|  |  |  |  |
| --- | --- | --- | --- |
| Title | Software Subsystem Test Report <NV Memory Subsystem> | | |
| **Distribution** | Software Archive | | |
| **Author** | Zuochen Wang | Date | 2014-06-10 |
| **Review** | See review section | Date | See review section |
| **Approved** |  | **Date** |  |
| **Remarks** |  | | |

Contents

[1 Introduction 3](#_Toc381088413)

[1.1 Definitions, acronyms, and abbreviations 3](#_Toc381088414)

[2 Strategy 3](#_Toc381088415)

[2.1 Static Check 4](#_Toc381088416)

[2.2 Module test 4](#_Toc381088417)

[3 Environment / Tools 4](#_Toc381088418)

[3.1 PC-lint 4](#_Toc381088419)

[3.2 IAR workbench 4](#_Toc381088420)

[4 Test cases 5](#_Toc381088421)

[4.1 Module Test 5](#_Toc381088422)

[4.1.1 FEBus.c 5](#_Toc381088423)

[4.1.2 Test Coverage 14](#_Toc381088424)

[4.2 Functional test 14](#_Toc381088425)

[5 Code Reviews 14](#_Toc381088426)

[6 Test plan Review 15](#_Toc381088427)

[7 Review-Participant: 15](#_Toc381088428)

[8 Decision of the Review: 15](#_Toc381088429)

[9 Changes are proved: The Reviewer confirms that all changes are done: 15](#_Toc381088430)

[10 References 16](#_Toc381088431)

[11 Revision History 16](#_Toc381088432)

**List of Figures**

[Figure 1: Test Progress 3](#_Toc355340295)

# Introduction

The subsystem is reused from the project 266. A module test and Code review were done by 266 and 2WTCW project. ABB has gotten SIL3 certificate for 266 pressure transmitter, and is applying the software SIL3 certificate for 2WTCW.

VT5 has only one specific file FEBus.c in this subsystem. FEBus.c includes functions for communication between FE and NV Subsystem.

In this document, it covers PC lint results and module test for FEBus.c. Functional test will be included in validation test [3].

## Definitions, acronyms, and abbreviations

| **Term** | **Definition** |
| --- | --- |
| Black Box Test | Test that requires no internal knowledge of the unit under test. |
| 2WCTW | 2 Wire Common Top Works |
| VT5 | A Vortex project developed along with 2-Wire. Many concepts are same between VT5 project and 2-Wire common platform. |
| 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. |
| NV | None Volatile |
| 266 | A pressure product which has passed SIL3. |

# Strategy



Figure 1: Test Progress

## Static Check

Lint [2] 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: ..\FEBus\FEBus.c

--- Module: ..\source\fileList.c

--- Module: ..\source\nv\_mem.c

--- Module: ..\nv\_service\source\nv\_service.c

--- Module: ..\I2C\_Bus\source\I2C\_Bus.c

--- Module: ..\File\source\File.c

--- Module: ..\chip\_handler\source\chip\_handler\_segtab.c

--- Module: ..\chip\_handler\source\chip\_handler.c

--- Global Wrap-up

:

Note 900 -- Successful completion, 0 messages produced

## Module test

* Test should be done automatically, not execute test code step by step thru breakpoint.
* As this subsystem is reused from 266, the module test has been done by ‘266’ project. An extra testing of FEBus will be done and recorded in this document. It will achieve 100% branch coverage and test result will be appending.

# Environment / Tools

## PC-lint

PC-Lint level 3 will be used for code static check on all source files.

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

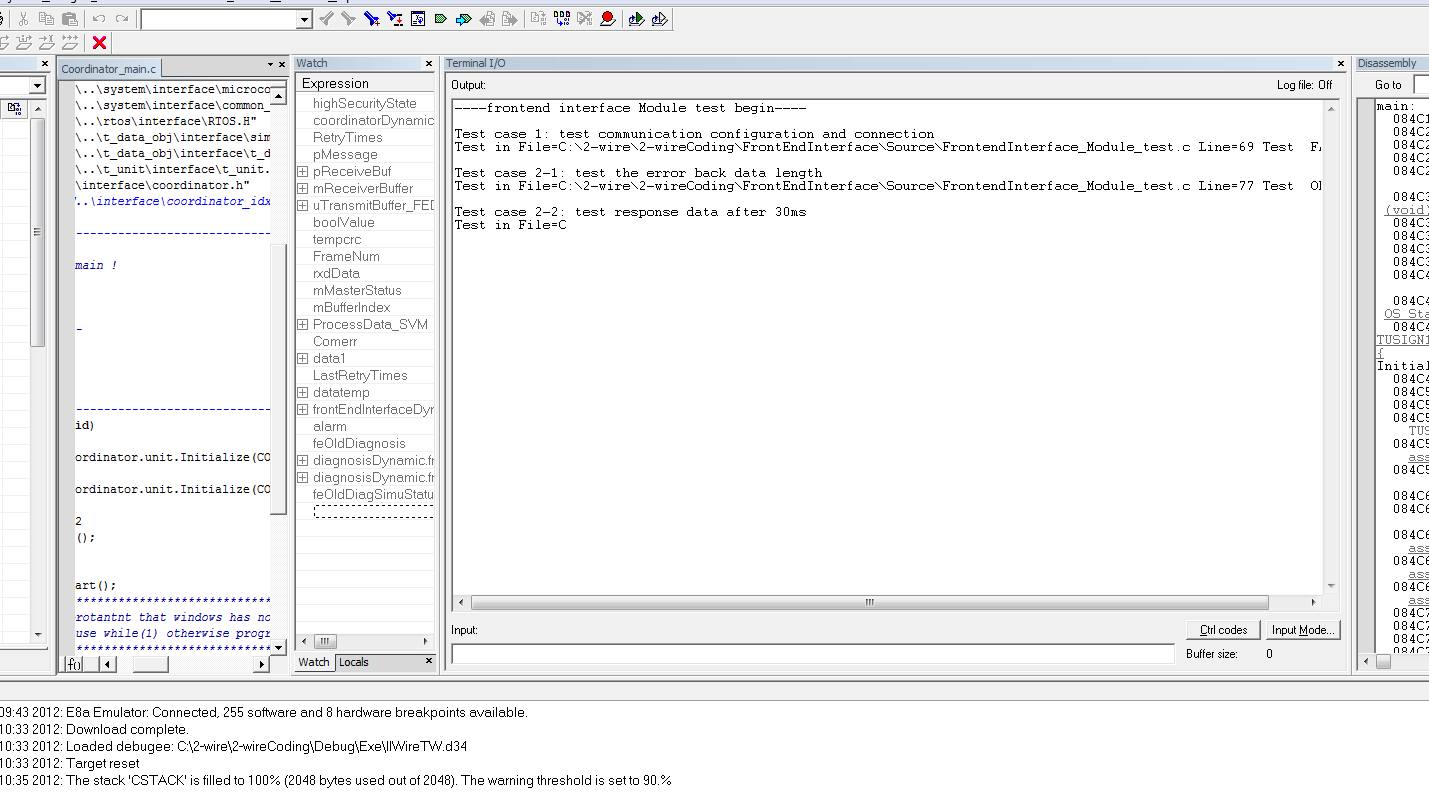
## IAR workbench

IAR Embedded Workbench IDE 3.40

Compiler (v3.40.5)

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

### FEBus.c

|  |  |  |
| --- | --- | --- |
| Fucntion name | TFLOAT SetFlagFEBus() | |
| Test function name | void SetFlagFEBusTest(void) | |
| Test case 1 | Test procedure | 1. Set the following value.   mWaitFlag = 0x00  flagSet = 0x01  mReplaceFlag = 0x00   1. Call SetFlagFEBus(flagSet) |
| Expected result | mWaitFlag == 0X01  mReplaceFlag == 0x01 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set the following value.   mWaitFlag = 0x00  flagSet = 0x00  mReplaceFlag = 0x01   1. Call SetFlagFEBus(flagSet) |
| Expected result | mWaitFlag == 0X01  mReplaceFlag == 0x01 |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void WriteSegment\_Replace(TUSIGN8 length,TUSIGN8\* const pDataBuf) | |
| Test function name | void WriteSegment\_ReplaceTest(void) | |
| Test case 1 | Test procedure | 1. Set the following value.   assertTestOK = 0   1. Call SetFlagFEBus(flagSet) |
| Expected result | assertTestOK = 1 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set data for buffer   memset(wB,0x00,sizeof(wB))  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  mReadBufferReplace[iLoop] = (iLoop+0XA0);  }   1. Call WriteSegment\_Replace(CH\_SEG\_NETTO\_LENGTH,wB) |
| Expected result | wB == mReadBufferReplace |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | void ReadSegment\_Replace(TUSIGN8 length,TUSIGN8\* const pDataBuf) | |
| Test function name | void ReadSegment\_ReplaceTest(void) | |
| Test case 1 | Test procedure | 1. Set test data for buffer   memset(mReadBufferReplace,0x00,sizeof(mReadBufferReplace));  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }   1. Call ReadSegment\_Replace(CH\_SEG\_NETTO\_LENGTH - 0x01,rB); |
| Expected result | Data in mReadBufferReplace do not affect by rb. |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set test data for buffer   memset(mReadBufferReplace,0x00,sizeof(mReadBufferReplace));  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }   1. Call ReadSegment\_Replace(CH\_SEG\_NETTO\_LENGTH - 0x01,rB); |
| Expected result | Data in mReadBufferReplace is the same as rb. |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TUSIGN16 PutNvData\_Replace (TUSIGN8 fIdx,TUSIGN8 sgIdx,const TUSIGN8\* pBuf,TUSIGN8 dataLength) | |
| Test function name | Void PutNvData\_ReplaceTest(void) | |
| Test case 1 | Test procedure | 1. Set Test Data   assertTestOK=0   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | assertTestOK != 0 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set Test Data   memset(wB,0x00,sizeof(wB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = 0x00;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00): repResult |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as wB. |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set Test Data   memset(wB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = 0x00;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00):repResult |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as wB. |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set Test Data   memset(wB,0x00,sizeof(wB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_ERROR;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as wB. |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set Test Data   memset(wB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as wB. |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set Test Data   memset(wB,0x00,sizeof(wB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = 0x00;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_ERROR;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ OK  Feeeprom[0] is the same as wB. |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set Test Data   memset(wB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = 0x00;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as wB. |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set Test Data   memset(wB,0x00,sizeof(wB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_OK;  repResult = REPLACE\_ERROR;  dataLength = CH\_SEG\_NETTO\_LENGTH;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as wB. |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set Test Data   memset(wB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult=REPLACE\_OK;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as wB. |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set Test Data   memset(wB,0x00,sizeof(wB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_NO\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_ERROR;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult=REPLACE\_OK;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as wB. |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set Test Data   memset(wB,0x00,sizeof(wB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  wB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_DISABLED;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult=REPLACE\_OK;   1. Call PutNvData\_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as wB. |
| Test result | Pass |

|  |  |  |
| --- | --- | --- |
| Fucntion name | TUSIGN16 GetNvData\_Replace (TUSIGN8 fIdx,TUSIGN8 sgIdx,const TUSIGN8\* pBuf,TUSIGN8 dataLength) | |
| Test function name |  | |
| Test case 1 | Test procedure | 1. Set Test Data   assertTestOK=0   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | assertTestOK != 0 |
| Test result | Pass |
| Test case 2 | Test procedure | 1. Set Test Data   memset(rB,0x00,sizeof(rB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = 0x00;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_ERROR;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as rB. |
| Test result | Pass |
| Test case 3 | Test procedure | 1. Set Test Data   memset(rB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = 0x00;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ ERROR  Feeeprom[0] is not the same as rB. |
| Test result | Pass |
| Test case 4 | Test procedure | 1. Set Test Data   memset(rB,0x00,sizeof(rB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_ERROR;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as rB. |
| Test result | Pass |
| Test case 5 | Test procedure | 1. Set Test Data   memset(rB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_STATIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK;     1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as rB. |
| Test result | Pass |
| Test case 6 | Test procedure | 1. Set Test Data   memset(rB,0x00,sizeof(rB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = 0x00;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_ERROR;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as rB. |
| Test result | Pass |
| Test case 7 | Test procedure | 1. Set Test Data   memset(rB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = 0x00;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ERROR  Feeeprom[0] is not the same as rB. |
| Test result | Pass |
| Test case 8 | Test procedure | 1. Set Test Data   memset(rB,0x00,sizeof(rB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_ERROR;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_OK  Feeeprom[0] is the same as rB. |
| Test result | Pass |
| Test case 9 | Test procedure | 1. Set Test Data   memset(rB,0x00,CH\_SEG\_NETTO\_LENGTH);  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  dataLength = CH\_SEG\_NETTO\_LENGTH;  FEPutResult = REPLACE\_ERROR;  repResult = REPLACE\_OK;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ ERROR  Feeeprom[0] is not the same as rB. |
| Test result | Pass |
| Test case 10 | Test procedure | 1. Set Test Data   memset(rB,0x00,sizeof(rB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);    for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_NO\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_ERROR;  dataLength = CH\_SEG\_NETTO\_LENGTH;  repResult = REPLACE\_OK;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ ERROR  Feeeprom[0] is not the same as rB. |
| Test result | Pass |
| Test case 11 | Test procedure | 1. Set Test Data   memset(rB,0x00,sizeof(rB));  memset(FEeeprom,0x00,CH\_SEG\_NETTO\_LENGTH);  for(TUSIGN8 iLoop = 0x00; iLoop < CH\_SEG\_NETTO\_LENGTH; iLoop++)  {  rB[iLoop] = (iLoop+0X80);  }    fIdx = FIDX\_CYCLIC\_COMMON\_REPLACE;  sgIdx = FESVR\_SEG;  FEPutResult = REPLACE\_DISABLED;  repResult = REPLACE\_OK;  dataLength = CH\_SEG\_NETTO\_LENGTH;   1. Call GetNvData \_Replace( fIdx,sgIdx,wB, 0X00) |
| Expected result | repResult == REPLACE\_ ERROR  Feeeprom[0] is not the same as rB. |
| Test result | Pass |

### Test Coverage

**File:**FEBus.c

|  |  |  |
| --- | --- | --- |
| Num | Function | Coverage(%) |
| 1 | SetFlag | 100 |
| 2 | WriteSegment\_Replace | 100 |
| 3 | ReadSegment\_Replace | 100 |
| 4 | PutNvData\_Replace | 100 |
| 5 | GetNvData\_Replace | 100 |
|  |  |  |

## Functional test

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

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

The function test will be covered in the validation test [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 [3] have been followed.

The reviews will also aim to identify defects in the source code for the NV Memory 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 NV Memory subsystem code review [3].

# Test plan Review

# Review-Participant:

|  |  |  |
| --- | --- | --- |
| *Dept.* | *Name* | *Date/Version* |
| SH R&D | Merrick Huang | 2013-07-23 Ver0.1 |
| SH R&D | Merrick Huang | 2013-07-24 Ver0.2 |
| SH R&D | Merrick Huang | 2014-02-25 Ver0.3 |
| SH R&D | Merrick Huang | 2014-06-10 Ver0.4 |

# 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: |
| 0.2 | 2013-07-24 | Merrick Huang |
| 0.3 | 2014-02-25 | Merrick Huang |
| 0.4 | 2014-06-10 | Merrick Huang |

**Check list:**

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

**Remarks:**

**Defects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No.. | Checkpoint | Description | Major Defect | done  Date |
| 1 |  | In chapter 1, the introduction should be more detailed about 266. As this subsystem is reused from 266. | N | 2013-07-24 |
| 2 |  | The documents number should be added in reference. | N | 2013-07-24 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# References

|  |  |
| --- | --- |
| **Ref.** | **Document** |
| [1] | [RS004] VT5 Software Requirements Specification. |
| [2] | PC-Lint for C/C++ version 8  © 2001 Gimpel Software  <http://www.gimpel.com> |
| [3] | [RR015] NV MEM Software Code Review. |
| [4] | [TP001] Validation test plan. |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev.** | **Description of Version/Changes** | **Primary Author(s)** | **Date** |
| 0.1 | Initial revision. | Zuochen Wang | 2012-07-17 |
| 0.2 | Update the docuement according to review comments. | Zuochen Wang | 2013-07-24 |
| 0.3 | Add detailed test cases description for FEBus.c | Zuochen Wang | 2014-02-18 |
| 0.4 | Retest NV | Zuochen Wang | 2014-06-10 |