAM\_MASS;  T\_DEV\_RNGFLT range;  TFLOAT VF = 57.580f;  TFLOAT inputDensity = 30.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QmSim = 68.00f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_QM\_PERCENTAGE;  mVMeasurementUnprotected.QmPerSim = 50.0f;   1. Call CalculateSteamQm(VF,densityReal); |
| Expected result | Qm shall be equal to QmPerSim\*range.hiLim / 100.0f  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be cleared |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 0.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_QM;  TUSIGN8 lOpMode = CDT\_STEAM\_MASS;  T\_DEV\_RNGFLT range;  TFLOAT VF = 57.580f;  TFLOAT inputDensity = 30.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QmSimEnable = MVM\_ENABLE;  mVMeasurementUnprotected.QmSim = 68.00f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  mVMeasurementUnprotected.QmPerSim = 50.0f;     1. Call CalculateSteamQm(VF,densityReal); |
| Expected result | Qm shall be equal to QmSim  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be cleared |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 0.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_QM;  TUSIGN8 lOpMode = CDT\_STEAM\_MASS;  T\_DEV\_RNGFLT range;  TFLOAT VF = 57.580f;  TFLOAT inputDensity = 1.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QmSim = 68.00f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  mVMeasurementUnprotected.QmPerSim = 50.0f;   1. Call CalculateSteamQm(VF,densityReal); |
| Expected result | Qm shall be equal to 57.58f  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be cleared |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 90.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_QM;  TUSIGN8 lOpMode = CDT\_STEAM\_MASS;  T\_DEV\_RNGFLT range;  TFLOAT VF = 57.580f;  TFLOAT inputDensity = 1.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QmSim = 68.00f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  mVMeasurementUnprotected.QmPerSim = 50.0f;  MVM\_CLRDIAG\_(MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF, mVMeasurementDynamicDuplicated.mvmDiagnosis);   1. Call CalculateSteamQm(VF,densityReal); |
|  | Expected result | Qm shall be equal to 0  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be set |
|  | Test result | Pass |

#### Module SteamQp.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateSteamQp(TFLOAT inMF) | |
| Test function name | void SteamQpTest(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 0.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_POWER;  TUSIGN8 lOpMode = CDT\_STEAM\_POWER;  T\_DEV\_RNGFLT range;  TFLOAT VF = 57.580f;  TFLOAT Hw = 60.0f, Hc = 20.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QpSimEnable = MVM\_ENABLE;  mVMeasurementUnprotected.QpSim = 68.00f;  mVMeasurementDynamicDuplicated.Hw = Hw;  mVMeasurementDynamicDuplicated.Hc = Hc;  MVM\_SETDIAG\_(MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF, mVMeasurementDynamicDuplicated.mvmDiagnosis);   1. Call CalculateSteamQp(MF); |
| Expected result | Qp shall be equal to QpSim  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be cleared |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 0.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_POWER;  TUSIGN8 lOpMode = CDT\_STEAM\_POWER;  T\_DEV\_RNGFLT range;  TFLOAT VF = 2.0f;  TFLOAT Hw = 60.0f, Hc = 30.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurementProtected.QpSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QpSim = 68.00f;  mVMeasurementDynamicDuplicated.Hw = Hw;  mVMeasurementDynamicDuplicated.Hc = Hc;  MVM\_SETDIAG\_(MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF, mVMeasurementDynamicDuplicated.mvmDiagnosis);   1. Call CalculateSteamQp(MF); |
| Expected result | Qp shall be equal to 34.548f  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be cleared |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 90.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_POWER;  TUSIGN8 lOpMode = CDT\_STEAM\_POWER;  T\_DEV\_RNGFLT range;  TFLOAT VF = 2.0f;  TFLOAT Hw = 60.0f, Hc = 40.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  (void)memcpy((void\*)&mVMeasurementStaticFrequentStaticDefault.QpDampingTime, &lDampingTime, sizeof(lDampingTime));  (void)memcpy((void\*)&sVMeasurementReplaceStaticFreqStaticDefault.QvLowCutOff, &lCutOff, sizeof(lCutOff));  (void)mapper\_Put((TUSIGN16)MAPPER\_IDX\_PV\_Assignment, (TINT16)WHOLE\_OBJECT, &lPvAssign);  (void)coordinator\_Put( (TUSIGN16)CDT\_IDX\_operatingMode, (TINT16)WHOLE\_OBJECT, &lOpMode );  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurement\_Put(MVM\_IDX\_QpRange,WHOLE\_OBJECT,&range);  mVMeasurementProtected.QpSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QpSim = 68.00f;  mVMeasurementDynamicDuplicated.Hw = Hw;  mVMeasurementDynamicDuplicated.Hc = Hc;  MVM\_CLRDIAG\_(MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF, mVMeasurementDynamicDuplicated.mvmDiagnosis);   1. Call CalculateSteamQp(MF); |
| Expected result | Qp shall be equal to 0  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be set |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT lDampingTime = 0.0f, lCutOff = 90.0f;  TUSIGN8 lPvAssign = MAPPER\_DEVICE\_VAR\_QM;  TUSIGN8 lOpMode = CDT\_STEAM\_POWER;  T\_DEV\_RNGFLT range;  TFLOAT VF = 2.0f;  TFLOAT Hw = 60.0f, Hc = 40.0f;  TFLOAT desiredValue = 0.0f;  TFLOAT RV = 0.00f;  (void)memcpy((void\*)&mVMeasurementStaticFrequentStaticDefault.QpDampingTime, &lDampingTime, sizeof(lDampingTime));  (void)memcpy((void\*)&sVMeasurementReplaceStaticFreqStaticDefault.QvLowCutOff, &lCutOff, sizeof(lCutOff));  (void)mapper\_Put((TUSIGN16)MAPPER\_IDX\_PV\_Assignment, (TINT16)WHOLE\_OBJECT, &lPvAssign);  (void)coordinator\_Put( (TUSIGN16)CDT\_IDX\_operatingMode, (TINT16)WHOLE\_OBJECT, &lOpMode );  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurement\_Put(MVM\_IDX\_QpRange,WHOLE\_OBJECT,&range);  mVMeasurementProtected.QpSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QpSim = 68.00f;  mVMeasurementDynamicDuplicated.Hw = Hw;  mVMeasurementDynamicDuplicated.Hc = Hc;  MVM\_CLRDIAG\_(MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF, mVMeasurementDynamicDuplicated.mvmDiagnosis);   1. Call CalculateSteamQp(MF); |
| Expected result | Qp shall be equal to 0  MVM\_ALM\_LOW\_Q\_VALUE\_CUTOFF shall be set |
| Test result | Pass |

#### Module MVMeasurement\_execute.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalcSteamDensityEXE\_MVM(TFLOAT inTReal, TFLOAT inPReal) | |
| Test function name | void CalcSteamDensityEXE\_MVM\_TEST(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   p = 6000.0;  t = 110.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 0.82686340 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   p = 6000.0;  t = 259.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 23.30983377 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   p = 16733.1004;  t = 351.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 116.1107079 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   p = 16833;  t = 351.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | Density shall be 116.1107079; |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   p = 16633;  t = 351.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 113.627462 |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following   t = 810.0;  p = 7000.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 1.0 |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   t = 20.0;  p = 100.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 998.205486 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   t = 289.26;  p = 7000.0;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_P;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | Density shall be 36.523592 ; |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   t = 280.00;  p = 3000.0;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | Density shall be 12.9607488; |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set the Precondition parameter as following   t = 230.00;  p = 3000.0;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 13.984012254 |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set the Precondition parameter as following   t = 233.85844;  p = 3000.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 15.000580859 |
| Test result | Pass |
| Test case 12 | Test procedure | 1. Set the Precondition parameter as following   t = 233.858445;  p = 3000.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 821.89487 |
| Test result | Pass |
| Test case 13 | Test procedure | 1. Set the Precondition parameter as following   t = 233.868445;  p = 3000.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 821.89487 |
| Test result | Pass |
| Test case 14 | Test procedure | 1. Set the Precondition parameter as following   t = 233.868445;  p = 3560.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p); |
| Expected result | 821.89487 |
| Test result | Pass |
| Test case 15 | Test procedure | 1. Set the Precondition parameter as following   p = 3975.9391;  t = 250;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p) STEAM\_NOT\_MATCH\_COUNTER times |
| Expected result | MVM\_ALM\_STEAM\_STATUS\_MISMATCH shall be raised |
| Test result | Pass |
| Test case 16 | Test procedure | 1. Set the Precondition parameter as following   p = 4000;  t = 250;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p) STEAM\_NOT\_MATCH\_COUNTER times |
| Expected result | MVM\_ALM\_STEAM\_STATUS\_MISMATCH shall be raised |
| Test result | Pass |
| Test case 17 | Test procedure | To test the if((inputT >= (tSelf - 0.001)) && (inputT <= (tSelf + 0.001))) with TRUE && FALSE with saturated steam   1. Set the Precondition parameter as following   p = 6000;  t = 289.62;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 28.812664 |
|  | Test result | Pass |
| Test case 18 | Test procedure | To test the if((inputT >= (tSelf - 0.001)) && (inputT <= (tSelf + 0.001))) with TRUE && FALSE with overheated steam   1. Set the Precondition parameter as following   p = 6000;  t = 289.62;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 28.812664 |
|  | Test result | Pass |
| Test case 19 | Test procedure | To test the if((inputT >= (tSelf - 0.001)) && (inputT <= (tSelf + 0.001))) with TRUE && FALSE with water   1. Set the Precondition parameter as following   p = 6000;  t = 289.62;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);     Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 732.644592 |
|  | Test result | Pass |
| Test case 20 | Test procedure | To test the if((inputT >= 300.15) && (inputT <=647.096))with TRUE && FALSE with water   1. Set the Precondition parameter as following   p = 6000;  t = 289.62;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall remain unchanged. |
|  | Test result | Pass |
| Test case 21 | Test procedure | To test the if((inputT >= 300.15) && (inputT <=647.096))with FALSE && TRUE with water   1. Set the Precondition parameter as following   p = 6000;  t = -1.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);     Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall remain unchanged. |
|  | Test result | Pass |
| Test case 22 | Test procedure | To test the if ((steamStatus != MVM\_WATER) && (alarmCounter >= STEAM\_NOT\_MATCH\_COUNTER)) with FALSE && FALSE with water   1. Set the Precondition parameter as following   p = 6000;  t = 258.00244;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);     Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 788.586547 |
|  | Test result | Pass |
| Test case 23 | Test procedure | To test the if ((steamStatus != MVM\_WATER) && (alarmCounter >= STEAM\_NOT\_MATCH\_COUNTER)) with FALSE && TRUE with water   1. Set the Precondition parameter as following   p = 6000;  t = 275.586487;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);     Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 788.002441 |
|  | Test result | Pass |
| Test case 24 | Test procedure | To test the if ((steamStatus == MVM\_WATER) && (alarmCounter >= STEAM\_NOT\_MATCH\_COUNTER)) with FALSE && FALSE with water   1. Set the Precondition parameter as following   p = 6000;  t = 275.586487;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);     Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 30.8175239 |
|  | Test result | Pass |
| Test case 25 | Test procedure | To test the if ((steamStatus == MVM\_WATER) && (alarmCounter >= STEAM\_NOT\_MATCH\_COUNTER)) with TRUE && FALSE with water   1. Set the Precondition parameter as following   p = 6000;  t = 250.586487;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);     Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | The return value shall be 20.1694908 |
|  | Test result | Pass |
| Test case 26 | Test procedure | to cover the else branch of if(volume > 0.0)   1. Set the Precondition parameter as following   p = 4000;  t = 270;  steamType = MVM\_STEAMTYPE\_SATURATED  densitySelection = CDT\_CALCULATION\_FROM\_TP;  testObjMvEx1 = 1   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call CalcSteamDensityEXE\_MVM(t,p) |
|  | Expected result | testObjMvEx = 8 |
|  | Test result | Pass |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalcSteamHEXE\_MVM (TFLOAT inTReal, TFLOAT inPReal) | |
| Test function name | void CalcSteamHEXE\_MVM\_TEST(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   p = 6000.0;  t = 110.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 4. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW is 2691.06763  HC is 461.36335 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   p = 6000.0;  t = 259.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW is 2797.19204  HC is 1129.86469 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   p = 22293.0643;  t = 376.85;  t2 = 376.85;  Hc = 60;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t2); |
| Expected result | Hw shall be 2375.12400.  Hc shall remain unchanged |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   p = 16733;  t = 351.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | Hw shall be 2556.71871  Hc shall be 1679.14401 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   t = 810.0;  p = 7000.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | Hw and Hc shall remain unchanged |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following   t = 210.0;  t2 = 130.0;  p = 7000.0;  mVMeasurementDynamicDuplicated.Hw = 98.28;  mVMeasurementDynamicDuplicated.Hc = 48.58;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 2797.35235f  HC 546.38783f |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   t = 410.0;  t2 = 130.0;  p = 3500.0;  mVMeasurementDynamicDuplicated.Hw = 98.28;  mVMeasurementDynamicDuplicated.Hc = 48.58;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_ONLY;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 3246.18374  HC 0 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   p = 11800;  t = 112.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_P;  steamPwrrFR = MVM\_FWD\_ONLY;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 2689. 88944  HC 0 |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   p = 3000;  t =233.858445;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_P;  steamPwrrFR = MVM\_FWD\_ONLY;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 2803.26474  HC 0 |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set the Precondition parameter as following   p = 3000;  t = 233. 868445;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW1008.37136  HC 1008.37136 |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set the Precondition parameter as following   p = 3100;  t = 233. 8584;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 961.67242  HC 961.67242 |
| Test result | Pass |
| Test case 12 | Test procedure | 1. Set the Precondition parameter as following   p = 3560;  t = 233. 868445;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 1008.41859  HC 1008.41859 |
| Test result | Pass |
| Test case 13 | Test procedure | 1. Set the Precondition parameter as following   p = 3560;  t = 233. 868445;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_EXT\_DENSITY;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 100.0  HC 50.0 |
| Test result | Pass |
| Test case 14 | Test procedure | 1. Set the Precondition parameter as following   p = 3560;  t = 233. 868445;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 150.0  HC 100.0 |
| Test result | Pass |
| Test case 15 | Test procedure | 1. Set the Precondition parameter as following   p = 3560;  t = 233. 868445;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_P;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | HW 200.0  HC 150.0 |
| Test result | Pass |
| Test case 16 | Test procedure | Test "if((fTInternal >= 300.15) && (fTInternal <=647.096))" with (FALSE) && (TRUE)   1. Set the Precondition parameter as following   p = 3560;  t = 200;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
|  | Expected result | HW 852.393676f  HC 852.393676f |
|  | Test result | Pass |
| Test case 17 | Test procedure | Test "if((fTInternal >= 300.15) && (fTInternal <=647.096))" with (TRUE) && (FALSE)   1. Set the Precondition parameter as following   p = 3560;  t = 700;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
|  | Expected result | HW 200  HC 150 |
|  | Test result | Pass |
| Test case 18 | Test procedure | 1. Set the Precondition parameter as following   p = 3560;  t = 733. 868445;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 3. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);      1. Call   CalcSteamDensityEXE\_MVM(t,p);  CalcSteamHEXE \_MVM(p,t,t); |
| Expected result | Code coverage: Missing else statement execution |
| Test result | Pass |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void CalcSteamHPreset() | |
| Test function name | void CalcSteamHPreset\_Test (void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   p = 6000.0;  t = 110.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 4. coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 5. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 7. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 8. Call CalcSteamHPreset |
| Expected result | Hw 2691.06763  Hc 461.36335 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   p = 6000.0;  t = 259.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | Hw: 2797.19204  Hc: 1129.86469 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   p = 22293.0643;  t = 376.85;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR,(TINT16)WHOLE\_OBJECT,&steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HW 2375.12400  HC shall remain unchanged. |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   p = 16733;  t = 351.00;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | Hw: 2556.71871  Hc: 1679.14401 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   t = 110.0;  p = 7000.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 2691.06763  HcPreset 461.36335 |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the Precondition parameter as following   t = 420.0;  p = 4000.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_ONLY;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 3261.35869  HcPreset 0.0 |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   t = 251.0;  p = 6100.0;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_P;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 2783.45948  HcPreset 1090.49165 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   t = 350.0;  p = 1633.0;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 2563.59200  HcPreset 1670.85822 |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   t = 350.5114;  p = 6100.0;  steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 2560.12563  HcPreset 1675.09322 |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set the Precondition parameter as following   t = 350.0;  p = 1633.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 2563.59200  HcPreset 1670.85822 |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set the Precondition parameter as following   t = 350.5114;  p = 6100.0;  steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 2560.12563  HcPreset 1675.09322 |
| Test result | Pass |
| Test case 12 | Test procedure | 1. Set the Precondition parameter as following   t = 350.0;  p = 1633.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 1669.88227  HcPreset 1669.88227 |
| Test result | Pass |
| Test case 13 | Test procedure | 1. Set the Precondition parameter as following   t = 350.5114;  p = 16633.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 1675.09321;  HcPreset 1675.09321 |
| Test result | Pass |
| Test case 14 | Test procedure | 1. Set the Precondition parameter as following   t = 355.5114;  p = 16633.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_TP;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 1718.33266;  HcPreset 1718.33266 |
| Test result | Pass |
| Test case 15 | Test procedure | 1. Set the Precondition parameter as following   t = 350.5114;  p = 16633.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 1675.09322;  HcPreset 1675.09322 |
| Test result | Pass |
| Test case 16 | Test procedure | 1. Set the Precondition parameter as following   t = 110.0;  p = 7000.0;  steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  steamPwrrFR = MVM\_FWD\_REV;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | HwPreset 466.36193  HcPreset 466.36193 |
| Test result | Pass |
| Test case 17 | Test procedure | 1. Set the Precondition parameter as following   steamType = MVM\_STEAMTYPE\_SATURATED;  densitySelection = CDT\_EXT\_DENSITY;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
| Expected result | Hw and Hc shall remain unchanged. |
| Test result | Pass |
| Test case 18 | Test procedure | 1. Set the Precondition parameter as following   steamType = MVM\_STEAMTYPE\_OVERHEAT;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
|  | Expected result | Hw and Hc shall remain unchanged. |
|  | Test result | Pass |
| Test case 19 | Test procedure | 1. Set the Precondition parameter as following   steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_P;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t); 7. Call CalcSteamHPreset |
|  | Expected result | Hw and Hc shall remain unchanged. |
|  | Test result | Pass |
| Test case 20 | Test procedure | Test "if((fTInternal >= 300.15) && (fTInternal <=647.096))" with (FALSE) && (TRUE)   1. Set the Precondition parameter as following   steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t);   Call CalcSteamHPreset |
|  | Expected result | Hw 852  Hc 462 |
|  | Test result | Pass |
| Test case 21 | Test procedure | Test "if((fTInternal >= 300.15) && (fTInternal <=647.096))" with (TRUE) && (FALSE)   1. Set the Precondition parameter as following   steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  t = 700   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t);   Call CalcSteamHPreset |
|  | Expected result | Hw 852  Hc 462 |
|  | Test result | Pass |
| Test case 22 | Test procedure | Test water, from T, but the temperature is not valid   1. Set the Precondition parameter as following   steamType = MVM\_STEAMTYPE\_WATER;  densitySelection = CDT\_CALCULATION\_FROM\_T;  t = 20, p = 16633   1. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType); 2. Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR, 0, &steamPwrrFR); 3. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection); 4. Call coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&t); 5. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&p); 6. Call mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&t);   Call CalcSteamHPreset |
|  | Expected result | Hw and Hc shall remain unchanged. |
|  | Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | Void CalcCompressFactor\_MVM(void) | |
| Test function name | CalcCompressFactor\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[9]= {  24.0 ,  36.0 ,  48.0 ,  60.0 ,  72.0 ,  84.0 ,  96.0 ,  108.0 ,  120.0  };  TFLOAT tArray[7] = {  -3.15,  6.85,  16.85,  26.85,  36.85,  46.85,  56.85  };  TFLOAT cFArray[63] = {  0.9357927,  0.9035867,  0.8716867,  0.8405274,  0.8107021,  0.7829730,  0.7582385,  0.7374294,  0.7213312,  0.9437102,  0.9158381,  0.8884824,  0.8619934,  0.8368118,  0.8134642,  0.7925379,  0.7746267,  0.7602510,  0.9505358,  0.9263186,  0.9027338,  0.8800623,  0.8586379,  0.8388388,  0.8210689,  0.8057272,  0.7931657,  0.9564576,  0.9353549,  0.9149431,  0.8954477,  0.8771248,  0.8602537,  0.8451230,  0.8320112,  0.8211638,  0.9616241,  0.9431986,  0.9254871,  0.9086707,  0.8929484,  0.8785306,  0.8656289,  0.8544441,  0.8451518,  0.9661544,  0.9500471,  0.9346550,  0.9201237,  0.9066092,  0.8942724,  0.8832717,  0.8737551,  0.8658513,  0.9701447,  0.9560576,  0.9426732,  0.9301090,  0.9184879,  0.9079344,  0.8985689,  0.8905025,  0.8838309  };   1. Set Gas ConfFlag.   gasConfFlag = MVM\_GONFIGING;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_updateCFPresetRequest,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Pressure and Temperature unit   unitCode = MAPPER\_BAR;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_unit,&unitCode);  unitCode = CDT\_CELSIUS;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_unit,&unitCode);   1. Write Pressure   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_value(0),&pArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray1Obj,UOM\_PF\_value(0),&pArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray2Obj,UOM\_PF\_value(0),&pArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray3Obj,UOM\_PF\_value(0),&pArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray4Obj,UOM\_PF\_value(0),&pArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray5Obj,UOM\_PF\_value(0),&pArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray6Obj,UOM\_PF\_value(0),&pArray[6]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray7Obj,UOM\_PF\_value(0),&pArray[7]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray8Obj,UOM\_PF\_value(0),&pArray[8]);   1. Write Temperature   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_value(0),&tArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray1Obj,UOM\_PF\_value(0),&tArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray2Obj,UOM\_PF\_value(0),&tArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray3Obj,UOM\_PF\_value(0),&tArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray4Obj,UOM\_PF\_value(0),&tArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray5Obj,UOM\_PF\_value(0),&tArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray6Obj,UOM\_PF\_value(0),&tArray[6]);   1. Write Compress Factor Array.   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,(TINT16)WHOLE\_OBJECT,&cFArray);   1. Set Matrix Size   mSize = MVM\_SIZE9X7;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_matrixSize,(TINT16)WHOLE\_OBJECT,&mSize);   1. Set Flag.   gasConfFlag = MVM\_GONFIGED;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Real Temperature and Pressure   tReal = 16.85;  pReal = 8400.0f;  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressure,(TINT16)WHOLE\_OBJECT,&pReal);  coordinator\_Put((TUSIGN16)CDT\_IDX\_temperature,(TINT16)WHOLE\_OBJECT,&tReal);   1. Call CalcCompressFactor\_MVM() |
| Expected result | 0.83884 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[9]= {  24.0 ,  36.0 ,  48.0 ,  60.0 ,  72.0 ,  84.0 ,  96.0 ,  108.0 ,  120.0  };  TFLOAT tArray[7] = {  -3.15,  6.85,  16.85,  26.85,  36.85,  46.85,  56.85  };  TFLOAT cFArray[63] = {  0.9357927,  0.9035867,  0.8716867,  0.8405274,  0.8107021,  0.7829730,  0.7582385,  0.7374294,  0.7213312,  0.9437102,  0.9158381,  0.8884824,  0.8619934,  0.8368118,  0.8134642,  0.7925379,  0.7746267,  0.7602510,  0.9505358,  0.9263186,  0.9027338,  0.8800623,  0.8586379,  0.8388388,  0.8210689,  0.8057272,  0.7931657,  0.9564576,  0.9353549,  0.9149431,  0.8954477,  0.8771248,  0.8602537,  0.8451230,  0.8320112,  0.8211638,  0.9616241,  0.9431986,  0.9254871,  0.9086707,  0.8929484,  0.8785306,  0.8656289,  0.8544441,  0.8451518,  0.9661544,  0.9500471,  0.9346550,  0.9201237,  0.9066092,  0.8942724,  0.8832717,  0.8737551,  0.8658513,  0.9701447,  0.9560576,  0.9426732,  0.9301090,  0.9184879,  0.9079344,  0.8985689,  0.8905025,  0.8838309  };   1. Set GasConfFlag.   gasConfFlag = MVM\_GONFIGING;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_updateCFPresetRequest,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Pressure and Temperature unit   unitCode = MAPPER\_BAR;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_unit,&unitCode);  unitCode = CDT\_CELSIUS;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_unit,&unitCode);   1. Write Pressure   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_value(0),&pArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray1Obj,UOM\_PF\_value(0),&pArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray2Obj,UOM\_PF\_value(0),&pArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray3Obj,UOM\_PF\_value(0),&pArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray4Obj,UOM\_PF\_value(0),&pArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray5Obj,UOM\_PF\_value(0),&pArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray6Obj,UOM\_PF\_value(0),&pArray[6]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray7Obj,UOM\_PF\_value(0),&pArray[7]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray8Obj,UOM\_PF\_value(0),&pArray[8]);   1. Write Temperature   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_value(0),&tArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray1Obj,UOM\_PF\_value(0),&tArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray2Obj,UOM\_PF\_value(0),&tArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray3Obj,UOM\_PF\_value(0),&tArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray4Obj,UOM\_PF\_value(0),&tArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray5Obj,UOM\_PF\_value(0),&tArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray6Obj,UOM\_PF\_value(0),&tArray[6]);   1. Write Compress Factor Array.   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,(TINT16)WHOLE\_OBJECT,&cFArray);   1. Set Matrix Size   mSize = MVM\_SIZE9X7;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_matrixSize,(TINT16)WHOLE\_OBJECT,&mSize);   1. Set Flag.   gasConfFlag = MVM\_GONFIGED;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Real Temperature and Pressure   tReal = 36.85;  pReal = 9600.0;  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressure,(TINT16)WHOLE\_OBJECT,&pReal);  coordinator\_Put((TUSIGN16)CDT\_IDX\_temperature,(TINT16)WHOLE\_OBJECT,&tReal);   1. Call CalcCompressFactor\_MVM() |
| Expected result | 0. 86563 |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[3]= {  72.0 ,  84.0 ,  96.0 ,    };  TFLOAT tArray[3] = {  6.85,  16.85,  26.85  };    TFLOAT cFArray[9] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.87712 ,  0.86025 ,  0.84512  };   1. Set Gas ConfFlag as configuring.   gasConfFlag = MVM\_GONFIGING;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_updateCFPresetRequest,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Pressure and Temperature unit   unitCode = MAPPER\_BAR;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_unit,&unitCode);  unitCode = CDT\_CELSIUS;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_unit,&unitCode);   1. Write Pressure   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_value(0),&pArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray1Obj,UOM\_PF\_value(0),&pArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray2Obj,UOM\_PF\_value(0),&pArray[2]);     1. Write Temperature   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_value(0),&tArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray1Obj,UOM\_PF\_value(0),&tArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray2Obj,UOM\_PF\_value(0),&tArray[2]);     1. Write Compress Factor Array.   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_0,&cFArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_1,&cFArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_2,&cFArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_3,&cFArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_4,&cFArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_5,&cFArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_6,&cFArray[6]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_7,&cFArray[7]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,ATTRIB\_8,&cFArray[8]);   1. Set Matrix Size   mSize = MVM\_SIZE3X3;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_matrixSize,(TINT16)WHOLE\_OBJECT,&mSize);   1. Set Flag.   gasConfFlag = MVM\_GONFIGED;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Real Temperature and Pressure   tReal = 26.85;  pReal = 8400.0f mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressure,(TINT16)WHOLE\_OBJECT,&pReal);  coordinator\_Put((TUSIGN16)CDT\_IDX\_temperature,(TINT16)WHOLE\_OBJECT,&tReal);   1. Call CalcCompressFactor\_MVM() |
| Expected result | 0.86025 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[4]= {  72.0 ,  84.0 ,  96.0 ,  108.0  };    TFLOAT tArray[3] = {  6.85,  16.85,  26.85  };    TFLOAT cFArray[12] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.80573 ,    0.87712 ,  0.86025 ,  0.84512 ,  0.83201  };   1. Set Gas ConfFlag .   gasConfFlag = MVM\_GONFIGING;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_updateCFPresetRequest,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Pressure and Temperature unit   unitCode = MAPPER\_BAR;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_unit,&unitCode);  unitCode = CDT\_CELSIUS;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_unit,&unitCode);   1. Write Pressure   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_value(0),&pArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray1Obj,UOM\_PF\_value(0),&pArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray2Obj,UOM\_PF\_value(0),&pArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray3Obj,UOM\_PF\_value(0),&pArray[3]);     1. Write Temperature   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_value(0),&tArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray1Obj,UOM\_PF\_value(0),&tArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray2Obj,UOM\_PF\_value(0),&tArray[2]);     1. Write Compress Factor Array.   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,0,&cFArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,1,&cFArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,2,&cFArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,3,&cFArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,4,&cFArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,5,&cFArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,6,&cFArray[6]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,7,&cFArray[7]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,8,&cFArray[8]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,9,&cFArray[9]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,10,&cFArray[10]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,11,&cFArray[11]);   1. Set Matrix Size   mSize = MVM\_SIZE4X3;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_matrixSize,(TINT16)WHOLE\_OBJECT,&mSize);   1. Set Flag .   gasConfFlag = MVM\_GONFIGED;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Real Temperature and Pressure   tReal = 26.85;  pReal = 10800.0f;  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressure,(TINT16)WHOLE\_OBJECT,&pReal);  coordinator\_Put((TUSIGN16)CDT\_IDX\_temperature,(TINT16)WHOLE\_OBJECT,&tReal);   1. Call CalcCompressFactor\_MVM() |
| Expected result | 0.8320112 |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT pArray[4]= {  72.0 ,  84.0 ,  101.325f ,  108.0  };    TFLOAT tArray[3] = {  6.85,  20.0f,  26.85  };    TFLOAT cFArray[12] = {  0.83681 ,  0.81346 ,  0.79254 ,  0.77463 ,  0.85864 ,  0.83884 ,  0.82107 ,  0.80573 ,    0.87712 ,  0.86025 ,  0.84512 ,  0.83201  };   1. Set GasConfFlag.   gasConfFlag = MVM\_GONFIGING;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_updateCFPresetRequest,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Pressure and Temperature unit   unitCode =MAPPER\_KILOPASCAL;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_unit,&unitCode);  unitCode = CDT\_CELSIUS;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_unit,&unitCode);   1. Write Pressure   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray0Obj,UOM\_PF\_value(0),&pArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray1Obj,UOM\_PF\_value(0),&pArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray2Obj,UOM\_PF\_value(0),&pArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFPressArray3Obj,UOM\_PF\_value(0),&pArray[3]);     1. Write Temperature   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray0Obj,UOM\_PF\_value(0),&tArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray1Obj,UOM\_PF\_value(0),&tArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_CFTempArray2Obj,UOM\_PF\_value(0),&tArray[2]);     1. Write Compress Factor Array.   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,0,&cFArray[0]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,1,&cFArray[1]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,2,&cFArray[2]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,3,&cFArray[3]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,4,&cFArray[4]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,5,&cFArray[5]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,6,&cFArray[6]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,7,&cFArray[7]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,8,&cFArray[8]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,9,&cFArray[9]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,10,&cFArray[10]);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorArray,11,&cFArray[11]);   1. Set Matrix Size   mSize = MVM\_SIZE4X3;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_matrixSize,(TINT16)WHOLE\_OBJECT,&mSize);   1. Set Flag.   gasConfFlag = MVM\_GONFIGED;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Set Real Temperature and Pressure     TUSIGN8 gasRef = MVM\_DEG20;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasRef,(TINT16)WHOLE\_OBJECT,&gasRef);   1. Call CalcCompressFactor\_MVM() |
| Expected result | 0.82107 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQmEXE\_MVM(TFLOAT inVF, TFLOAT inNVF, TFLOAT densityReal) | |
| Test function name | void CalculateGasQmEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT VF = 89.580f;  TFLOAT NF = 1128.580f;  TFLOAT dRef = 1.293f;  TFLOAT dReal = 13.987f;  TUSIGN8 gasDensitySelection= MVM\_REFERENCY\_DENSITY;  mVMeasurementUnprotected.QmSim = 28862.00f;  mVMeasurementUnprotected.QmPerSim = 75.00f;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_QM\_VALUE;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;    mVMeasurementDynamicDuplicated.QmMaxDN = 30000.00f T\_DEV\_RNGFLT range;  range.hiLim = 30000.00f;  range.loLim = 100.0f;  range.span = 29900.0f;  Call mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QmRange,(TINT16)WHOLE\_OBJECT,&range); mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasDensitySelection,(TINT16)WHOLE\_OBJECT,&gasDensitySelection);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_densityRef,(TINT16)WHOLE\_OBJECT,&dRef);     1. Call CalculateGasQmEXE\_MVM(VF,NF,dReal); |
| Expected result | 28862.000000f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQnEXE\_MVM(TFLOAT volumeFlow, TFLOAT tReal, TFLOAT pReal) | |
| Test function name | void CalculateGasQnEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT VF = 1127.580f;  TBOOL result = eFALSE;    TFLOAT tRef = 21.35f;  TFLOAT tReal = 31.25f;  TFLOAT pRef = 105.36f;  TFLOAT pReal = 235.23f;  mVMeasurementDynamicDuplicated.compressFactorS = 0.8956f;  mVMeasurementDynamicDuplicated.compressFactorR = 0.8896f;  range.hiLim = 30000.00f;  range.loLim = 10.0f;  range.span = 29990.0f;  mVMeasurementDynamicDuplicated.QnMaxDN = 30000.00f ;  mVMeasurementUnprotected.QnSim = 28862.00f;  mVMeasurementUnprotected.QnPerSim = 75.00f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QnRange,(TINT16)WHOLE\_OBJECT,&range);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_refTemperature,(TINT16)WHOLE\_OBJECT,&tRef);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_refPressure,(TINT16)WHOLE\_OBJECT,&pRef);    coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  mVMeasurementProtected.QnSimEnable = MVM\_ENABLE;   1. Call CalculateGasQnEXE\_MVM(VF,tReal,pReal); |
| Expected result | 28862.0000f; |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQnPartialEXE\_MVM(TFLOAT inNVF, TFLOAT biogasPerc) | |
| Test function name | void CalculateGasQnPartialEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT NF = 67.870f;  TBOOL result = eFALSE;  TFLOAT biaGas = 56.00f;  T\_DEV\_RNGFLT range;  range.hiLim = 72.00f;  range.loLim = 2.0f;  range.span = 70.0f;  mVMeasurementDynamicDuplicated.QnMaxDN = 80.00f ;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QnPartialRange,(TINT16)WHOLE\_OBJECT,&range);  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  mVMeasurementProtected.QnPartialSimEnable = MVM\_DISABLE;  mVMeasurementUnprotected.QnPartialSim = 65.00f;  mVMeasurementUnprotected.QnPartialPerSim = 78.00f;     1. Call CalculateGasQnPartialEXE\_MVM(NF,biaGas); |
| Expected result | 38.007200f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQvPartialEXE\_MVM(TFLOAT inVF, TFLOAT biogasPerc) | |
| Test function name | void CalculateGasQvPartialEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT VF = 57.580f;  TBOOL result = eFALSE;  mVMeasurementUnprotected.QvPartialSim = 68.00f;  TFLOAT biaGas = 60.00f;  TFLOAT RV = 0.00f;  TFLOAT DValue = 34.548f;  T\_DEV\_RNGFLT range;  range.hiLim = 79.00f;  range.loLim = 2.0f;  range.span = 77.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QvPartialRange,(TINT16)WHOLE\_OBJECT,&range);  mVMeasurementProtected.QvPartialSimEnable = MVM\_DISABLE;  RV = CalculateGasQvPartialEXE\_MVM(VF,biaGas);   1. Call |
| Expected result |  |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateGasQpEXE\_MVM(TFLOAT inVF) | |
| Test function name | void CalculateGasQpEXE\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT VF = 57.580f;  TBOOL result = eFALSE;  mVMeasurementUnprotected.QpSim = 2040000.00f;  TFLOAT calorificEnergy = 30.0f;  T\_DEV\_RNGFLT range;  range.hiLim = 2370000.00f;  range.loLim = 1000.0f;  range.span = 2369000.0f;  mVMeasurementDynamicDuplicated.QpMaxDN = 2400000.00f ;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QpRange,(TINT16)WHOLE\_OBJECT,&range);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_calorificEnergy,(TINT16)WHOLE\_OBJECT,&calorificEnergy);  mVMeasurementProtected.QpSimEnable = MVM\_ENABLE;   1. Call CalculateGasQpEXE\_MVM(VF); |
| Expected result | 2040000.000f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateLiquidQmEXE\_MVM(TFLOAT inVF, TFLOAT DensityReal, TFLOAT tReal) | |
| Test function name | void CalculateLiquidQmEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT VF = 77.580f;  TBOOL result = eFALSE;  TFLOAT densityReal =998.89 ;  TFLOAT tReal = 31.25f;  TFLOAT lCutOff = 1.0;  mVMeasurementUnprotected.QmPerSim = 70.00f;  mVMeasurementUnprotected.QmSim = 78800.00000f;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_OFF;  T\_DEV\_RNGFLT range;  range.hiLim = 79800.00f;  range.loLim = 100.0f;  range.span = 79700.0f;    mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QmRange,(TINT16)WHOLE\_OBJECT,&range);  sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvLowCutOff,(TINT16)WHOLE\_OBJECT,&lCutOff);   1. Call CalculateLiquidQmEXE\_MVM(VF,densityReal,tReal); |
| Expected result | 77493.8862f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateLiquidQnEXE\_MVM(TFLOAT inVF, TFLOAT tReal) | |
| Test function name | void CalculateLiquidQnEXE\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TUSIGN8 result = eFALSE;  TFLOAT VF = 87.58f;  mVMeasurementUnprotected.QnSim = 78.00f;  TFLOAT tReal = 31.25f;  TFLOAT tRef = 21.35f;  mVMeasurementDynamicDuplicated.QnMaxDN = 80.00f ;  T\_DEV\_RNGFLT range;  range.hiLim = 80.00f;  range.loLim = 2.10f;  range.span = 77.90f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QnRange,(TINT16)WHOLE\_OBJECT,&range);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_refTemperature,(TINT16)WHOLE\_OBJECT,&tRef);    mVMeasurementProtected.QnSimEnable = MVM\_ENABLE;   1. Call = CalculateLiquidQnEXE\_MVM(VF,tReal); |
| Expected result | 78.00f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateLiquidQpEXE\_MVM(TFLOAT inVF, TFLOAT T1Real, TFLOAT T2Real) | |
| Test function name | void CalculateLiquidQpEXE\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT MF = 8240.580f;  TFLOAT tReal = 121.25f;  TFLOAT tExtReal = 112.25f;  mVMeasurementUnprotected.QpSim = 298493.99000;  mVMeasurementProtected.QpSimEnable = MVM\_ENABLE;  T\_DEV\_RNGFLT range;  range.hiLim = 320000.00f;  range.loLim = 100.0f;  range.span = 319900.0f;  mVMeasurementDynamicDuplicated.QpMaxDN = 320000.00f ;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QpRange,(TINT16)WHOLE\_OBJECT,&range);     1. Call CalculateLiquidQpEXE\_MVM(MF,tReal,tExtReal); |
| Expected result | 298493.99000f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateSteamQmEXE\_MVM(TFLOAT inVF, TFLOAT DensityReal) | |
| Test function name | void CalculateSteamQmEXE\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT VF = 77.580f;  TFLOAT densityReal = 902.5270758f;  mVMeasurementUnprotected.QmPerSim = 69.00f;  mVMeasurementUnprotected.QmSim = 78430.000f;  mVMeasurementProtected.QmSimEnable = MVM\_DISABLE;  coordinatorProtected.simulationMode = CDT\_SIMULATION\_CB\_QM\_PERCENTAGE;  T\_DEV\_RNGFLT range;  range.hiLim = 78890.00f;  range.loLim = 100.0f;  range.span = 78790.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QmRange,(TINT16)WHOLE\_OBJECT,&range);     1. Call CalculateSteamQmEXE\_MVM(VF,densityReal); |
| Expected result | 54434.1000f; |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT CalculateSteamQpEXE\_MVM(TFLOAT inMF) | |
| Test function name | void CalculateSteamQpEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT MF = 740.580f;  mVMeasurementUnprotected.QpSim = 298493.99000;  mVMeasurementProtected.QpSimEnable = MVM\_DISABLE;  mVMeasurementDynamicDuplicated.Hw = 2685.57;  mVMeasurementDynamicDuplicated.Hc = 2634.83;  T\_DEV\_RNGFLT range;  range.hiLim = 320000.00f;  range.loLim = 100.0f;  range.span = 319900.0f;  mVMeasurementDynamicDuplicated.QpMaxDN = 320000.00f ;  mVMeasurement\_Put((TUSIGN16)(TUSIGN16)MVM\_IDX\_QpRange,(TINT16)WHOLE\_OBJECT,&range);     1. Call CalculateSteamQpEXE\_MVM(MF); |
| Expected result | 37577.02920f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT GetQmPercentageSRV\_MVM(void) | |
| Test function name | void GetQmPercentageSRV\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT DValue =;  mVMeasurementDynamicDuplicated.QmPercentage = DValue;   1. Call GetQmPercentageSRV\_MVM(); |
| Expected result | 20.0f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT GetQnPartialPercentageSRV\_MVM(void) | |
| Test function name | void GetQnPartialPercentageSRV \_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT DValue = 70.0f;  TFLOAT per = 0.0f;  mVMeasurementDynamicDuplicated.QnPartialPercentage = DValue;   1. Call GetQnPartialPercentageSRV\_MVM(); |
| Expected result | 70.0f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT GetQnPercentageSRV\_MVM(void) | |
| Test function name | void GetQnPercentageSRV\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT DValue = 50.0f  TFLOAT per = 0.0f;  mVMeasurementDynamicDuplicated.QnPercentage = DValue;  //change if get the right value  per =   1. Call GetQnPercentageSRV\_MVM(); |
| Expected result | 50.0f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT GetQpPercentageSRV\_MVM(void) | |
| Test function name | void GetQpPercentageSRV\_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT DValue = 85.0f;  mVMeasurementDynamicDuplicated.QpPercentage = DValue;     1. Call GetQpPercentageSRV\_MVM(); |
| Expected result | 85.0f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT GetQvPartialPercentageSRV\_MVM(void) | |
| Test function name | void GetQvPartialPercentageSRV \_MVM\_Test() | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT DValue = 60.0f;  TFLOAT per = 0.0f;  mVMeasurementDynamicDuplicated.QvPartialPercentage = DValue;   1. Call GetQvPartialPercentageSRV\_MVM(); |
| Expected result | 70.0f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void CheckDataCrcSRV\_MVM(void) | |
| Test function name | void CheckDataCrcSRV\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   elettronicServices.checkDataCrc\_value = 0xA5;  mVMeasurementProtected.protectedDataStatus = MVM\_CALCULATE;     1. Call CheckDataCrcSRV\_MVM(); |
| Expected result | elettronicServices.checkDataCrc\_value == 0xA5; |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   elettronicServices.checkDataCrc\_value = 0xA5;  mVMeasurementProtected.protectedDataStatus = MVM\_CALCULATE;   1. Call CheckDataCrcSRV\_MVM(); 2. Clear mVMeasurementUnprotected.protectedDataCrc   mVMeasurementUnprotected.protectedDataCrc= 0;   1. Call CheckDataCrcSRV\_MVM(); |
| Expected result | elettronicServices.checkDataCrc\_value == 0x5A |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   elettronicServices.checkDataCrc\_value = 0xA5;  mVMeasurementProtected.protectedDataStatus = MVM\_PROTECTED + 0X01;     1. Call CheckDataCrcSRV\_MVM(); |
| Expected result | elettronicServices.checkDataCrc\_value == 0x5A |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateDiagnosisEXE\_MVM(void) | |
| Test function name | void UpdateDiagnosisEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM\_SETDIAG\_( MVM\_ALM\_FLOWRATE\_REACH\_QMAX ,mVMeasurementDynamicDuplicated.mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_FLOWRATE\_REACH\_QMAX is set in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM\_CLRDIAG\_( MVM\_ALM\_FLOWRATE\_REACH\_QMAX ,mVMeasurementDynamicDuplicated.mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_FLOWRATE\_REACH\_QMAX is reset in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM\_SETDIAG\_(MVM\_ALM\_FLOWRATE\_REACH\_QMIN,mVMeasurementDynamicDuplicated.mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_FLOWRATE\_REACH\_QMIN is set in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM\_CLRDIAG\_\_(MVM\_ALM\_FLOWRATE\_REACH\_QMIN,mVMeasurementDynamicDuplicated.mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_FLOWRATE\_REACH\_QMIN is reset in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM\_CLRDIAG\_\_(MVM\_ALM\_FLOWRATE\_GT\_103,mVMeasurementDynamicDuplicated.mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_FLOWRATE\_GT\_103 is reset in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM\_CLRDIAG\_\_(MVM\_ALM\_FLOWRATE\_GT\_103,mVMeasurementDynamicDuplicated.mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_FLOWRATE\_GT\_103 is reset in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM \_CLRDIAG\_( MVM\_ALM\_STEAM\_STATUS\_MISMATCH , mVMeasurementDynamicDuplicated. mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_STEAM\_STATUS\_MISMATCH Is reset in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. MVM \_CLRDIAG(MVM\_ALM\_STEAM\_STATUS\_MISMATCH , mVMeasurementDynamicDuplicated. mvmDiagnosis); 3. Call UpdateDiagnosisEXE\_MVM(); |
| Expected result | MVM\_ALM\_STEAM\_STATUS\_MISMATCH is reset in diagnosisDynamic.mvmeasurementAlarm |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set DIAGNOSIS\_ME->ptrState as INITIALIZED; 2. diagnosisSimulationStatus = DIAGNOSIS\_SIMULATION\_ENABLED; 3. Set Alarm Index i as 0x00 4. Call mVMeasurement\_Put((TINT16)MVM\_IDX\_mvmAlarmSimulation, i, &alarmBitSimulation); 5. Call UpdateDiagnosisEXE\_MVM(); 6. Repeat Step 3 and 4 untill i reach to MVM\_DIAG\_SIZE |
| Expected result | Set alarm bit by bit from 0b00000001 to 0b0001000 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name |  | |
| Test function name | Void UpdateBiagasQvMaxDN\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT QvMaxDn = 30000.00f;  TFLOAT QvBiogasMaxDN = 0.0f;  TFLOAT gasPresetContent = 68.0f;    sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvMaxDN,(TINT16)WHOLE\_OBJECT,&QvMaxDn);  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvContentPreset,(TINT16)WHOLE\_OBJECT,&gasPresetContent);   1. Call UpdateBiagasQvMaxDN\_MVM(); |
| Expected result | - 20400.0f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | Void UpdateGasQnMaxDN\_MVM(void) | |
| Test function name | void UpdateGasQnMaxDN\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT inTPreset = 28.0f;  TFLOAT inPPreset = 801.0f;  TFLOAT QvMaxDn = 78.00f;  TUSIGN8 gasConfFlag = MVM\_NONGONFIGED;  TUSIGN8 gasRef = MVM\_UK;    coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&inTPreset);  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&inPPreset);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasRef,(TINT16)WHOLE\_OBJECT,&gasRef);  sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvMaxDN,(TINT16)WHOLE\_OBJECT,&QvMaxDn);  (void) mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);     1. Call UpdateGasQnMaxDN\_MVM(); |
| Expected result | 590.61938f |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT inTPreset = 28.0f;  TFLOAT inPPreset = 801.0f;  TFLOAT QvMaxDn = 78.00f;  TFLOAT compressFactorS = 0.9856f;  TFLOAT compressFactorPreset = 0.9536f;  TUSIGN8 gasRef = MVM\_UK;  TUSIGN8 gasConfFlag = MVM\_GONFIGED;  TUSIGN8 gasComputeType = MVM\_ISO12213\_MOLAR;  coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&inTPreset);  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&inPPreset);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasRef,(TINT16)WHOLE\_OBJECT,&gasRef);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorS,(TINT16)WHOLE\_OBJECT,&compressFactorS);  (void)mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_compressFactorPreset,(TINT16)WHOLE\_OBJECT,&compressFactorPreset);  (void)mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasConfFlag,(TINT16)WHOLE\_OBJECT,&gasConfFlag);  (void)mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasComputeType,(TINT16)WHOLE\_OBJECT,&gasComputeType);     1. Call UpdateGasQnMaxDN\_MVM(); |
| Expected result | 610.43882f |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the Precondition parameter as following   gasRef = MVM\_US;  updateCFSRequest = MVM\_NONGONFIGED;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_gasRef,(TINT16)WHOLE\_OBJECT,&gasRef);  mVMeasurement\_ Put ((TUSIGN16)MVM\_IDX\_,(TINT16)WHOLE\_OBJECT,&gasConfFlag);   1. Call UpdateGasQnMaxDN\_MVM(); |
| Expected result | updateCFSRequest is set as MVM\_GONFIGING |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateQmMaxDN\_MVM(void) | |
| Test function name | void UpdateQmMaxDN\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT density = 988.0f;  TFLOAT QvMaxDn = 78.00f;  mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvDensityPreset,(TINT16)WHOLE\_OBJECT,&density);  sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvMaxDN,(TINT16)WHOLE\_OBJECT,&QvMaxDn);     1. Call UpdateQmMaxDN\_MVM(); |
| Expected result | 77064.00f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateGasPowerMaxDN \_MVM(void) | |
| Test function name | void UpdateGasPowerMaxDN\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT calorificEnergy = 36.0f;  TFLOAT QnMaxDn = 78.00f;  mVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QnMaxDN,(TINT16)WHOLE\_OBJECT,&QnMaxDn);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_calorificEnergy,(TINT16)WHOLE\_OBJECT,&calorificEnergy);     1. Call UpdateGasPowerMaxDN \_MVM(); |
| Expected result | 2808000.00f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateLiquidPowerMaxDN \_MVM(void) | |
| Test function name | void UpdateLiquidPowerMaxDN\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT density = 988.0f;  TFLOAT QvMaxDn = 78.00f;  TFLOAT heatCapacity = 4.200;    mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvDensityPreset,(TINT16)WHOLE\_OBJECT,&density);  sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvMaxDN,(TINT16)WHOLE\_OBJECT,&QvMaxDn);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_heatCapacity,(TINT16)WHOLE\_OBJECT,&heatCapacity);     1. Call UpdateLiquidPowerMaxDN \_MVM(); |
| Expected result | 25893504.00f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateLiquidQnMaxDN \_MVM(void) | |
| Test function name | void UpdateLiquidQnMaxDN\_MVM\_Test (void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT inTPreset = 35.0f;  TFLOAT refTemperature = 15.00f;  TFLOAT QvMaxDn = 78.00f;  TFLOAT volumeExpandBeta1 = 0.036f;    coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&inTPreset);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_volumeExpandBeta1,(TINT16)WHOLE\_OBJECT,&volumeExpandBeta1);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_refTemperature,(TINT16)WHOLE\_OBJECT,&refTemperature);  sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvMaxDN,(TINT16)WHOLE\_OBJECT,&QvMaxDn);     1. Call UpdateLiquidQnMaxDN \_MVM(); |
| Expected result | 45.34884f |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateSteamPowerMaxDN \_MVM(void) | |
| Test function name | void UpdateSteamPowerMaxDN\_MVM\_Test (void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TFLOAT QvMaxDn = 78.00f;  TFLOAT TPreset = 235.0 ;  TFLOAT PPreset = 8587.7 ;  TFLOAT TxtPreset = 160.00 ;  TFLOAT density = 16.00 ;  TUSIGN8 steamType = MVM\_STEAMTYPE\_SATURATED;  TUSIGN8 aiSelection = AI\_TEMPERATURE;  TUSIGN8 densitySelection = CDT\_CALCULATION\_FROM\_TP;   1. Call functions for QvMaxDN   sVMeasurement\_Put((TUSIGN16)SVM\_IDX\_QvMaxDN,(TINT16)WHOLE\_OBJECT,&QvMaxDn)   1. Call functions for QmMaxDN   mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvDensityPreset,(TINT16)WHOLE\_OBJECT,&density);   1. Call functions for Hw and Hc   mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamPwrrFR,(TINT16)WHOLE\_OBJECT,&steamPwrrFR);  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_steamType,(TINT16)WHOLE\_OBJECT,&steamType);  (void) coordinator\_Put((TUSIGN16)CDT\_IDX\_temperaturePreset,(TINT16)WHOLE\_OBJECT,&TPreset);  (void)coordinator\_Put((TUSIGN16)CDT\_IDX\_actualDensitySelection,(TINT16)WHOLE\_OBJECT,&densitySelection);  (void) mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvPressurePreset,(TINT16)WHOLE\_OBJECT,&PPreset);  (void) mapper\_Put((TUSIGN16)MAPPER\_IDX\_dvTExtPreset,(TINT16)WHOLE\_OBJECT,&TxtPreset);     1. Call UpdateSteamPowerMaxDN \_MVM(); |
| Expected result | 2653597.670506 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void UpdateQAlarmEXE\_MVM (void) | |
| Test function name | void UpdateQAlarmEXE\_MVM\_Test(void) | |
| Test case 1 | Test procedure | Testing for setting Reach Qmax Alarm   1. Prepare testing.Set Alarm Range Limits   T\_RANGE\_FLT QAlmRange;  TUSIGN8 mvmDiag;  TFLOAT pvPerc = 0.0f;  QAlmRange.hiLim = 100.00f;  QAlmRange.loLim = 2.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QAlmRange,(TINT16)WHOLE\_OBJECT,&QAlmRange);     1. clear all alarm   mVMeasurementDynamicDuplicated.mvmDiagnosis = 0x00;   1. compare value     pvPerc = 100.1f;  UpdateQAlarmEXE\_MVM(pvPerc);     1. check if set alarm   mvmDiag = (0X01 << MVM\_ALM\_FLOWRATE\_REACH\_QMAX);  TEST( (mVMeasurementDynamicDuplicated.mvmDiagnosis & mvmDiag) == mvmDiag); |
| Expected result | mVMeasurementDynamicDuplicated.mvmDiagnosis bit index 0 is set. |
| Test result | Pass |
| Test case 2 | Test procedure | Testing for Clearing Reach Qmax Alarm   1. Prepare testing.Set Alarm Range Limits   QAlmRange.hiLim = 98.00f;  QAlmRange.loLim = 4.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QAlmRange,(TINT16)WHOLE\_OBJECT,&QAlmRange);     1. Set Reaching Qmax Alarm   mvmDiag = (0X01 << MVM\_ALM\_FLOWRATE\_REACH\_QMAX);  mVMeasurementDynamicDuplicated.mvmDiagnosis = mvmDiag;   1. compare value     pvPerc = 97.0f;  UpdateQAlarmEXE\_MVM(pvPerc);     1. check if clear alarm   TEST( mVMeasurementDynamicDuplicated.mvmDiagnosis == 0x00); |
| Expected result | mVMeasurementDynamicDuplicated.mvmDiagnosis bit index 0 is reset. |
| Test result | Pass |
| Test case 3 | Test procedure | Testing for setting Reach Qmin Alarm   1. Prepare testing.Set Alarm Range Limits   T\_RANGE\_FLT QAlmRange;  TUSIGN8 mvmDiag;  TFLOAT pvPerc = 0.0f;  QAlmRange.hiLim = 100.00f;  QAlmRange.loLim = 2.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QAlmRange,(TINT16)WHOLE\_OBJECT,&QAlmRange);     1. clear all alarm   mVMeasurementDynamicDuplicated.mvmDiagnosis = 0x00;   1. compare value     pvPerc = 1.0f;  UpdateQAlarmEXE\_MVM(pvPerc);     1. check if set alarm   mvmDiag = (0X01 << MVM\_ALM\_FLOWRATE\_REACH\_QMIN);  TEST( (mVMeasurementDynamicDuplicated.mvmDiagnosis & mvmDiag) == mvmDiag); |
| Expected result | mVMeasurementDynamicDuplicated.mvmDiagnosis bit index 1 is set. |
| Test result | Pass |
| Test case 4 | Test procedure | Testing for Clearing Reach Qmin Alarm   1. Prepare testing.Set Alarm Range Limits   QAlmRange.hiLim = 98.00f;  QAlmRange.loLim = 4.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QAlmRange,(TINT16)WHOLE\_OBJECT,&QAlmRange);     1. Set Reaching Qmax Alarm   mvmDiag = (0X01 << MVM\_ALM\_FLOWRATE\_REACH\_QMAX);  mVMeasurementDynamicDuplicated.mvmDiagnosis = mvmDiag;   1. compare value     pvPerc = 5.0f;  UpdateQAlarmEXE\_MVM(pvPerc);     1. check if clear alarm   TEST( mVMeasurementDynamicDuplicated.mvmDiagnosis == 0x00); |
| Expected result | mVMeasurementDynamicDuplicated.mvmDiagnosis bit index 1 is reset. |
| Test result | Pass |
| Test case 5 | Test procedure | Testing for setting Big than 103 Alarm   1. Prepare testing.Set Alarm Range Limits   T\_RANGE\_FLT QAlmRange;  TUSIGN8 mvmDiag;  TFLOAT pvPerc = 0.0f;  QAlmRange.hiLim = 100.00f;  QAlmRange.loLim = 2.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QAlmRange,(TINT16)WHOLE\_OBJECT,&QAlmRange);     1. clear all alarm   mVMeasurementDynamicDuplicated.mvmDiagnosis = 0x00;   1. compare value     pvPerc = 103.1f;  UpdateQAlarmEXE\_MVM(pvPerc);     1. check if set alarm   mvmDiag = (0X01 << MVM\_ALM\_FLOWRATE\_\_GT\_103);  TEST( (mVMeasurementDynamicDuplicated.mvmDiagnosis & mvmDiag) == mvmDiag); |
| Expected result | mVMeasurementDynamicDuplicated.mvmDiagnosis bit index 2 is set. |
| Test result | Pass |
| Test case 6 | Test procedure | Testing for Clearing Reach Qmin Alarm   1. Prepare testing.Set Alarm Range Limits   QAlmRange.hiLim = 98.00f;  QAlmRange.loLim = 4.0f;  mVMeasurement\_Put((TUSIGN16)MVM\_IDX\_QAlmRange,(TINT16)WHOLE\_OBJECT,&QAlmRange);     1. Set Reaching Qmax Alarm   mvmDiag = (0X01 << MVM\_ALM\_FLOWRATE\_REACH\_QMAX);  mVMeasurementDynamicDuplicated.mvmDiagnosis = mvmDiag;   1. compare value     pvPerc = 102.9f;  UpdateQAlarmEXE\_MVM(pvPerc);     1. check if clear alarm   TEST( mVMeasurementDynamicDuplicated.mvmDiagnosis == 0x01); |
| Expected result | mVMeasurementDynamicDuplicated.mvmDiagnosis is 0x01 |
| Test result | Pass |

#### Module MVMeasurement\_overload.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TUSIGN16 Initialize\_MVM(const T\_UNIT SLOW\*me, TUSIGN8 typeOfStartUp) | |
| Test function name | void Initialize\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the Precondition parameter as following   TUSIGN16 result = (TINT16)FATAL\_ERROR;  mVMeasurementUnprotected.QMinDN = 50.00f;  mVMeasurementUnprotected.protectedDataCrc = 0x55;    mVMeasurementDynamicDuplicated.Qn = 80.0f;  mVMeasurementDynamicDuplicated.mvmAlarmSimulationDuplicated = 0xAA;     1. Call result =Initialize\_MVM(MVMEASUREMENT\_ME, INIT\_DATA); |
| Expected result | result == OK  mVMeasurementUnprotected.QminDN == 0.0  mVMeasurementUnprotected.protectedDataCrc == 0X0000  mVMeasurementDynamicDuplicated.Qn == 0.0f  mVMeasurementDynamicDuplicated.mvmAlarmSimulationDuplicated == 0X00 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the Precondition parameter as following   assertTestOK=0   1. Call Initialize\_MVM(SVMEASUREMENT\_ME, INIT\_DATA) |
| Expected result | assertTestOK== 1 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TUSIGN16 Put\_T\_MVM(const T\_UNIT SLOW\*me, TUSIGN16 objectIndex, TINT16 attributeIndex, void FAST\* ptrValue) | |
| Test function name | void Put\_T\_MVM\_Test(void) | |
| Test case 1 | Test procedure | 1. Set the test input   TUSIGN16 result = FATAL\_ERROR;   1. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_HARDWARE|INIT\_TASKS); 2. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_DATA|INIT\_CALCULATION); 3. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, (MVMEASUREMENT\_ME->maxIdx + 0x01), (TINT16)WHOLE\_OBJECT, (void FAST\*)&Qn); |
| Expected result | result == ILLEGAL\_OBJ\_IDX |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the test input   TUSIGN16 result = FATAL\_ERROR; mVMeasurementDynamicDuplicated.Qn = 80.0f;   1. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_HARDWARE|INIT\_TASKS); 2. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_DATA|INIT\_CALCULATION); 3. Call   result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_Qn, (TINT16)WHOLE\_OBJECT, (void FAST\*)&Qn) ; |
| Expected result | result == OK  mVMeasurementDynamicDuplicated.Qn == 50.0f |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the test input   TUSIGN16 result = FATAL\_ERROR; mVMeasurementProtected.protectedDataStatus = MVM\_PROTECTED;  TUSIGN8 QnSimEnable = MVM\_ENABLE;   1. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_HARDWARE|INIT\_TASKS); 2. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_DATA|INIT\_CALCULATION); 3. Call   result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_QnSimEnable, (TINT16)WHOLE\_OBJECT, (void FAST\*)&QnSimEnable); |
| Expected result | result == OK  mVMeasurementProtected.protectedDataStatus == MVM\_CALCULATE |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the test input   TUSIGN16 result = FATAL\_ERROR;  TUSIGN8 gasConfFlag = MVM\_GONFIGING;   1. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_HARDWARE|INIT\_TASKS); 2. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_DATA|INIT\_CALCULATION); 3. Call   result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_gasConfFlag, (TINT16)WHOLE\_OBJECT, (void FAST\*)&gasConfFlag); |
| Expected result | result == OK  mVMeasurementStaticRare.gasConfFlag == MVM\_GONFIGING |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the test input   TUSIGN16 result = FATAL\_ERROR;  TUSIGN8 mvmAlarmSimulation = 0x01;   1. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_HARDWARE|INIT\_TASKS); 2. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_DATA|INIT\_CALCULATION); 3. Call   result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_mvmAlarmSimulation, (TINT16)0x01, (void FAST\*)&mvmAlarmSimulation); |
| Expected result | result == OK  mVMeasurementDynamicDuplicated.mvmAlarmSimulation == 0X02 |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the test input   TUSIGN16 result = FATAL\_ERROR;  TUSIGN8 gasRef = MVM\_DEG0;.   1. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_HARDWARE|INIT\_TASKS); 2. Call mVMeasurement.unit.Initialize(MVMEASUREMENT\_ME,INIT\_DATA|INIT\_CALCULATION); 3. Call   result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_gasRef, (TINT16)WHOLE\_OBJECT, (void FAST\*)&gasRef); |
| Expected result | result == OK  mVMeasurementStaticFrequentStaticDefault.gasRef == MVM\_DEG0 |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set the Precondition parameter as following   assertTestOK=0   1. result = Put\_T\_MVM((NULL), MVM\_IDX\_protectedDataStatus, (TINT16)WHOLE\_OBJECT, (void FAST\*)&dataStatus); |
| Expected result | assertTestOK== 1 |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set the Precondition parameter as following   assertTestOK=0   1. result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_protectedDataStatus, (TINT16)WHOLE\_OBJECT, (void FAST\*)NULL); |
| Expected result | assertTestOK== 1 |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set the Precondition parameter as following   assertTestOK=0  ptrState = MVMEASUREMENT\_ME->ptrState;  \*ptrState = NOT\_INITIALIZED;  mVMeasurementDynamicDuplicated.Qn = 80.0f;   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_Qn, (TINT16)WHOLE\_OBJECT, (void FAST\*)&Qn) ; |
| Expected result | assertTestOK== 1 |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set the Precondition parameter as following   MVMEASUREMENT\_ME->ptrState = INITIALIZED;  coordinatorReplaceStaticRare.operatingMode=CDT\_LIQUID\_MASS;  sVMeasurementReplaceStaticRareCalibration.KSET=SVM\_KFACTOR\_GAS;  steamtype = MVM\_STEAMTYPE\_WATER;   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
| Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_LIQUID |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set the Precondition parameter as following   MVMEASUREMENT\_ME->ptrState = INITIALIZED;  coordinatorReplaceStaticRare.operatingMode = CDT\_STEAM\_MASS;  sVMeasurementReplaceStaticRareCalibration.KSET = SVM\_KFACTOR\_LIQUID;  steamtype = MVM\_STEAMTYPE\_SATURATED;   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
| Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_GAS |
| Test result | Pass |
| Test case 12 | Test procedure | 1. Set the Precondition parameter as following   MVMEASUREMENT\_ME->ptrState = INITIALIZED;  coordinatorReplaceStaticRare.operatingMode = CDT\_STEAM\_MASS;  sVMeasurementReplaceStaticRareCalibration.KSET = SVM\_KFACTOR\_ GAS;  steamtype = MVM\_STEAMTYPE\_WATER;   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
| Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_ LIQUID |
| Test result | Pass |
| Test case 13 | Test procedure | Test "if((operationMode <= CDT\_LIQUID\_POWER) || ((operationMode >= CDT\_STEAM\_ACTUAL\_VOLUME) && (steamType == MVM\_STEAMTYPE\_WATER)))" with (FALSE) || ((FALSE) && (FALSE))   1. Set the Precondition parameter as following   Operation mode = CDT\_GAS\_ACTUAL\_VOLUME;  Steam type = MVM\_STEAMTYPE\_SATURATED   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_GAS |
|  | Test result | Pass |
| Test case 14 | Test procedure | Test "if((operationMode <= CDT\_LIQUID\_POWER) || ((operationMode >= CDT\_STEAM\_ACTUAL\_VOLUME) && (steamType == MVM\_STEAMTYPE\_WATER)))" with *(FALSE) || ((FALSE) && (TRUE)*   1. Set the Precondition parameter as following   Operation mode = CDT\_GAS\_ACTUAL\_VOLUME;  Steam type = MVM\_STEAMTYPE\_WATER   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_GAS |
|  | Test result | Pass |
| Test case 15 | Test procedure | Test "if((operationMode <= CDT\_LIQUID\_POWER) || ((operationMode >= CDT\_STEAM\_ACTUAL\_VOLUME) && (steamType == MVM\_STEAMTYPE\_WATER)))" with *(FALSE) || ((TRUE) && (FALSE))*   1. Set the Precondition parameter as following   Operation mode = CDT\_GAS\_ACTUAL\_VOLUME;  Steam type = MVM\_STEAMTYPE\_SATURATED   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_GAS |
|  | Test result | Pass |
| Test case 16 | Test procedure | Test "if((operationMode <= CDT\_LIQUID\_POWER) || ((operationMode >= CDT\_STEAM\_ACTUAL\_VOLUME) && (steamType == MVM\_STEAMTYPE\_WATER)))" with *(FALSE) || ((TRUE) && (TRUE))*   1. Set the Precondition parameter as following   Operation mode = CDT\_GAS\_ACTUAL\_VOLUME;  Steam type = MVM\_STEAMTYPE\_WATER   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_LIQUID |
|  | Test result | Pass |
| Test case 17 | Test procedure | Test "if((operationMode <= CDT\_LIQUID\_POWER) || ((operationMode >= CDT\_STEAM\_ACTUAL\_VOLUME) && (steamType == MVM\_STEAMTYPE\_WATER)))" with *(TRUE) || ((FALSE) && (FALSE)) for water*   1. Set the Precondition parameter as following   Operation mode = CDT\_LIQUID\_POWER;  Steam type = MVM\_STEAMTYPE\_SATURATED   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_LIQUID |
|  | Test result | Pass |
| Test case 18 | Test procedure | Test "if((operationMode <= CDT\_LIQUID\_POWER) || ((operationMode >= CDT\_STEAM\_ACTUAL\_VOLUME) && (steamType == MVM\_STEAMTYPE\_WATER)))" with *(TRUE) || ((FALSE) && (TRUE)) with water*   1. Set the Precondition parameter as following   Operation mode = CDT\_LIQUID\_POWER;  Steam type = MVM\_STEAMTYPE\_WATER   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | sVMeasurementReplaceStaticRareCalibration.KSET is SVM\_KFACTOR\_LIQUID |
|  | Test result | Pass |
| Test case 19 | Test procedure | "if (result == (TUSIGN16)OK)" at line 169 with *False*   1. Set the Precondition parameter as following   gs\_put\_t\_unit\_inControl = eTRUE;  gs\_put\_t\_unit\_retVal = ERROR\_CODES;   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | 1. sVMeasurementReplaceStaticRareCalibration.KSET shall remain unchanged. 2. The function shall return ERROR\_CODES. |
|  | Test result | Pass |
| Test case 20 | Test procedure | "if (KSetOld != KSet)" at line 195 with *False*   1. Set the Precondition parameter as following   Set steam type to MVM\_STEAMTYPE\_SATURATED;   1. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; 2. Record the kset 3. Call result = Put\_T\_MVM(MVMEASUREMENT\_ME, MVM\_IDX\_steamType, (TINT16)WHOLE\_OBJECT, (void FAST\*)& steamtype) ; |
|  | Expected result | 1. sVMeasurementReplaceStaticRareCalibration.KSET shall remain unchanged. 2. The function shall return OK. |
|  | Test result | Pass |

#### Module QCommonCal.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TBOOL IsPVMatchWithQ(TUSIGN8 cal\_Q\_type) | |
| Test function name | static void IsPVMatchWithQ\_TEST(void) | |
| Test case 1 | Test procedure | 1. Set the parameter as following   lInputQ = CDT\_LIQUID\_ACTUAL\_VOLUME  lPvAssignment = MAPPER\_DEVICE\_VAR\_QV;   1. Call function to set PV:   (void)mapper\_Put((TUSIGN16)MAPPER\_IDX\_PV\_Assignment, (TINT16)WHOLE\_OBJECT, &lPvAssignment);   1. Call the tested function:   lMatch = IsPVMatchWithQ(lInputQ); |
| Expected result | lMatch shall be eTRUE |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the parameter as following   lInputQ = CDT\_LIQUID\_ACTUAL\_VOLUME  lPvAssignment = MAPPER\_DEVICE\_VAR\_QN;   1. Call function to set PV:   (void)mapper\_Put((TUSIGN16)MAPPER\_IDX\_PV\_Assignment, (TINT16)WHOLE\_OBJECT, &lPvAssignment);   1. Call the tested function: 2. lMatch = IsPVMatchWithQ(lInputQ); |
| Expected result | lMatch shall be eFALSE |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the parameter as following   lInputQ = NOT\_VALID\_Q\_TYPE;   1. Call the tested function:   lMatch = IsPVMatchWithQ(lInputQ); |
| Expected result | lMatch shall be eFALSE |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | static TBOOL IsSupportedQCalculation(TUSIGN8 cal\_Q\_type, TUSIGN8 operationMode) | |
| Test function name | static void IsSupportedQCalculation\_TEST (void) | |
| Test case 1 | Test procedure | 1. Set the parameter as following   lInputQ = CDT\_LIQUID\_ACTUAL\_VOLUME  lInputOp = CDT\_LIQUID\_ACTUAL\_VOLUME;;   1. Call the tested function:   lSupport = IsSupportedQCalculation(lInputQ, lInputOp); |
| Expected result | lMatch shall be eTRUE |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the parameter as following   lInputQ = CDT\_LIQUID\_POWER  lInputOp = CDT\_LIQUID\_ACTUAL\_VOLUME;;   1. Call the tested function:   lSupport = IsSupportedQCalculation(lInputQ, lInputOp); |
| Expected result | lMatch shall be eFALSE |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TBOOL CalculateQ(TUSIGN8 cal\_Q\_type, TFLOAT\* pBaseValue, TFLOAT\* pDampValue, TBOOL simulation, TFLOAT hiLimit, TFLOAT cutoffPercentage, TFLOAT dampingTime) | |
| Test function name | static void CalculateQ\_TEST(void) | |
| Test case 1 | Test procedure | 1. Set the parameter as following   set input Q is not supported in the current op mode;   1. Call the tested function: |
| Expected result | Cut off shall be eFALSE |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the parameter as following   set input Q is supported in the current op mode, but in simulation   1. Call the tested function: |
| Expected result | Cut off shall be eFALSE |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set the parameter as following   set input Q is supported in the current op mode, not in simulation, match with PV, cut off happen   1. Call the tested function: |
| Expected result | Cut off shall be eTRUE;  Base and Damped Values shall be 0 |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set the parameter as following   set input Q is supported in the current op mode, not in simulation, not match with PV   1. Call the tested function: |
| Expected result | Cut off shall be eFALSE; |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set the parameter as following   set input Q is supported in the current op mode, not in simulation, match with PV, back from cut off state   1. Call the tested function: |
| Expected result | Cut off shall be eFALSE;  Damped value shall be cut off value |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set the parameter as following   set input Q is supported in the current op mode, not in simulation, match with PV, last time cut off, still in cut off state   1. Call the tested function: |
| Expected result | Cut off shall be still eTRUE;  Base and Damped Values shall be 0 |
| Test result | Pass |

### Module Test Coverage

**Module "LiquidQm" coverage:**

Function "CalculateLiquidQm" coverage: 100.00%

**Module "LiquidQn" coverage:**

Function "CalculateLiquidQn" coverage: 100.00%

**Module "LiquidQp" coverage:**

Function "CalculateLiquidQp" coverage: 100.00%

**Module "gasQm" coverage:**

Function "CalculateGasQm" coverage: 100.00%

**Module "gasQn" coverage:**

Function "CalculateGasQn" coverage: 100.00%

**Module "gasQp" coverage:**

Function "CalculateGasQp" coverage: 100.00%

**Module "GasQvPartial" coverage:**

Function "CalculateGasQvPartial" coverage: 100.00%

**Module "GasQnPartial" coverage:**

Function "CalculateGasQnPartial" coverage: 100.00%

**Module "IAPWS\_IF97" coverage:**

Function "diter3\_" coverage: 100.00%

Function "fb23\_" coverage: 100.00%

Function "fwnpt3\_1" coverage: 100.00%

Function "fwnpt3\_2" coverage: 100.00%

Function "hpt1n\_" coverage: 100.00%

Function "hpt2n\_" coverage: 100.00%

Function "hvt3n\_" coverage: 100.00%

Function "nullp3n\_" coverage: 100.00%

Function "pbt\_" coverage: 100.00%

Function "psattn\_" coverage: 100.00%

Function "pvt3n\_" coverage: 100.00%

Function "reg3s\_" coverage: 100.00%

Function "regsopt\_" coverage: 100.00%

Function "tsatpn\_" coverage: 100.00%

Function "vest3\_" coverage: 100.00%

Function "vpt1n\_" coverage: 100.00%

Function "vpt2n\_" coverage: 100.00%

Function "vpt3n\_" coverage: 100.00%

Function "wnpt3\_" coverage: 100.00%

**Module "splint\_SplineCubicInterpolation" coverage:**

Function "spline1d" coverage: 100.00%

Function "spline2d" coverage: 100.00%

Function "splineInterpolate1d" coverage: 100.00%

Function "splineInterpolate2d" coverage: 100.00%

**Module "SteamQm" coverage:**

Function "CalculateSteamQm" coverage: 100.00%

**Module "SteamQp" coverage:**

Function "CalculateSteamQp" coverage: 100.00%

**Module "MVMeasurement\_execute" coverage:**

Function "CalcCompressFactor\_MVM" coverage: 100.00%

Function "CalcCompressFactor\_MVM\_Test" coverage: 100.00%

Function "CalcSteamDensityEXE\_MVM" coverage: 100.00%

Function "CalcSteamHEXE\_MVM" coverage: 100.00%

Function "CalcSteamHPreset" coverage: 100.00%

Function "CalculateGasQmEXE\_MVM" coverage: 100.00%

Function "CalculateGasQnEXE\_MVM" coverage: 100.00%

Function "CalculateGasQnPartialEXE\_MVM" coverage: 100.00%

Function "CalculateGasQpEXE\_MVM" coverage: 100.00%

Function "CalculateGasQvPartialEXE\_MVM" coverage: 100.00%

Function "CalculateLiquidQmEXE\_MVM" coverage: 100.00%

Function "CalculateLiquidQnEXE\_MVM" coverage: 100.00%

Function "CalculateLiquidQpEXE\_MVM" coverage: 100.00%

Function "CalculateSteamQmEXE\_MVM" coverage: 100.00%

Function "CalculateSteamQpEXE\_MVM" coverage: 100.00%

Function "CheckDataCrcSRV\_MVM" coverage: 100.00%

Function "GetQmPercentageSRV\_MVM" coverage: 100.00%

Function "GetQnPartialPercentageSRV\_MVM" coverage: 100.00%

Function "GetQnPercentageSRV\_MVM" coverage: 100.00%

Function "GetQpPercentageSRV\_MVM" coverage: 100.00%

Function "GetQvPartialPercentageSRV\_MVM" coverage: 100.00%

Function "UpdateBiagasQvMaxDN\_MVM" coverage: 100.00%

Function "UpdateDiagnosisEXE\_MVM" coverage: 100.00%

Function "UpdateGasPowerMaxDN\_MVM" coverage: 100.00%

Function "UpdateGasQnMaxDN\_MVM" coverage: 100.00%

Function "UpdateLiquidPowerMaxDN\_MVM" coverage: 100.00%

Function "UpdateLiquidQnMaxDN\_MVM" coverage: 100.00%

Function "UpdateQAlarmEXE\_MVM" coverage: 100.00%

Function "UpdateQmMaxDN\_MVM" coverage: 100.00%

Function "UpdateSteamPowerMaxDN\_MVM" coverage: 100.00%

**Module "QCommonCal" coverage:**

Function "CalculateQ" coverage: 100.00%

**Module "MVMeasurement\_overload" coverage:**

Function "Initialize\_MVM" coverage: 100.00%

Function "Put\_T\_MVM" coverage: 92.45%

Steppoint(s) not covered in function (file C:\ProjectM\VT5\SMVM\_test\MVMeasurement\Source\MVMeasurement\_overload.c)

Code Analysis:

Code not test in Put\_T\_MVM.

assert(me->ptrObjectList);

As the ptrObjectList is a constant static code. So the Pointer cannot be NULL.

### Module Test Results

*<< Logging of terminal I/O resumes >>*

*VIP\_ASSERT in File=cb\_application\SMVMTest\T\_Unit\Source\T\_UNIT\_PUBLIC.c Line=582*

*Module test of "LiquidQn"*

*file: MVMeasurement\Source\..\Test\Test\_LiquidQn.c, compilation date: Jun 3 2021, 20:20:19*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=75 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=76 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=144 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=145 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=146 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=147 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=215 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=216 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=217 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQn.c Line=218 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 12 of 12 Tests-OK*

*Module test PASSED\nnModule test of "Gas Qm"*

*file: MVMeasurement\Source\..\Test\Test\_gasQm.c, compilation date: Jun 3 2021, 20:20:28*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=75 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=76 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=141 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=142 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=143 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=144 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=210 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=211 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=212 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=213 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=279 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=280 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=281 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=282 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=348 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=349 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=350 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=351 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=418 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQm.c Line=419 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 22 of 22 Tests-OK*

*Module test PASSED\nnModule test of "Liquid Qp"*

*file: MVMeasurement\Source\..\Test\Test\_LiquidQp.c, compilation date: Jun 3 2021, 20:20:20*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=72 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=75 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=145 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=146 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=147 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=148 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=217 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=218 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=219 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=220 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=289 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=290 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=291 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQp.c Line=292 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 16 of 16 Tests-OK*

*Module test PASSED\nnModule test of "Liquid Qm"*

*file: MVMeasurement\Source\..\Test\Test\_LiquidQm.c, compilation date: Jun 3 2021, 20:20:19*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=75 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=76 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=141 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=142 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=143 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=144 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=210 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=211 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=212 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=213 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=279 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=280 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=281 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=282 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=348 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=349 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=350 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=351 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=413 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_LiquidQm.c Line=414 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 22 of 22 Tests-OK*

*Module test PASSED\nnModule test of "gasQn"*

*file: MVMeasurement\Source\..\Test\Test\_gasQn.c, compilation date: Jun 3 2021, 20:20:28*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=77 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=78 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=79 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=80 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=152 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=153 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=154 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=155 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=230 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=231 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=232 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=233 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=308 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=309 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=310 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQn.c Line=311 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 16 of 16 Tests-OK*

*Module test PASSED\nnModule test of "Gas Qp"*

*file: MVMeasurement\Source\..\Test\Test\_gasQp.c, compilation date: Jun 3 2021, 20:20:28*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=72 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=75 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=140 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=141 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=142 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=143 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=208 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=209 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=210 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=211 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=275 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=276 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=277 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_gasQp.c Line=278 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 16 of 16 Tests-OK*

*Module test PASSED\nnModule test of "GasQvPartial"*

*file: MVMeasurement\Source\..\Test\Test\_GasQvPartial.c, compilation date: Jun 3 2021, 20:20:18*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=71 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=72 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=138 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=139 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=140 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=141 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=205 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=206 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=207 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQvPartial.c Line=208 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 12 of 12 Tests-OK*

*Module test PASSED\nnModule test of "GasQnPartial"*

*file: MVMeasurement\Source\..\Test\Test\_GasQnPartial.c, compilation date: Jun 3 2021, 20:20:18*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=71 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=72 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=74 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=138 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=139 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=140 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=141 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=205 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=206 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=207 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_GasQnPartial.c Line=208 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 12 of 12 Tests-OK*

*Module test PASSED\nnModule test of "IAPWS\_IF97.c"*

*file: MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c, compilation date: Jun 3 2021, 20:20:18*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=31 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=57 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=82 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=132 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=142 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=152 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=162 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=172 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=182 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=193 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=203 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=213 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=225 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=237 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=248 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=109 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=274 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=285 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=297 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=309 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=321 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=333 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=345 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=357 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=370 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=383 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=395 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=408 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=572 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=583 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=593 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=603 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=624 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=433 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=439 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=445 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=452 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=458 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=464 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=470 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=476 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=482 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=508 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_IAPWS\_IF97.c Line=514 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 44 of 44 Tests-OK*

*Module test PASSED\nnModule test of "SteamQm"*

*file: MVMeasurement\Source\..\Test\Test\_SteamQm.c, compilation date: Jun 3 2021, 20:20:22*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=71 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=72 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=139 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=140 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=207 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=208 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=275 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=276 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=343 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=344 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=413 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQm.c Line=414 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 12 of 12 Tests-OK*

*Module test PASSED\nnModule test of "SteamQp"*

*file: MVMeasurement\Source\..\Test\Test\_SteamQp.c, compilation date: Jun 3 2021, 20:20:23*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=72 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=73 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=138 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=139 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=204 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=205 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=269 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_SteamQp.c Line=270 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 8 of 8 Tests-OK*

*Module test PASSED\nnModule test of "MVMeasurement\_execute"*

*file: MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c, compilation date: Jun 3 2021, 20:26:40*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=71 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=90 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=108 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=126 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=144 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=163 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=181 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=199 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=217 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=235 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=253 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=271 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=289 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=307 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=330 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=352 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=374 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=396 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=418 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=439 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=460 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=483 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=505 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=526 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=549 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=568 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=608 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=615 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=636 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=642 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=665 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=672 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=692 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=699 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=723 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=730 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=754 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=760 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=787 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=794 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=816 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=822 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=845 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=851 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=874 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=880 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=903 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=909 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=933 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=939 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=962 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=969 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=993 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1000 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1024 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1031 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1058 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1065 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1092 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1099 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1122 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1128 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1509 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1516 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1545 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1552 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1580 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1587 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1616 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1623 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1653 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1660 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1689 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1696 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1725 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1732 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1761 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1768 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1797 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1804 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1833 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1840 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1869 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1876 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1905 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1912 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1941 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1948 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1977 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1984 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2013 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2020 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2050 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2057 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2080 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2087 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2110 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2117 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2140 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2147 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2168 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2175 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2196 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2203 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2231 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2238 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2395 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2405 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2493 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2591 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2686 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2748 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2811 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2898 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2853 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2939 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=2996 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3035 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3084 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3130 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3176 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3197 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3218 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3239 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3260 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3281 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3302 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3310 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3316 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1435 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1441 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1435 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1441 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1435 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1441 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1435 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1441 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1435 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1441 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1460 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1461 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1460 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1461 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1460 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1461 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1460 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1461 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1460 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1461 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1250 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1200 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1212 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1220 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1157 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1282 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1318 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1356 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=1412 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3347 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3365 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3381 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3399 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3415 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_execute.c Line=3433 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 164 of 164 Tests-OK*

*Module test PASSED\nnModule test of "QCommonCal"*

*file: MVMeasurement\Source\..\Test\Test\_QCommonCal.c, compilation date: Jun 3 2021, 20:20:22*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=24 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=33 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=39 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=62 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=71 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=111 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=130 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=131 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=132 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=152 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=153 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=154 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=155 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=175 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=176 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=177 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=197 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=198 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=199 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=219 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=220 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=221 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_QCommonCal.c Line=222 Test OK*

*0 of 0 VIP-Tests OK; 0 of 0 ASSERT-Tests OK; 23 of 23 Tests-OK*

*Module test PASSED\nnModule test of "SVMeasurement\_overload.c"*

*file: MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c, compilation date: Jun 3 2021, 20:26:41*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=28 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=31 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=32 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=34 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=35 Test OK*

*ASSERT call from line=38 assert in File=cb\_application\SMVMTest\MVMeasurement\Source\MVMeasurement\_overload.c Line=81*

*TEST OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=71 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=78 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=79 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=85 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=86 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=91 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=92 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=97 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=98 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=103 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=104 Test OK*

*ASSERT call from line=107 VIP\_ASSERT in File=cb\_application\SMVMTest\MVMeasurement\Source\MVMeasurement\_overload.c Line=143*

*TEST OK*

*ASSERT call from line=112 VIP\_ASSERT in File=cb\_application\SMVMTest\MVMeasurement\Source\MVMeasurement\_overload.c Line=144*

*TEST OK*

*ASSERT call from line=120 VIP\_ASSERT in File=cb\_application\SMVMTest\MVMeasurement\Source\MVMeasurement\_overload.c Line=145*

*TEST OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=138 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=153 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=168 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=182 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=194 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=206 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=218 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=230 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=242 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=257 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=258 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=272 Test OK*

*Test in File=MVMeasurement\Source\..\Test\Test\_MVMeasurement\_overload.c Line=273 Test OK*

*3 of 3 VIP-Tests OK; 1 of 1 ASSERT-Tests OK; 29 of 29 Tests-OK*

*Module test PASSED\nn*

*<< Debugger has been reset >>*

## Function Test

### Function Test mapping table

The table below describes the Mapping between test case and requirements from SSRS

Table 1: Mapping between Test Case and Requirements from SSRS

|  |  |  |
| --- | --- | --- |
| **Requirements from SSRS** | **Test case** | **Description** |
| **Non-Functional** | | |
| NFR3.1.1 | CodeReview | C or Assemble Language shall be used. |
| NFR3.1.2 | Review | Check IAR Tool General Device whether device is R5F363AM |
| NFR3.1.3 | Review | Check IAR Tool Compiler whether Optimization is None |
| NFR3.1.4 | Code Review | Check if Dynamic memory allocation is used or not. |
| NFR3.1.5 | Code Review | Check if subsystem is based on the common ABB framework concept |
| NFR3.1.6 | Procedure | Check if development activity is followed procedure. |
| NFR3.2.1 | Code Review | Refer Code review result. |
| NFR3.2.2 | PC-lint | Refer to Static Check of section 2.1 |
| NFR3.2.3 | Module-Test | Refer to Static Check of section 2.2 |
| NFR3.2.4 | Integration test |  |
|  |  |  |
|  |  |  |
| **Functional** | | |
| FR 4.1 | TC01 | Measure standard volume flow for liquid |
| FR 4.2 | TC02  TC03  TC04  TC05 | Measure Mass flow for Liquid |
| FR 4.3 | TC06 | Measure energy flow for Liquid |
| FR 4.4 | TC07  TC08  TC09  TC10 | Measure Standard Volume flow for gas |
| FR 4.5 | TC11  TC12 | Measure mass flow of gas |
| FR 4.6 | TC13 | Measure power flow of gas |
| FR 4.7 | TC14 | Measure biogas actual volume flow for methane |
| FR 4.8 | TC15 | Measure biogas standard volume rate for methane |
| FR 4.9 | TC16 | Measure Steam Mass flow |
| FR 4.10 | TC17 | Measure Steam Power Flow |
| FR 4.11 | TC02  TC03  TC04  TC05  TC12  TC16  TC17 | Check limits for flow rate |
| FR 4.12 | See Validation Test | Change flow Units. |
| FR 4.13 | TC01  TC07  TC08  TC09  TC10 | Compute percentage of QMax for Standard Volume flow |
| FR 4.14 | TC02  TC03  TC04  TC05  TC11  TC12  TC16 | Compute percentage of QMax for mass flow |
| FR 4.15 | TC06  TC17 | Compute percentage of QMax for energy flow |
| FR 4.16 | See Validation Test | Setting damping time |
| FR 4.17 | TC16 | Supervisor Steam Status |
|  |  |  |
|  |  |  |

### Function Test Cases

#### Test Case 01

| ID: TC-01 | | Req.-ID:  FR 4.1  FR 4.13 | | Tester: Zuochen Wang | Test date: | | **2015-11-02** | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: This case will test the calculation of Standard Volume flow and compute the percentage of QMax. | | | | | | | |
| Requirement Details | FR 4.1 ：Measure standard volume flow for liquid  FR 4.13：Compute percentage of QMax for Standard Volume flow | | | | | | | |
| Pass Criteria | Calculate standard volume flow and percentage flow correctly. | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench .  Data:  QnRange  Ref Temperature  Temperature  Volumeexpansion  VolumeFlow: | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | QnRange : [2.10f,90.00f];  Ref Temperature:21.35 ℃ ;  Temperature :31.25 ℃;  Volumeexpansion :0.036;  VolumeFlow:87.580 m3/h; | | Call CalculateLiquidQn  to calculate the Normal volume flow and percentage flow. | | | Normal Volume flow:64.567974 m3/h  Percentage flow:  71.74% | | Y |
|  | ……. | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 02

| ID: TC-02 | | Req.-ID:  FR 4.2  FR 4.11  FR 4.14 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: This case will test the calculation of mass flow and percentage of QMax. Check imits for flow rate and will raise alarm when flow lower than Qmin | | | | | | | |
| Requirement Details | FR 4.2 ：Measure mass flow for liquid  FR 4.11：Check limits for flow rate  FR 4.14：Compute percentage of QMax for mass flow | | | | | | | |
| Pass Criteria | Calculate Mass flow Correctly  Calculate Percentage correctly  Raines alarm when alarm condition is met. | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  SimulationMode  QmSimEnable  LiquidCorrection  QmRange  QmSim  QmPerSim  Tempereture Ref  Tempereture Real  DensityRef  DensityReal  VolumeExpandBeta1  densityExpandBeta2  QmAlarmLimits  VolumeFlow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. SimulationMode: CDT\_SIMULATION\_CB\_OFF; 2. QmSimEnable: MVM\_DISABLE; 3. LiquidCorrection: MVM\_WITHOUT\_CORRECTION;      1. QmRange:   hiLim :79800 kg/h  loLim :100.0 kg/h  span :79700.0 kg/h   1. QmSim:   78800 kg/h;   1. QmPerSim:   70.00%;   1. Tempereture Ref 21.35 ℃. 2. Tempereture Real 31 ℃. 3. DensityRef 890.88 kg/m2 4. DensityReal 998.89 kg/m3 5. VolumeExpandBeta1 0.036 6. densityExpandBeta2 0.018 7. VolumeFlow:   20.00 m3/h | | Call CalculateLiquidQMEXE of MVM to calculate the mass volume flow and percentage flow. | | | mass flow:  19977.8 Kg/h  Percentage flow:  25.035 % | | Y |
| 2 | 1. QmAlarmLimits   hiLim :90.0 (%)  loLim :35.0 (%) | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | Alarm State:   * MVM\_ALM\_FLOWRATE\_REACH\_QMIN shall be Set | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 03

| ID: TC-03 | | Req.-ID:  FR 4.2  FR 4.11  FR 4.14 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: This case tests the calculation of mass flow and percentage of QMax. When flow is in the range of QmRange. | | | | | | | |
| Requirement Details | FR 4.2 ：Measure mass flow for liquid  FR 4.11：Check limits for flow rate.  FR 4.14：Compute percentage of QMax for mass flow | | | | | | | |
| Pass Criteria | Calculate Mass flow Correctly  Calculate Percentage correctly  Raines alarm when alarm condition is met. | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  SimulationMode  QmSimEnable  LiquidCorrection  QmRange  QmSim  QmPerSim  Tempereture Ref  Tempereture Real  DensityRef  DensityReal  VolumeExpandBeta1  densityExpandBeta2  QmAlarmLimits  VolumeFlow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. SimulationMode: CDT\_SIMULATION\_CB\_OFF; 2. QmSimEnable: MVM\_DISABLE; 3. LiquidCorrection: MVM\_DENSITY\_CORRECTION;      1. QmRange:   hiLim :79800 kg/h  loLim :100.0 kg/h  span :79700.0 kg/h   1. QmSim:   78800 kg/h;   1. QmPerSim:   70.00%;   1. Tempereture Ref 21.35 ℃. 2. Tempereture Real 31.25 ℃ 3. DensityRef 890.88 kg/m2 4. DensityReal 998.89 kg/m3 5. VolumeExpandBeta1 0.036 6. densityExpandBeta2 0.018 7. VolumeFlow:   65.32 m3/h | | Call CalculateLiquidQM of MVM to calculate the mass volume flow and percentage flow. | | | mass flow:  68562.146 Kg/h  Percentage flow:  85.917 % | | Y |
| 2 | 1. QmAlarmLimits   hiLim :90.0 (%)  loLim :35.0 (%) | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | Alarm State:  No Alarm be Set | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 04

| ID: TC-04 | | Req.-ID:  FR 4.2  FR 4.11  FR 4.14 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: This case tests the calculation of mass flow and percentage of QMax. When flow is greater than Qmax but lower than 103% QMax | | | | | | | |
| Requirement Details | FR 4.2 ：Measure mass flow for liquid  FR 4.11：Check limits for flow rate.  FR 4.14：Compute percentage of QMax for mass flow | | | | | | | |
| Pass Criteria | Calculate Mass flow Correctly  Calculate Percentage correctly  Raines alarm when alarm condition is met. | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  SimulationMode  QmSimEnable  LiquidCorrection  QmRange  QmSim  QmPerSim  Tempereture Ref  Tempereture Real  DensityRef  DensityReal  VolumeExpandBeta1  densityExpandBeta2  QmAlarmLimits  VolumeFlow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. SimulationMode: CDT\_SIMULATION\_CB\_OFF; 2. QmSimEnable: MVM\_DISABLE; 3. LiquidCorrection: MVM\_DENSITY\_CORRECTION;      1. QmRange:   hiLim :79800 kg/h  loLim :100.0 kg/h  span :79700.0 kg/h   1. QmSim:   78800 kg/h;   1. QmPerSim:   70.00%;   1. Tempereture Ref 21.35 ℃. 2. Tempereture Real 31.25 ℃ 3. DensityRef 890.88 kg/m2 4. DensityReal 998.89 kg/m3 5. VolumeExpandBeta1 0.036 6. densityExpandBeta2 0.018 7. VolumeFlow:   72.32 m3/h | | Call CalculateLiquidQM of MVM to calculate the mass volume flow and percentage flow. | | | mass flow:  75909.56 Kg/h  Percentage flow:  95.123 % | | Y |
| 2 | 1. QmAlarmLimits   hiLim :90.0 (%)  loLim :25.0 (%) | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | Alarm State:   * MVM\_ALM\_FLOWRATE\_REACH\_QMAX shall be Set | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 05

| ID: TC-05 | | Req.-ID:  FR 4.2  FR 4.11  FR 4.14 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: This case tests the calculation of mass flow and percentage of QMax. When flow is greater than 103% QMax of Mass flow. | | | | | | | |
| Requirement Details | FR 4.2 ：Measure mass flow for liquid  FR 4.11：Check limits for flow rate.  FR 4.14：Compute percentage of QMax for mass flow | | | | | | | |
| Pass Criteria | Calculate Mass flow Correctly  Calculate Percentage correctly  Raines alarm when alarm condition is met. | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  SimulationMode  QmSimEnable  LiquidCorrection  QmRange  QmSim  QmPerSim  Tempereture Ref  Tempereture Real  DensityRef  DensityReal  VolumeExpandBeta1  densityExpandBeta2  QmAlarmLimits  VolumeFlow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. SimulationMode: CDT\_SIMULATION\_CB\_OFF; 2. QmSimEnable: MVM\_DISABLE; 3. LiquidCorrection: MVM\_DENSITY\_CORRECTION;      1. QmRange:   hiLim :79800 kg/h  loLim :100.0 kg/h  span :79700.0 kg/h   1. QmSim:   78800 kg/h;   1. QmPerSim:   70.00%;   1. Tempereture Ref 21.35 ℃. 2. Tempereture Real 31.25 ℃ 3. DensityRef 1080.88 kg/m2 4. DensityReal 998.89 kg/m3 5. VolumeExpandBeta1 0.016 6. densityExpandBeta2 0.018 7. VolumeFlow:   88.58 m3/h | | Call CalculateLiquidQM of MVM to calculate the mass volume flow and percentage flow. | | | mass flow:  82652.236 Kg/h  Percentage flow:  103.57 % | | Y |
| 2 | 1. QmAlarmLimits   hiLim :90.0 (%)  loLim :25.0 (%) | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | Alarm State:   * MVM\_ALM\_FLOWRATE\_REACH\_QMAX shall be Set * MVM\_ALM\_FLOWRATE\_GT\_103 shall be Set | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 06

| ID: TC-06 | | Req.-ID:  FR 4.3  FR 4.15 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: This case tests the calculation of Energy flow and percentage of upper range of energy flow when flow is in range of QpRange. | | | | | | | |
| Requirement Details | FR 3. ：Measure energy flow for liquid  FR 4.15：Compute percentage of QMax for energy flow | | | | | | | |
| Pass Criteria | Calculate energy flow Correctly  Calculate Percentage correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  QpSimEnable  QpRange.hiLim  QpRange.LoLim  QpRange.Span  QpMaxDN  QpSim  T ExtReal  T Real  heatCapacity  Mass Flow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   * 1. QpSimEnable   MVM\_DISABLE   * 1. QpRange   hiLim: 320000 KJ/h  LoLim: 100 KJ/h  Span: 319900 KJ/h   * 1. QpMaxDN   320000 KJ/h   * 1. QpSim   298493.99 KJ/h   * 1. T ExtReal   112.25 ℃   * 1. T Real   121.35 ℃   * 1. heatCapacity   4.200 KJ/(kg·K)   * 1. Mass Flow   8240.58 kg/h | | 1. Call CalculateLiquidQpEXE of MVM to calculate the energy flow and percentage flow. | | | Energy flow:  314954.97 KJ/h  Percentage flow:  98.42 % | | Y |
|  |  | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 07

| ID: TC-07 | | Req.-ID:  FR 4.4  FR 4.13 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests flow calculation for gas at UK standard status. | | | | | | | |
| Requirement Details | FR 4.4 ：Measure standard volume flow for gas  FR 4.13：Compute percentage of QMax for Standard Volume flow | | | | | | | |
| Pass Criteria | Calculate standard volume flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  temperaturePreset  dvPressurePreset  QvMaxDn  QnSimEnable  QnRange.hiLim  QnRange.loLim  QnRange.span  QnSim  T Real  P real  gasRef | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. temperaturePreset   35.0 ℃   1. dvPressurePreset   3201 KPa   1. QvMaxDn   1220 m3/h | | Call UpdateGasQnMaxDN of MVM to calculate the QnMaxDN. | | | QnMaxDN shall be:  36099.58 m3/h | | Y |
| 2 | 1. QnSimEnable   MVM\_DISABLE   1. QnRange   hiLim: 30000 m3/h  LoLim: 10 m3/h  Span: 29990 m3/h   1. QnSim   28862.00 m3/h   1. T Real   31.25 ℃   1. P Real   2535.23 KPa   1. volume flow   1127.58 m3/h   1. gasRef:   MVM\_UK | | Call CalculateGasQnEXE of MVM to calculate the flow and percentage flow. | | | flow:  26750.93 m3/h  Percentage flow:  89.17% | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 08

| ID: TC-08 | | Req.-ID:  FR 4.4  FR 4.13 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests flow calculation for gas at US standard status. | | | | | | | |
| Requirement Details | FR 4.4 ：Measure standard volume flow for gas  FR 4.13：Compute percentage of QMax for Standard Volume flow | | | | | | | |
| Pass Criteria | Calculate standard volume flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  temperaturePreset  dvPressurePreset  QvMaxDn  QnSimEnable  QnRange.hiLim  QnRange.loLim  QnRange.span  QnSim  T Real  P real  gasRef | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. temperaturePreset   35.0 ℃   1. dvPressurePreset   3201 KPa   1. QvMaxDn   1220 m3/h | | Call UpdateGasQnMaxDN of MVM to calculate the QnMaxDN. | | | QnMaxDN shall be:  36794.24 m3/h | | Y |
| 2 | 1. QnSimEnable   MVM\_DISABLE   1. QnRange   hiLim: 30000 m3/h  LoLim: 10 m3/h  Span: 29990 m3/h   1. QnSim   28862.00 m3/h   1. T Real   31.25 ℃   1. P Real   2535.23 KPa   1. volume flow   1127.58 m3/h   1. gasRef:   MVM\_US | | Call CalculateGasQnEXE of MVM to calculate the flow and percentage flow. | | | flow:  27265.70 m3/h  Percentage flow:  90.86% | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 09

| ID: TC-09 | | Req.-ID:  FR 4.4  FR 4.13 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests flow calculation for gas at (103.25 KPa,0.0℃) standard status. | | | | | | | |
| Requirement Details | FR 4.4 ：Measure standard volume flow for gas  FR 4.13：Compute percentage of QMax for Standard Volume flow | | | | | | | |
| Pass Criteria | Calculate standard volume flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  temperaturePreSet  dvPressurePreSet  QvMaxDn  QnSimEnable  QnRange.hiLim  QnRange.loLim  QnRange.span  QnSim  T Real  P real  gasRef | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. temperaturePreSet   35.0 ℃   1. dvPressurePreSet   3201 KPa   1. QvMaxDn   1220 m3/h | | Call UpdateGasQnMaxDN of MVM to calculate the QnMaxDN. | | | QnMaxDN shall be:  34163.94 m3/h | | Y |
| 2 | 1. QnSimEnable   MVM\_DISABLE   1. QnRange   hiLim: 30000 m3/h  LoLim: 10 m3/h  Span: 29990 m3/h   1. QnSim   28862.00 m3/h   1. T Real   31.25 ℃   1. P Real   2535.23 KPa   1. volume flow   1127.58 m3/h   1. gasRef:   MVM\_DEG0 | | Call CalculateGasQnEXE of MVM to calculate the flow and percentage flow. | | | flow:  25316.56 m3/h  Percentage flow:  84.39% | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 10

| ID: TC-10 | | Req.-ID:  FR 4.4  FR 4.13 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests flow calculation for gas at (103.25 KPa,20.0℃) standard status. | | | | | | | |
| Requirement Details | FR 4.4 ：Measure standard volume flow for gas  FR 4.13：Compute percentage of QMax for Standard Volume flow | | | | | | | |
| Pass Criteria | Calculate standard volume flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  temperaturePreSet  dvPressurePreSet  QvMaxDn  QnSimEnable  QnRange.hiLim  QnRange.loLim  QnRange.span  QnSim  T Real  P real  gasRef | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | Set the Parameters :   1. temperaturePreSet   35.0 ℃   1. dvPressurePreSet   3201 KPa   1. QvMaxDn   1220 m3/h | | Call UpdateGasQnMaxDN of MVM to calculate the QnMaxDN. | | | QnMaxDN shall be:  36665.42 m3/h | | Y |
| 2 | 1. QnSimEnable   MVM\_DISABLE   1. QnRange   hiLim: 30000 m3/h  LoLim: 10 m3/h  Span: 29990 m3/h   1. QnSim   28862.00 m3/h   1. T Real   31.25 ℃   1. P Real   2535.23 KPa   1. volume flow   1227.58 m3/h   1. gasRef:   MVM\_ DEG20 | | Call CalculateGasQnEXE of MVM to calculate the flow and percentage flow. | | | flow:  29579.84 m3/h  Percentage flow:  98.60% | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 11

| ID: TC-11 | | Req.-ID:  FR 4.5  FR 4.11  FR 4.14 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests mass flow calculation for gas with real density | | | | | | | |
| Requirement Details | FR 4.5 ：Measure mass flow for gas  FR 4.11：Check limits of flow  FR 4.14：Compute percentage of QMax for mass flow | | | | | | | |
| Pass Criteria | Calculate mass flow Correctly  Check limits of flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  Density PreSet  QvMaxDn  SimulationMode  QmSimEnable  GasDensitySelection  QmRange.hiLim  QmRange.loLim  QmRange.span  QmSim  QmPerSim  Density Ref  Density Real  VolumeFlow  Normal Volue flow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. Density PreSet   1.30 Kg/m3   1. QvMaxDn   1230.000 m3/h | | Call UpdateGasQmMaxDN of MVM to calculate the QmMaxDN. | | | QmMaxDN shall be:  1599.00 kg/h | | Y |
| 2 | 1. SimulationMode   CDT\_SIMULATION\_CB\_OFF   1. QmSimEnable   MVM\_DISABLE   1. GasDensitySelection   MVM\_ACTUAL\_DENSITY   1. QmRange   .hiLim 1400 kg/h  .loLim 10 kg/h  .span 1390 kg/h   1. QmSim   1300 kg/h   1. QmPerSim   72.00 %     1. Density Ref   1.293 kg/m3   1. Density Real   2.38 kg/m3   1. VolumeFlow   523.000 m3/h   1. Normal Volue flow   1128.580 m3/h | | Call CalculateGasQmEXE of MVM to calculate the flow and percentage flow. | | | The Mass flow shall be:  1244.74 kg/h | | Y |
| 3 | 1. QmAlarmLimits   hiLim :97.0 (%)  loLim :25.0 (%) | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | The percentage shall be: 88.91 %  No Alarm is set. | | Y |
|  | | | | | | | | |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 12

| ID: TC-12 | | Req.-ID:  FR 4.5  FR 4.11  FR 4.14 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests mass flow calculation for gas with refer density | | | | | | | |
| Requirement Details | FR 4.5 ：Measure mass flow for gas  FR 4.11：Check limits of flow  FR 4.14：Compute percentage of QMax for mass flow | | | | | | | |
| Pass Criteria | Calculate mass flow Correctly  Check limits of flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  Density Preset  QvMaxDn  SimulationMode  QmSimEnable  GasDensitySelection  QmRange.hiLim  QmRange.loLim  QmRange.span  QmSim  QmPerSim  Density Ref  Density Real  VolumeFlow  Normal Volue flow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. Density Preset   1.30 Kg/m3   1. QvMaxDn   1230.000 m3/h | | Call UpdateGasQmMaxDN of MVM to calculate the QmMaxDN. | | | QmMaxDN shall be:  1599.00 kg/h | | Y |
| 2 | 1. SimulationMode   CDT\_SIMULATION\_CB\_OFF   1. QmSimEnable   MVM\_DISABLE   1. GasDensitySelection   MVM\_ACTUAL\_DENSITY   1. QmRange   .hiLim 1400 kg/h  .loLim 10 kg/h  .span 1390 kg/h   1. QmSim   1300 kg/h   1. QmPerSim   72.00 %     1. Density Ref   1.293 kg/m3   1. Density Real   2.38 kg/m3   1. VolumeFlow   523.000 m3/h   1. Normal Volue flow   1128.580 m3/h | | Call CalculateGasQmEXE of MVM to calculate the flow and percentage flow. | | | The Mass flow shall be:  1459.254.kg/h | | Y |
| 3 | 1. QmAlarmLimits   hiLim :90.0 (%)  loLim :25.0 (%) | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | The percentage shall be:  104.23 %  Alarm State:   * MVM\_ALM\_FLOWRATE\_REACH\_QMAX shall be Set * MVM\_ALM\_FLOWRATE\_GT\_103 shall be Set | | Y |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 13

| ID: TC-13 | | Req.-ID:  FR 4.6  FR 4.15 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests energy flow calculation for gas | | | | | | | |
| Requirement Details | FR 4.6 ：Measure Energy flow for gas  FR 4.15：Compute percentage of QMax for energy flow | | | | | | | |
| Pass Criteria | Calculate mass flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  calorificEnergy  QnMaxDn  QpSimEnable  QpRange.hiLim  QpRange.LoLim  QpRange.Span  QpMaxDN  QpSim  VolumeFlow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. calorificEnergy   36.00 MJ/m3   1. QnMaxDn   78.000 m3/h | | Call UpdateGasQpMaxDN of MVM to calculate the QpMaxDN. | | | QpMaxDN shall be:  2808000.00 kJ/h | | Y |
| 2 | 1. QpSimEnable   MVM\_DISABLE   1. QpRange   hiLim: 2370000 KJ/h  LoLim: 2500 KJ/h  Span: 2367500 KJ/h     1. QpSim   1085500 KJ/h   1. VolumeFlow   57.580 m3/h | | Call CalculateGasQpEXE of MVM to calculate the flow and percentage flow. | | | The Energy flow shall be:  2072880.00.KJ/h  The percentage shall be:  87.46 % | | Y |
|  |  | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 14

| ID: TC-14 | | Req.-ID:  FR 4.7 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests Biogas volume flow calculation | | | | | | | |
| Requirement Details | FR 4.7 ：Measure biogas actual volume flow | | | | | | | |
| Pass Criteria | Calculate biogas volume flow Correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  biogasPerc  VolumeFlow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. biogasPerc   66.00 %   1. VolumeFlow   80.000 m3/h | | Call CalculateGasQvPartialEXE of MVM to calculate the biogas volume flow. | | | biogas volume flow shall be:  52.8.00 m3/h | | Y |
|  |  | |  | | |  | |  |
|  |  | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 15

| ID: TC-15 | | Req.-ID:  FR 4.8 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests Biogas standard flow calculation | | | | | | | |
| Requirement Details | FR 4.8 ：Measure biogas standard volume flow | | | | | | | |
| Pass Criteria | Calculate biogas standard volume flow Correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  biogasPerc  StandarVoluemflow | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. biogasPerc   78.00 %   1. StandarVoluemflow   800.00 m3/h | | Call CalculateGasQnPartialEXE of MVM to calculate the biogas volume flow. | | | biogas volume flow shall be:  624.00 m3/h | | Y |
|  |  | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 16

| ID: TC-16 | | Req.-ID:  FR 4.9  FR 4.11  FR 4.14  FR 4.17 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case test steam mass flow calculation | | | | | | | |
| Requirement Details | FR 4.9 ：Measure steam mass flow  FR 4.11 ：Check flow limits  FR 4.14 ：Compute Percentage of QMax for mass flow  FR 4.17 ：Supervisor Steam Status | | | | | | | |
| Pass Criteria | Calculate mass flow Correctly  check flow limits Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  Density Preset  QvMaxDn  T Real  P Real  Volume flow  QmRange  QmAlmRange | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. Density Preset   100 kg/m3   1. QvMaxDn   90.00 m3/h | | Call UpdateSteamQmMaxDN of MVM to calculate the QmMaxDN. | | | QmMaxDN shall be:  9000.00 kg/h | | Y |
| 2 | 1. T Real =   337.00 ℃   1. P Real =   14059.00 Kpa   1. densitySelection = CDT\_CALCULATION\_FROM\_TP | | Call CalcSteamDensityEXE of MVM to calculate the steam density | | | Density shall be 87.537f kg/m3  Steam shall be Saturated Status. | | Y |
| 3 | 1. Volume flow =   75.38 m3/h   1. QSimEnable = MVM\_DISABLE 2. simulationMode =CDT\_SIMULATION\_CB\_OFF 3. QmRange:   range.hiLim = 8120.0f ;  range.loLim = 120.0f ;  range.span = 8000.0f ; | | Call CalculateSteamQmEXE of MVM to calculate the mass flow | | | mass flow shall be:  6598.560fKg/h | | Y |
| 4 | 1. QmAlmRange   hiLim :90 %  loLim :15 % | | Call GetQmPercentageSRV get the percentage value.  Call UpdateQmAlarmEXE  And Call UpdateDiagnosisEXE to update alarm status. | | | The percentage shall be:  85.326%  Alarm State:  No alarm shall be set | | Y |
|  |  | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

#### Test Case 17

| ID: TC-17 | | Req.-ID:  FR 4.10  FR 4.11  FR 4.15 | | Tester: Zuochen Wang | Test date: | | 2015-11-02 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Purpose | Description: The case tests energy flow calculation for steam | | | | | | | |
| Requirement Details | FR 4.10 ：Measure energy flow for steam  FR 4.11 ：check flow limits  FR 4.15 ：Compute percentage of QMax for energy flow | | | | | | | |
| Pass Criteria | Calculate mass flow Correctly  Calculate Percentage flow correctly | | | | | | | |
| Prepared Data and tools | Tool: IAR Workbench.  Data:  temperaturePreset  dvPressurePreset  dvTExtPreset  steamType  dvDensityPreset  QvMaxDN  T Real  TExt Real  P Real  Mass flow  QpSimEnable  QpSim  QpRange | | | | | | | |
| Procedures | | | | | | | | |
| Steps | Conditions | | Actions | | | Expected results and comments | | Pass  (Y/N) |
| 1 | 1. Density Preset   988 kg/m3   1. QvMaxDn   78.00 m3/h   1. temperaturePreset   430 .0 ℃   1. dvPressurePreset   4000.0   1. dvTExtPreset   470.0 ℃   1. steamType   STEAMTYPE\_OVERHEAT | | Call UpdateSteamPowerMaxDN of MVM to calculate the QpMaxDN. | | | QpMaxDN shall be:  7119942.96 KJ/h | | Y |
| 2 | 1. T Real   350.00 ℃   1. TExt Real   300.00 ℃   1. P Real   70000.00 KPa | | Call CalcSteamHEXEof MVM to calculate the steam enthalpy | | | enthalpy shall be 3016.85 (KJ/KJ)  and  2839.83 (KJ/KJ) | |  |
| 3 | 1. Mass flow   35000.00 Kg/h   1. QpSimEnable MVM\_DISABLE 2. QpSim   5120000.00 KJ/h   1. QpRange   hiLim: 6200000 KJ/h  LoLim: 100 KJ/h  Span: 3199900 KJ/h | | Call CalculateSteamQpEXE of MVM to calculate the Enerty flow | | | Energy flow shall be:  6195700.00 KJ/h  Percentage shall be:  6195700.00 % | |  |
|  |  | |  | | |  | |  |
| Comments: | | | | | | | | |
| Review Result: OK | | | | | | | | |

### Functional Test Results:

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Begin MVMeasurement Function Test

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Begin TC01 Testing--------------------------------------------------------------------

FR 4.1: Measure standard volume flow for liquid Passed

FR 4.13: Compute percentage of QMax Passed

---------------------------------TC01 Testing is Passed-------------------------------

Begin TC02 Testing--------------------------------------------------------------------

FR 4.2 ：Measure mass flow for liquid Passed

FR 4.14: Compute percentage of QMax Passed

FR 4.11: Check Limits Passed

---------------------------------TC02 Testing is Passed-------------------------------

Begin TC03 Testing--------------------------------------------------------------------

FR 4.2 ：Measure mass flow for liquid Passed

FR 4.14: Compute percentage of QMax Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

---------------------------------TC03 Testing is Passed-------------------------------

Begin TC04 Testing--------------------------------------------------------------------

FR 4.2 ：Measure mass flow for liquid Passed

FR 4.14: Compute percentage of QMax Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

---------------------------------TC04 Testing is Passed-------------------------------

Begin TC05 Testing--------------------------------------------------------------------

FR 4.2 ：Measure mass flow for liquid Passed

FR 4.14: Compute percentage of QMax Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

---------------------------------TC05 Testing is Passed-------------------------------

Begin TC06 Testing--------------------------------------------------------------------

FR 3. ：Measure energy flow for liquid Passed

FR 4.15：Compute percentage of QMax for energy flow Passed

---------------------------------TC06 Testing is Passed-------------------------------

Begin TC07 Testing--------------------------------------------------------------------

FR 4. ：Measure standard volume flow for gas Passed

FR 4.13：Compute percentage of QMax for Standard Volume flow Passed

---------------------------------TC07 Testing is Passed-------------------------------

Begin TC08 Testing--------------------------------------------------------------------

FR 4. ：Measure standard volume flow for gas Passed

FR 4.13：Compute percentage of QMax for Standard Volume flow Passed

---------------------------------TC08 Testing is Passed-------------------------------

Begin TC09 Testing--------------------------------------------------------------------

FR 4. ：Measure standard volume flow for gas Passed

FR 4.13：Compute percentage of QMax for Standard Volume flow Passed

---------------------------------TC09 Testing is Passed-------------------------------

Begin TC10 Testing--------------------------------------------------------------------

FR 4. ：Measure standard volume flow for gas Passed

FR 4.13：Compute percentage of QMax for Standard Volume flow Passed

---------------------------------TC10 Testing is Passed-------------------------------

Begin TC11 Testing--------------------------------------------------------------------

FR 4.5 ：Measure mass flow for gas Passed

FR 4.14：Compute percentage of QMax for mass flow Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

---------------------------------TC11 Testing is Passed-------------------------------

Begin TC12 Testing--------------------------------------------------------------------

FR 4.5 ：Measure mass flow for gas Passed

FR 4.14：Compute percentage of QMax for mass flow Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

---------------------------------TC12 Testing is Passed-------------------------------

Begin TC13 Testing--------------------------------------------------------------------

FR 4.6 ：Measure energy flow for gas Passed

FR 4.15：Compute percentage of QMax for energy flow Passed

---------------------------------TC13 Testing is Passed-------------------------------

Begin TC14 Testing--------------------------------------------------------------------

FR 4.7 ：Measure biogas actual volume flow Passed

---------------------------------TC14 Testing is Passed-------------------------------

Begin TC15 Testing--------------------------------------------------------------------

FR 4.8 ：Measure biogas stanadard volume flow Passed

---------------------------------TC15 Testing is Passed-------------------------------

Begin TC16 Testing--------------------------------------------------------------------

FR 4.9 ：Calculate density for gas Passed

FR 4.17 ：Supervisor steam status Passed

FR 4.9 ：Measure mass flow for Steam Passed

FR 4.14：Compute percentage of QMax for mass flow Passed

FR 4.11: Check Limits Passed

FR 4.11: Check Limits Passed

---------------------------------TC16 Testing is Passed-------------------------------

Begin TC17 Testing--------------------------------------------------------------------

FR 4.10 ：Calculate enthalpy for Steam Passed

FR 4.10 ：Calculate enthalpy for Steam Passed

FR 4.10 ：Measure energy flow for Steam Passed

FR 4.15：Compute percentage of QMax for energy flow Passed

---------------------------------TC17 Testing is Passed-------------------------------

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

MVMeasurement Function Test Passed

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

<< Logging of terminal I/O resumes >>

# Test plan and report Review

## Review-Participant:

|  |  |  |
| --- | --- | --- |
| *Dept.* | *Name* | *Date / version* |
| PAMP | Jax Yang, Spring Zhou | 2013-07-08 Ver0.1 |
| PAMP | Jax Yang, Merrick Huang | 2014-02-25 Ver0.2 |
| PAMP | Jax Yang, Merrick Huang | 2014-06-06 Ver0.3 |
| PAMP | Jax Yang, Lawrence Shi | 2015-11-02 Ver0.4 |
| IAMA | Gary-GongYan Chen,Jimmy-Ge Wang | 2018-05-02 Ver0.5 |
| PAMA | Kelly-DongMing Li | 2021-03-31 Ver0.6 |

## Decision of the Review:

|  |  |  |
| --- | --- | --- |
|  | *Decision* | *next steps* |
| Y | Inspection passed ***without restrictions*** | Phase finished |
|  | Inspection passed ***with restrictions*** | Some changes must be done |
|  | Inspection ***not*** passed | Inspection must be repeated |

Changes are proved: The Reviewer confirms that all changes are done:

|  |  |  |
| --- | --- | --- |
| proved Rev: | Date: | Reviewer: |
| 0.1 | 2013-07-08 | Jax Yang, Spring Zhou |
| 0.2 | 2014-02-25 | Jax Yang, Merrick Huang |
| 0.3 | 2014-06-06 | Jax Yang, Merrick Huang |
| 0.4 | 2015-11-02 | Jax Yang, Lawrence Shi |
| 0.6 | 2021-03-31  2021-06-03 | Kelly-DongMing Li  Sunil S. Patil |

**Check list:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | yes | no |
| 1. | The Module Tests are passed successful and documented in a proper way. | **Y** |  |
| 2. | The Code Reviews are passed successful and documented in a proper way. | **Y** |  |
| 3. | The Software System Test is passed successful and documented in a proper way. | **Y** |  |
| 4. | All safety requirements (Safety Function/Integrity/Measures) are considered. | **Y** |  |
| 5. | The accomplished Test Procedures are sufficient too start the Product System Test. | **Y** |  |
| 6. | Are all open issues transferred to the defects table? | **Y** |  |

**Remarks:**

**Defects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No.. | Checkpoint | Description | Major Defect | done  Date |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# References

|  |  |
| --- | --- |
| **Ref.** | **Document** |
| [1] | [RS017]MVMeasurement Requirements Specification |
| [2] | ABB BUI Common Framework release 2.3.1 |
| [3] | PC-Lint for C/C++ version 8  © 2001 Gimpel Software  <http://www.gimpel.com> |
| [4] | [RR014]MVMeasurement Software Code Review |
|  |  |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev.** | **Description of Version/Changes** | **Primary Author(s)** | **Date** |
| 0.1 | Initial revision. | Zuochen Wang | 2013-06-27 |
| 0.2 | Update test case and reteset MVM. | Zuochen Wang | 2014-02-19 |
| 0.3 | reteset MVM. | Zuochen Wang | 2014-06-06 |
| 0.4 | Add Test Case 6~8 for CalcSteamHEXE\_MVM ,  Add Test Case 7~8 CalcSteamDensityEXE\_MVM  Change testing case 1-5 and add testing case 6~7 for CalcSteamHPreset.  Remove Testing for QmAlarmProcess.  Remove Testing for UpdateQmAlarmEXE\_MVM  Add testing case for UpdateQAlarmEXE\_MVM  Add testing case 10~12 for Put\_T\_MVM  Change function testing case: TC02,TC03,TC04,TC05,TC10,TC11,TC12,TC16 | Zuochen Wang | 2015-11-02 |
| 0.5 | Update/Add test Case for CalcSteamHEXE\_MVM  Update/Add test Case for CalcSteamDensityEXE\_MVM  Update/Add test Case for CalcSteamHPrese\_MVM | Zuochen Wang | 2018-05-02 |
| 0.6 | Update for CB firmware version 2.0.2:  1. Add the unit test of QcommonCal.c  2. Update the unit test of all the Q calculation functions.  3. Update the test cases for all the missing else branch | Jaden-Jinding Liu | 2021-06-03 |