TEST SPECIFICATION

HART COMMUNICATION PROTOCOL

Slave Universal Command, Test Specification

HCF_TEST-003, Revision 4.0

Release Date: 11 March 2009

Release Date: 11 March 2009

Document Distribution / Maintenance Control / Document Approval
To obtain information concerning document distribution control, maintenance control, and document approval please contact the HART Communication Foundation (HCF) at the address shown below.

Trademark Information
HART® is a registered trademark of the HART Communication Foundation, Austin, Texas, USA. Any use of the term HART hereafter in this document, or in any document referenced by this document, implies the registered trademark. WirelessHART™ is a trademark of the HART Communication Foundation. All other trademarks used in this or referenced documents are trademarks of their respective companies. For more information contact the HCF Staff at the address below.

Attention: Foundation Director
HART Communication Foundation
9390 Research Boulevard
Suite I-350
Austin, TX 78759, USA
Voice: (512) 794-0369
FAX: (512) 794-3904

http://www.hartcomm.org

Intellectual Property Rights
The HCF does not knowingly use or incorporate any information or data into the HART Protocol Standards which the HCF does not own or have lawful rights to use. Should the HCF receive any notification regarding the existence of any conflicting Private IPR, the HCF will review the disclosure and either (a) determine there is no conflict; (b) resolve the conflict with the IPR owner; or (c) modify the standard to remove the conflicting requirement. In no case does the HCF encourage implementers to infringe on any individual's or organization's IPR.

Table of Contents

Preface

This preface is included for informational purposes only.

This Test Specification is a companion to Revision 7.0 of the *Universal Command Specification*. The principle change to this version of the test specification is to provide support for HART7. This resulted in changes to UAL000, UAL007, UAL011 and the addition of UAL038, UAL048. UAL038 and UAL048 were also updated to HART 7 requirements. Specific changes include:

- UAL000: Test vectors were added for command 38 and 48; The Failure point code numbering and methodology was changed with the addition of more Universal Commands.
- UAL007: Test on the Private Label Distributor code were updated to HART 7 requirements.
- UAL009: Write protect tests were updated to include some preliminary tests manipulating the Configuration Changed status bit.
- UAL011: A table was added identifying the byte counts for HART 6 and 7 Command 9 responses and the test was updated accordingly. A second test case was added to verify support for mandatory device variables in HART 7 devices.
- UAL038: Updated to support HART 7 requirements. A second test case was added to verify the Configuration Changed status bit will only be reset when the correct Configuration Change Counter value is sent with Command 38. Also verified "Too Few Data Bytes Received" Response Code.
- UAL048: Updated to support HART 7 requirements. Verify "More Status Available" status bit can be reset in HART 7 devices.

In addition, a number of minor typos were also corrected.

Introduction

The HCF QA Program is designed to verify compliance with HART Protocol requirements. In other words, The HCF is committed to ensuring that Field Devices and Host Applications can successfully work together as a system no matter who supplies the Field Device or Host.

The Test Specifications are a key part of the HART QA Program and Device Registration. End-users look for the "HART Registered" mark to ensure the products they receive have successfully completed all HART QA Program requirements. All devices should be HART Registered.

This document is only a part of the HCF QA Program and it is designed to compliment the other test Specifications (e.g., Physical Layer and Data Link Layer). In fact, some Universal Commands (e.g., Command 0) are actually tested in the Slave Data Link Layer Test Specifications. These Test Specifications:

- Provide clear test requirements. The Test Specifications reduce the number of the Test Plans that must be developed by the manufacturer.
- Must be completed along with the Test Report prior to product release and product registration with the HCF.
- Clarify ambiguities in the Protocol. Since this specification is balloted and approved like all other HART Specifications it is equally binding.

This document defines tests for HART Universal Command requirements. These tests (generally) become useful later in the development life-cycle. They should be performed only after Physical Layer and Data Link Layer testing is successfully completed.

The Universal Command tests can be classified as follows:

- **Verify Support for All Universal Commands**. The Universal Command number range is scanned. All numbers without a corresponding Command Specification must return "Command Not Implemented". Valid Universal Commands must answer with "Success" or "Too Few Data Bytes Received" as required by the Command Specification.
- **Read Dynamic Variables**. Responses to Command 1, 2, 3, and 9 are evaluated. Universal Command, Command Summary and Common Tables requirements must be adhered to.
- **Verify Write Commands**. For each Write Command:
  - The DUT initial value is read and, upon completion of the test, restored
  - Several values are written and then read back from the DUT to confirm the success of the Write Command
  - Long and short Response Data Fields are sent and proper DUT reaction verified.

  In addition, proper operation of the Configuration Change Counter is confirmed.

- **Verify Configuration Read Commands**. Operation of Read Commands are verified. Only a basic "sanity check" of the data is performed at this time.

- **Confirm Write Protect**. The DUT is placed in Write Protect Mode (if supported) and all Write Commands issued. The DUT's database must not change

- **Test Cold Start Bit**. Each master must be able to detect a Cold Start (i.e., a power failure). In addition, retention of the Configuration Change Counter is confirmed.

- **Read Device Variables**. Proper operation of Command 9 is confirmed.

- **Test the Configuration Changed Bit**. Support for and proper operation of Command 38 is confirmed. Use of Configuration Changed Counter to reset the status bit is also confirmed.

- **Support for Command 48, Read Additional Device Status**. Support for Command 48 is confirmed. The ability to reset the More Status Available bit is verified.

Manufacturers of Field Devices must execute and pass all of these conformance tests.

1. SCOPE

This document defines conformance tests for the HART Universal Commands. Conformance with the *Universal Command Specification* is mandatory. This Test Specification provides Field Device implementers with a set of tests to assist in verifying conformance to the Application Layer and Universal Command requirements.

Field Devices must successfully complete all tests in this document.

Note: The DUT should successfully complete all tests in the *Slave Data Link Layer Test Specification* before attempting the testing prescribed by this document.

1.1 Features Tested

All major Universal Command features in the DUT are tested. This includes:

- Tests to provoke every allowed Response Code;
- Verification of data written to the DUT;
- Proper application of "Busy" and Delayed Response;
- Confirmation of DUT Short, Long and Broadcast Addresses;
- Support for both transmitters and actuators;
- Detailed tests for all Universal Commands.

1.2 Features Not Tested

Some features of the Universal Command Specification are not tested including:

- Commands tested in Data Link Layer Tests (0, 6, 11, 21)
- Proper conversion of DUT values is not confirmed. For example, the Loop Current, Percent of Range, and PV values are not compared (e.g., using the Upper and Lower Range Values)

2. REFERENCES

2.1 The HART-Field Communications Protocol Specifications

These documents published by the HART Communication Foundation are referenced throughout this specification:

Data Link Layer Specification, HCF_SPEC-81

Command Summary Specification, HCF_SPEC-99

Universal Command Specification, HCF_SPEC-127

Common Practice Command Specification, HCF_SPEC-151

Common Tables, HCF_SPEC-183

2.2 Related Documents

The following documents provide guidance and background information used in developing this Test Specification.:

IEEE Standard for Software Test Documentation, ANSI/IEEE Std 829

IEEE Standard for Software Unit Testing, ANSI/IEEE Std 1008

3. DEFINITIONS

Definitions for terms can be found in the *HART Communications Protocol Specification.* Terms used in this document include: ASCII, Broadcast Address, Data Link Layer, Delayed Response, Delayed Response Mechanism, Device Reset, Device Variable, Busy, Dynamic Variable, Fixed Current Mode, Floating Point, ISO Latin-1, Master, Multi-drop, Not-A-Number, Packed ASCII, Preamble, Request Data Bytes, Response Data Bytes, Response Message, Slave, Slave Time-Out, Software Revision Level, Time Constant, Units Code.

Some other terms used only within the context of the *Universal Command, Test Specification* are:

BYTE_COUNT	Refers to the value contained in the Byte Count Field of the DUT response.
COMMUNICATIONS_ ERROR	Indicates that communications itself was unsuccessful. In other words, there was no response or the DUT detected a communications error (see the *Command Summary Specification*).
Primitive Test	A Test designed to verify conformance with a narrowly focused set of requirements found in the HART Field Communications Protocol (see Test). Each Primitive Test consists of both Test Case(s) and the corresponding Test Procedure(s).
RESPONSE_CODE	When communications is successful (from a Data Link Layer viewpoint) a slave indicates the correctness of the master response using this byte (see the *Command Summary Specification*).
Test	A set of one or more Test Cases and Test Procedures.
Test Case	A narrowly focused set of conditions, inputs and expected outputs designed to verify proper operation of the DUT.
Test Procedure	A sequence of steps or actions designed to fully execute a Test Case.

4. SYMBOLS/ABBREVIATIONS

DUT	**D**evice **U**nder **T**est
HCF	**H**ART **C**ommunication **F**oundation
LEP	**L**ower **E**nd **P**oint
LRV	**L**ower **R**ange **V**alue
LTL	**L**ower **T**ransducer **L**imit
MRV	**M**ost **R**ecent **V**alue
SOM	**S**tart **O**f **M**essage
UEP	**U**pper **E**nd **P**oint
URV	**U**pper **R**ange **V**alue
UTL	**U**ppe**r** **T**ransducer **L**imit

5. APPROACH

This Test Specification uses a "black box" approach to confirming compliance with Universal Command requirements. Testing is decomposed into a series of narrowly focused Tests, each containing one of more test cases and test procedures.

- Each test is described in a narrative form, in some cases, with the assistance of tables containing test vectors.
- The test procedures are described using pseudo code.
- Within each test procedure termination points are uniquely numbered. This allows cross-referencing should the DUT fail a test.

Support for all Universal Commands is mandatory. All Field Devices must pass all tests in this document.

5.1 Testing Sequence

Since all Universal Commands must be supported the actual testing sequence is not particularly critical. However, some tests require operator interaction and it may be somewhat most efficient to group those tests together. Consequently, the following table shows the recommended order of testing.

Table 1 Test Execution Sequence

No.	Test	No.	Test	No.	Test	No.	Test
1	UAL000	5	UAL012	9	UAL013	13	UAL048
2	UAL001	6	UAL005	10	UAL009		
3	UAL007	7	UAL006	11	UAL010		
4	UAL011	8	UAL008	12	UAL038		

5.2 Conventions

Throughout the Test Definitions, some conventions are used. The most common are references to the command status bytes (i.e., Communication Status, Field Device Status, and Response Codes). References to these data are explained in the following sections. In addition, angle brackets are often included in test vectors or the pseudo code. Text shown in italics between < > brackets is to be replaced by the corresponding data for the DUT.

5.2.1 Communication Errors

A "COMMUNICATIONS_ERROR" consists of one or more of the following error indications (see *Command Summary Specification*):

- No Response
- Vertical Parity Error (i.e., Parity Error)
- Longitudinal Parity Error (i.e., Bad Check Byte)
- Framing Error
- Overrun Error
- Buffer Overflow

5.2.2 Response Code

"RESPONSE_CODE" indicates whether a Slave Device considered the master request valid or not. Common Response Codes used in this document include (see *Command Response Code Specification*):

Table 2 Common Response Codes

Slave Indication	Code No.
"Success"	0
"Invalid Selection"	2
"Too Few Data Bytes Received"	5
"Update Failure"	8
"Busy"	32
"Delayed Response Initiated"	33
"Delayed Response Running"	34
"Command Not Implemented"	64

5.2.3 Device Status

A "DEVICE_STATUS" consists of one or more of the following status indications (see *Command Summary Specification*):

- Device Malfunction (bit 7)
- Configuration Changed (bit 6)
- Cold Start (bit 5)
- More Status Available (bit 4)
- Loop Current Fixed (bit 3)
- Loop Current Saturated (bit 2)
- Non-PV Out of Limits (bit 1)
- PV Out of Limits (bit 0)

5.3 Comparing Floating-Point Numbers

Floating-point numbers are widely used in Universal Commands. As a result, the values of floating point numbers are sometimes compared. To ensure consistent results, actual testing must use a small "delta" when comparing floating-point numbers. Delta must be chosen to be as large as the value of a few (3-6) least significant bits of the fractional part of the floating-point number. For equalities, this delta establishes a dead-band. For inequalities, the dead-band is a small offset to the benefit of the DUT. When comparing loop current values, the dead-band is fixed at .05 mA.

6. DELIVERABLES

The Test Report included in ANNEX C shall be completed for each Field Device tested. This Test Report is a simple checklist indicating:

- Who performed the tests;
- What Field Device was used for testing;
- When the testing was completed; and
- The completion status for each test.

In some cases, (e.g., UAL048) a test may require additional data or documentation to be supplied. This must be attached to the Test Report.

This Test Report provides: a record of the testing; will satisfy most Quality Assurance Audits, and provides sufficient detail to allow the test results to be reproduced. The Test Report must be included when submitting the product to the HCF. This test report will be audited and tests may be repeated prior to the HCF completing Device Registration

Note: Submission of devices for registration is not limited to delivering the completed test report and supporting details. Other supporting materials and documentation must also provided as indicated by HCF procedures and policies.

7. TEST DEFINITIONS

7.1 UAL000 Confirm All Universal Commands Supported

Verifies the DUT responds properly to all Command 0 through 30. Basic command operation is verified as follows (see also the test vectors in Table 3):

- All commands must be answered.
- All defined Universal Commands must be implemented in the DUT as specified.
- Undefined Universal Commands must answer "Command not Implemented".
- All Commands are dispatched with no data bytes. Commands requiring request data bytes must answer "Too Few Data Bytes Received".
- All responses must contain the same command number as sent, valid response data, and the appropriate Response Code.

Each command in Table 3 is sent with zero data bytes so that the Write Commands will be rejected with Response Code = "Too Few Data Bytes Received" (i.e., RC=5) and not alter the DUTs configuration. The slave must respond to each command, replying with a Valid Address, Command Number, Byte Count and Response Code.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	All
Universal Command Specification	5.2	All

Table 3 Test Vectors for Scan of Universal Commands

Command (Cmd)	Legal Response Codes (RC)	Byte Count (BC)
1	0, 8	7
2	0, 8	10
3	0, 8	11, 16, 21 or 26
4	64	2
5	64	2
6	5	2
7	0	4
(H5)	64	2
8	0	3, 4, 5, 6
(H5)	64	2
9	5	2
(H5)	64	2
10	64	2
12	0	26

Command (Cmd)	Legal Response Codes (RC)	Byte Count (BC)
13	0	23
14	0	18
15	0	20
(H5)	0	19
16	0	5
17	5	2
18	5	2
19	5	2
20	0	34
(H5)	64	2
22	5	2
(H5)	64	2
23	64	2
24	64	2

Command (Cmd)	Legal Response Codes (RC)	Byte Count (BC)
25	64	2
26	64	2
27	64	2
28	64	2
29	64	2
30	64	2
38	0	4
(H5)	0, 64	2
(H6)	0, 64	2

Command (Cmd)	Legal Response Codes (RC)	Byte Count (BC)
48	0	11, 12, 14-27
(H5)	0, 64	2-8, 16-27
(H6)	0, 64	2, 10, 16-27

* (H5) and (H6) refer to correct values for HART 5 and HART 6 devices, respectively.

Note: Commands 11 and 21 are not tested here because a tag is required in the request bytes.

Test Procedure

Initialize response table. This table stores the correct Response Code and Byte Count for each Common Practice Command when messaged with zero data bytes.

```
CALL IdentifyDevice
```

Sequentially send Commands in Table 3 with zero data bytes.

```
FOR each TEST_CASE in Table 3
   SEND Cmd with zero data bytes
   IF there is no response
      THEN Test result is FAIL                    (2000+TEST_VECTOR_NUMBER)
   END IF
   CALL TestValidFrame()
   IF (RESPONSE_CODE is not in list)
      THEN Test Result is FAIL                    (2035+TEST_VECTOR_NUMBER)
   END IF
   IF (BYTE_COUNT is not in list)
      THEN Test Result is FAIL                    (2070+TEST_VECTOR_NUMBER)
   END IF
END FOR

END TEST
```

7.2 UAL001 Read Dynamic Variables (Commands 1, 2, and 3)

Checks Addresses, Command Number, Response Code and Byte Count for Commands 1, 2 and 3

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
```

Use Upper and Lower Sensor Limits to set dead band on PV comparisons

```
SEND Command 14 to get upper and lower sensor limits
```

Sequentially send Commands 1 through 3.

```
FOR (100 iterations)
   FOR (cmd = [1, 2, and 3] )
      SEND Command <cmd>
      CALL TestValidFrame()
      IF ( (RESPONSE_CODE != 0)
               AND (RESPONSE_CODE!= "Update Failure") )
         THEN Test result is FAIL                                  (2105+cmd)
      END IF

      SWITCH on (cmd)
      CASE 1
         IF (BYTE_COUNT != 7)
            THEN Test result is FAIL                               (2111)
         END IF
      CASE 2
         IF (BYTE_COUNT != 10)
            THEN Test result is FAIL                               (2117)
         END IF
      CASE 3
         IF (BYTE_COUNT != [ 11, 16, 21 or 26] )
            THEN Test result is FAIL                               (2123)
         END IF
   END FOR
END FOR
SEND Command 1 to read cmd1PV
SEND Command 2 to read cmd2LoopCurrent
SEND Command 3 to read cmd3PV, cmd3LoopCurrent
IF (cmd1PV != cmd3PV)
   THEN Test result is FAIL                                        (2125)
END IF

IF (cmd3LoopCurrent != cmd2LoopCurrent)
   THEN Test result is FAIL                                        (2126)
END IF
END TEST
```

7.3 UAL002 (Reserved)

In previous versions of this document UAL002 confirmed that commands 1-3 implemented the "Update Failure" Response Code properly. This Response Code is now supported in UAL001 and, as a result, this test is retired.

7.4 UAL003 (Reserved)

In previous versions of this document UAL003 verified operation of Command 6, Write Polling Address. This testing has always been performed in the *Slave Data Link Layer, Test Specification*. To eliminate unnecessary redundancy, this test has been retired.

7.5 UAL004 (Reserved)

In previous versions of this document UAL004 verified operation of Command 11, Read Unique Identifier Associated With Tag. This testing has always been performed in the *Slave Data Link Layer, Test Specification*. To eliminate unnecessary redundancy, this test has been retired.

7.6 UAL005 Write Message

Verifies the DUT responds properly to Command 17. Checks Addresses, Command Number, Response Code and Byte Count for Commands 12 and 17. In addition, it checks the message content to verify proper message data in the slave device.

The test sends Command 17 followed by Command 12 to confirm the DUT actions. This sequence of commands is repeated for the following conditions:

- Valid message content (new message must be accepted);
- Valid message with an extra data byte (new message must be accepted); and
- Invalid message short one data byte (new message must be rejected).

When the slave is required to respond, it shall reply with a Valid Address, Command Number, Byte Count and Response Code for each command.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.12, 6.17
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
CALL VerifyNotWriteProtected()
CALL IssueCommand12 (to read msg0, 2140)                              (2140)
IF ( UNIV_REVISION >= 6 )
   SEND Command 0 to read the configuration changed counter (cfgCntr)
END IF
```

Send Command 17 with a valid message. Slave should reply to command 12 with the same message.

```
Create message1 with 24 bytes of 0x00
SEND Command 17 with message1
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 26)
SET cfgCntr = cfgCntr + 1
CALL ValidateMsg(message1, cfgCntr,2155)                          (2155-2157)
```

Send Command 17 with a changed valid message + one byte. Slave should reply to command 12 with the new message, but with exactly 24 bytes.

```
Create message2 with 24 bytes of 0xCC
SEND Command 17 with message2 with one extra byte
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 26)
SET cfgCntr = cfgCntr + 1
CALL ValidateMsg(message2, cfgCntr, 2175)                         (2175-2177)
```

Send Command 17 with a 23-byte message. Slave should not store the invalid message and should reply to command 17 with Too Few Data Bytes Received. Slave should reply to command 12 with 24 bytes of 0xCC.

```
Create message3 with 23 bytes of 0x00
SEND Command 17 with message3
CALL TestValidFrame
CALL VerifyResponseAndByteCount("Too Few Data Bytes Received",2)
CALL ValidateMsg(message2, cfgCntr, 2195)                          (2195-2197)
```

Restore original message

```
SEND Command 17 with msg0
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 26)
SET cfgCntr = cfgCntr + 1
CALL ValidateMsg(msg0, cfgCntr, 2215)                              (2215-2217)

END TEST
```

ValidateMsg (msg, cfgCntr, failurePoint)

This procedure is unique to UAL005. It uses command 12 to read the "message" and compare it to "msg".

```
PROCEDURE ValidateMsg(msg, cfgCntr, failurePoint)

CALL IssueCommand12 (msg0, failurePoint)                        (failurePoint)
CALL VerifyResponseAndByteCount(0, 26)
CALL TestValidFrame
IF (msg != msg0)
   THEN Test result is FAIL                                    (failurePoint+1)
END IF

IF ( UNIV_REVISION >= 6 )
   SEND Command 0 to read the = configuration
      changed counter (cfgCntr0)
   IF (cfgCntr != cfgCntr0)
      THEN Test result is FAIL                                 (failurePoint+2)
   END IF
END IF

PROCEDURE END
```

7.7 UAL006 Write Tag Descriptor and Date

Verifies the DUT responds properly to Commands 13 and 18. Checks Addresses, Command Number, Response Code and Byte Count for Commands 13 and 18. In addition, it checks the message content to verify proper message data in the slave device.

The test sends Command 18 followed by Command 13 to confirm the DUT actions. This sequence of commands is repeated for the following conditions:

- 21 Data Bytes, valid date.
- 21 Data Bytes, invalid date.
- Changed 22 Data Bytes, valid date.
- Changed 20 Data Bytes, valid date.

When the slave is required to respond, it shall reply with a Valid Address, Command Number, Byte Count and Response Code for each command.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.13, 6.18
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
CALL VerifyNotWriteProtected()
CALL IssueCommand13 (to read tag0, desc0, date0, 2300)                (2300)
SEND Command 0 to read the configuration changed counter (cfgCntr)
```

Send Command 18 with a valid message and valid date. Slave should reply to command 13 with the same data.

```
SET tag1 = 6 bytes of 0x00
SET desc1 = 12 bytes of 0x00

SET date1 = "0x01, 0x01, 0x00"
SET invalidDate = "0xFF, 0xFF, 0x00"

SEND Command 18 (with tag1, desc1, date1)
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 23)
SET cfgCntr = cfgCntr + 1
CALL ValidateTDD(tag1, desc1, date1, cfgCntr, 2320)              (2320-2324)
```

Send Command 18 with a valid message and invalid date. The DUT may not check the date code

```
SET tag2 = 6 bytes of 0xCC
SET desc2 = 12 bytes of 0xCC

SEND Command 18 (with tag2, desc2, invalidDate)
CALL TestValidFrame
IF (RESPONSE_CODE == 0) THEN
   CALL VerifyResponseAndByteCount(0, 23)
   SET cfgCntr = cfgCntr + 1
   CALL ValidateTDD (tag2, desc2, invalidDate,
      cfgCntr, 2340)                                          (2340-2344)
ELSE
   CALL VerifyResponseAndByteCount("Invalid Date Code Detected", 2)
   CALL ValidateTDD(tag1, desc1, date1,
      cfgCntr, 2365)                                          (2365-2369)
ENDIF
```

Send Command 18 with a changed message and valid date + one byte. Slave should reply to command 13 with the new message, but with exactly 23 bytes.

```
SET tag3 = 6 bytes of 0x33
SET desc3 = 12 bytes of 0x33
SET date3 = "0x12, 0x04, 0x65"

SEND Command 18 (with tag3, desc3, date3) with an extra byte
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 23)
SET cfgCntr = cfgCntr + 1
CALL ValidateTDD(tag3, desc3, date3, cfgCntr, 2380)           (2380-2384)
```

Send Command 18 with a 20 byte message. Slave should not store the invalid message and should reply to command 18 with Too Few Data Bytes Received.

```
SEND Command 18 (with tag2, desc2, (invalidDate – one byte) )
CALL TestValidFrame
CALL VerifyResponseAndByteCount("Too Few Data Bytes Received", 2)
CALL ValidateTDD(tag3, desc3, date3,
   cfgCntr, 2400)    (2400-2404)
```

Restore original tdd

```
SEND Command 18 (with tag0, desc0, date0)
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 23)
SET cfgCntr = cfgCntr + 1
CALL ValidateTDD(tag0, desc0, date0, cfgCntr, 2420)           (2420-2424)

END TEST
```

ValidateTDD (tag, desc, date, cfgCntr, failurePoint)

This procedure is unique to UAL006. It uses command 13 to read and verify the tag, descriptor and date.

```
PROCEDURE ValidateTDD(tag, desc, date, cfgCntr, failurePoint)

SEND Command 11 (with tag)
CALL TestValidFrame
IF ( UNIV_REVISION >= 6 )
   THEN CALL VerifyResponseAndByteCount(0, 19)
ELSE
   THEN CALL VerifyResponseAndByteCount(0, 14)
END IF

CALL IssueCommand13 (to read tag0, desc0,                    (failurePoint)
   date0, failurePoint)
CALL VerifyResponseAndByteCount(0, 23)
IF (tag != tag0)
   THEN Test result is FAIL                                (failurePoint+1)
END IF
IF (desc != desc0)
   THEN Test result is FAIL                                (failurePoint+2)
END IF
IF (date != date0)
   THEN Test result is FAIL                                (failurePoint+3)
END IF

IF ( UNIV_REVISION >= 6 )
   SEND Command 0 to read the configuration
      changed counter (cfgCntr0)
   IF (cfgCntr != cfgCntr0)
      THEN Test result is FAIL                             (failurePoint+4)
   END IF
END IF

PROCEDURE END
```

7.8 UAL007 Verify Command 14 and 15 Response

Verifies the DUT responds properly to Commands 14 and 15. Checks addresses, Command Number, Response Code and Byte Count for Commands 14 and 15.

Note: In some devices transducer units or range units may not be the same as PV units. When this occurs the HCF may grant waiver on Failure Point 2510 or 2518. To request a waiver please provide detailed, written justification for this deviation to the HCF. The HCF will assess the justification and, when appropriate, grant the waiver.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.14, 6.15
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
SEND Command 1 to read pvUnits
```

Send Command 14 to read primary variable transducer information

```
SEND Command 14
CALL VerifyResponseAndByteCount(0, 18)
CALL TestValidFrame
```

Send Command 15 to read device information

```
SEND Command 15
IF (UNIV_REVISION >= 6) THEN
   CALL VerifyResponseAndByteCount(0, 20)
ELSE
   CALL VerifyResponseAndByteCount(0, 19)
END IF
CALL TestValidFrame
```

Do a sanity check on the transducer units

```
IF (analog channel flag indicates loop current is input/setpoint)
   IF (pvUnits != percent)
      THEN Test result is FAIL                                    (2515)
   END IF
   IF (transducer units != mA)
      THEN Test result is FAIL                                    (2516)
   END IF

ELSE IF (transducer units != 250) THEN
   IF (pvUnits != transducer units)
      THEN Test result is FAIL                                    (2510)
   END IF
ELSE
   IF (utl != "0x7F, 0xA0, 0x00, 0x00")
         OR (ltl != "0x7F, 0xA0, 0x00, 0x00")
         OR (minimum span != "0x7F, 0xA0, 0x00, 0x00")
      THEN Test result is FAIL                                    (2520)
   END IF
END IF
```

Do a sanity check on the range units

```
IF (urv == "0x7F, 0xA0, 0x00, 0x00")
          OR (lrv == "0x7F, 0xA0, 0x00, 0x00")
          OR (damping == "0x7F, 0xA0, 0x00, 0x00")
   THEN Test result is FAIL                                          (2532)
END IF

IF (analog channel flag indicates loop current is input/setpoint)
   IF (range units != mA)
      THEN Test result is FAIL                                       (2517)
   END IF
ELSE IF (pvUnits != range units)
      THEN Test result is FAIL                                       (2518)
END IF

IF (write protect != [0, 1, or 240-253])
   THEN Test result is FAIL                                          (2534)
END IF

IF (alarm code != [0, 1, or 239-253])
   THEN Test result is FAIL                                          (2536)
END IF

IF (UNIV_REVISION < 7) THEN
   IF (private label distributor code > 239)
      THEN Test result is FAIL                                       (2538)
ELSE
   IF (private label distributor code != 250)
      THEN Test result is FAIL                                       (2539)
END IF

END TEST
```

7.9 UAL008 Write Final Assembly Number

Verifies the DUT responds properly to Commands 16 and 19. Checks Addresses, Command Number, Response Code and Byte Count for Commands 16 and 19. In addition, it checks the message content to verify proper message data in the slave device. The test sends Command 19 followed by Command 16 to confirm the DUT actions. This sequence of commands is repeated for the following conditions:

- Valid message content.
- Changed Valid message with an extra data byte.
- Changed Invalid message short one data byte.

When the slave is required to respond, it shall reply with a valid address, Command Number, Byte Count and Response Code for each command.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.16, 6.19
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
CALL VerifyNotWriteProtected()
IF ( UNIV_REVISION >= 6 )
   SEND Command 0 to read the configuration changed counter (cfgCntr)
END OF
SEND Command 16 to read fan0
```

Valid message content.

```
SEND Command 19 with assembly number 1
CALL VerifyResponseAndByteCount(0, 5)
CALL TestValidFrame
SET cfgCntr = cfgCntr + 1
CALL ValidateAsemblyNumber(1, cfgCntr, 2620)                    (2620-2622)
```

Valid message content with one extra data byte.

```
SEND Command 19 with assembly number 2 and one extra data byte
CALL VerifyResponseAndByteCount(0, 5)
CALL TestValidFrame
SET cfgCntr = cfgCntr + 1
CALL ValidateAsemblyNumber(2, cfgCntr, 2640)                    (2640-2642)
```

Valid message content with one data byte short.

```
SEND Command 19 with assembly number 300 and one data byte short
CALL VerifyResponseAndByteCount(5, 2)
CALL TestValidFrame
CALL ValidateAsemblyNumber(2, cfgCntr, 2660)                    (2660-2662)

SEND Command 19 to restore fan0
END TEST
```

ValidateAssemblyNumber (fan, cfgCntr, failurePoint)

This procedure is unique to UAL008. It uses command 16 to read the assembly number and compare it to "fan".

```
PROCEDURE ValidateAssemblyNumber(fan, cfgCntr, failurePoint)

CALL IssueCommand16 (fan0, failurePoint)                      (failurePoint)
CALL VerifyResponseAndByteCount(0, 5)
CALL TestValidFrame
IF fan0 != fan
   THEN Test result is FAIL                                  (failurePoint+1)
END IF

IF ( UNIV_REVISION >= 6 )
   SEND Command 0 to read the = configuration
      changed counter (cfgCntr0)
   IF (cfgCntr != cfgCntr0)
      THEN Test result is FAIL                               (failurePoint+2)
   END IF
END IF

PROCEDURE END
```

7.10 UAL009 Verify Write Protect

Verifies the DUT responds properly in Write Protect. Responses from Commands 6, 17, 18, 19, and 22 are verified. Checks Addresses, Command Number, Response Code and Byte Count for each command.

Each command is sent and the "In Write Protect Mode" Response Code is verified. The DUT Data is checked to confirm no changes have occurred.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.6, 6.17, 6.18, 6.19, 6.21
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
SEND Command 15
IF (UNIV_REVISION >= 6)THEN
   CALL VerifyResponseAndByteCount(0, 20)
   SEND Command 0 to read the configuration changed counter (cfgCntr)
ELSE
   THEN CALL VerifyResponseAndByteCount(0, 19)
END IF
CALL TestValidFrame
IF (write protect = 251, "None")
   THEN Test result is ABORT                                    (2710)
END IF
```

Setup for testing command 38 in write protect. Command 17 (Write Message) is sent to write a sample message to the device. The DUT must set the Configuration Changed Flag.

```
SEND Command 17 (Primary) with message = 24 bytes of 0x00
CALL VerifyResponseAndByteCount(0, 26)
```

User must put device in write protect.

```
Prompt user: "Place DUT into Write Protect"
```

Verify we are in Write Protect

```
SEND Command 15
IF (UNIV_REVISION >= 6)
   THEN CALL VerifyResponseAndByteCount(0, 20)
ELSE
   THEN CALL VerifyResponseAndByteCount(0, 19)
END IF
CALL TestValidFrame
IF (DUT is NOT in "Write Protect")
   THEN Test result is FAIL                                     (2720)
END IF
```

Clear Configuration Change flag while in write protect.

```
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                    (2721)
END IF
```

Use Primary Master Command 38 is sent with no data bytes to reset configuration change flag.

```
SEND Command 38 (Primary)
IF (UNIV_REVISION > 6) THEN
   CALL VerifyResponseAndByteCount(0, 4)
   IF (DEVICE_STATUS == "Configuration Changed")
      THEN Test Result is FAIL                                 (2722)
   END IF
ELSE
   IF (RESPONSE_CODE != 0)
      THEN PRINT "WARNING: It is recommended practice that Command 38 allow
      the reset configuration change flag in write protect mode."
   END IF
   IF (BYTE_COUNT != 2)
      THEN Test Result is FAIL                                 (2723)
   END IF
END IF
```

Try to change the poll address

```
IF ( UNIV_REVISION >= 6 ) THEN
   SEND Command 7 to read pollAddr, lcMode
   CALL TestValidFrame
   CALL VerifyResponseAndByteCount(0, 4)

   SEND Command 6 (with poll address= (pollAddr XOR 0x3F)
   CALL TestValidFrame
   CALL VerifyResponseAndByteCount("In Write Protect Mode", 2)
   SEND Command 7 to read pollAddr0
   CALL TestValidFrame
   CALL VerifyResponseAndByteCount(0, 4)
   IF (pollAddr0 != pollAddr)
      THEN Test result is FAIL                                 (2755)
   END IF
```

Try to change the Loop Current mode

```
   SEND Command 6 (with loop current mode = (!lcMode)
   CALL TestValidFrame
   CALL VerifyResponseAndByteCount("In Write Protect Mode", 2)

   SEND Command 7 to read pollAddr0, lcMode0
   CALL TestValidFrame
   CALL VerifyResponseAndByteCount(0, 4)
   IF (lcMode0 != lcMode)
      THEN Test result is FAIL                                 (2756)
   END IF
END IF
```

Try to change message contents.

```
SEND Command 12 to read msg
CALL VerifyResponseAndByteCount(0, 26)
CALL TestValidFrame

SEND Command 17 (with (msg XOR 0xFF's) )
CALL VerifyResponseAndByteCount("In Write Protect Mode", 2)
CALL TestValidFrame

CALL IssueCommand12(to read msg0, 2785)                          (2785)
CALL VerifyResponseAndByteCount(0, 26)
CALL TestValidFrame
IF (msg0 != msg)
   THEN Test result is FAIL                                      (2790)
END IF
```

Try to change tag, descriptor date.

```
SEND Command 13 to read tdd
CALL VerifyResponseAndByteCount(0, 23)
CALL TestValidFrame

SEND Command 18 (with (tdd XOR 0xFF's) )
CALL VerifyResponseAndByteCount("In Write Protect Mode", 2)
CALL TestValidFrame

CALL IssueCommand13(to read tdd0, 2800)                          (2800)
CALL VerifyResponseAndByteCount(0, 23)
CALL TestValidFrame
IF (tdd0 != tdd)
   THEN Test result is FAIL                                      (2825)
END IF
```

Try to change assembly number.

```
SEND Command 16 to read fan
CALL VerifyResponseAndByteCount(0, 5)
CALL TestValidFrame

SEND Command 19 (with (fan XOR 0xFF's) )
CALL VerifyResponseAndByteCount("In Write Protect Mode", 2)
CALL TestValidFrame

CALL IssueCommand16( fan0, 2840)                                 (2840)
CALL VerifyResponseAndByteCount(0, 5)
CALL TestValidFrame
IF (fan0 != fan)
   THEN Test result is FAIL                                      (2860)
END IF
```

Try to change long tag.

```
IF ( UNIV_REVISION >= 6 ) THEN
   SEND Command 20 to read lTag
   CALL VerifyResponseAndByteCount(0, 5)
   CALL TestValidFrame

   SEND Command 22 (with (lTag XOR 0xFF's) )
   CALL VerifyResponseAndByteCount("In Write Protect Mode", 2)
   CALL TestValidFrame

   CALL IssueCommand20(lTag0, 2880)                                (2880)
   CALL VerifyResponseAndByteCount(0, 5)
   CALL TestValidFrame
   IF (lTag0 != lTag)
      THEN Test result is FAIL                                     (2895)
   END IF
END IF

Prompt user: "Take DUT out of Write Protect"

IF ( UNIV_REVISION >= 6 )
   SEND Command 0 to read the = configuration
      changed counter (cfgCntr0)
   IF (cfgCntr != cfgCntr0)
      THEN Test result is FAIL                                     (2897)
   END IF
END IF
```

Can we reset the Additional Device Status?

```
SEND Command 48 with no data
IF (RESPONSE_CODE == "Success")
   SEND Command 48 with Command48 response bits
   CALL TestValidFrame()

   SWITCH on (RESPONSE_CODE)

         CASE "Success"

         CASE "In Write Protect"
            Test result is FAIL                                    (2900)

         CASE DEFAULT
            Test result is FAIL                                    (2901)
   END SWITCH
END IF
END TEST
```

7.11 UAL010 Verify Cold Start Bit

Checks for the correct identification of cold starts. The user is prompted to cycle power to the DUT and the cold start bit is checked. Checks that the configuration change counter is retained through the power cycle.

References:

Specification	Rev.	Sections
Command Summary Specification	8.0	7.4.3
Universal Command Specification	5.2	

Test Procedure

```
CALL IdentifyDevice
CALL VerifyNotWriteProtected()
IF ( UNIV_REVISION >= 6 ) THEN
   SEND Command 0 to read the configuration changed counter (cfgCntr)
   IF (cfgCntr == 0)
      CALL IssueCommand12 (to read msg0, 3005)                    (3005)
      Create message1 with 24 bytes of 0x00
      SEND Command 17 with message1
      CALL TestValidFrame
      CALL VerifyResponseAndByteCount(0, 26)

      CALL IssueCommand12 (to read msg, 3020)                     (3020)
      Create message2 with 24 bytes of 0xCC
      SEND Command 17 with message2
      CALL TestValidFrame
      CALL VerifyResponseAndByteCount(0, 26)

      CALL IssueCommand12 (msg, 3040)                             (3040)
      SEND Command 17 with msg0
      CALL TestValidFrame
      CALL VerifyResponseAndByteCount(0, 26)
      CALL IssueCommand12 (msg, 3055)                             (3055)
      SEND Command 0 to read the configuration
         changed counter (cfgCntr)
   END IF
END IF

PROMPT: "Please remove power to the DUT.\n Do not re-apply
   power yet!"
```

Verify communications loss

```
SET Master = Secondary
SEND Command 0
IF COMMUNICATIONS_ERROR != "No Response")
   THEN Test Result is FAIL                                       (3060)
END IF
```

Power it back up. "Cold Start" must be set for both Primary and Secondary Master one command read only. First it must be set

```
PROMPT: "Please apply power to the DUT."
DO
   SEND Command 0
WHILE (COMMUNICATIONS_ERROR == "No Response")
IF (UNIV_REVISION >= 6)
   THEN CALL VerifyResponseAndByteCount(0, 19)
END IF
IF (UNIV_REVISION == 5)
   THEN CALL VerifyResponseAndByteCount(0, 14)
END IF
CALL TestValidFrame
IF (DEVICE_STATUS != "Cold Start")
   THEN Test Result is FAIL                                          (3075)
END IF
```

Now it must be reset

```
Send Command 0
IF (UNIV_REVISION >= 6)
   THEN CALL VerifyResponseAndByteCount(0, 19)
END IF
IF (UNIV_REVISION == 5)
   THEN CALL VerifyResponseAndByteCount(0, 14)
END IF
CALL TestValidFrame
IF (DEVICE_STATUS == "Cold Start")
   THEN Test Result is FAIL                                          (3090)
END IF
```

Check the configuration change counter

```
IF ( UNIV_REVISION >= 6 ) THEN
   IF (configuration changed counter != cfgCntr)
      THEN Test Result is FAIL                                       (3095)
   END IF
END IF
```

Check the other master

```
SET Master = Primary
Send Command 0
IF (UNIV_REVISION >= 6)
   THEN CALL VerifyResponseAndByteCount(0, 19)
ELSE
   THEN CALL VerifyResponseAndByteCount(0, 14)
END IF
CALL TestValidFrame
IF (DEVICE_STATUS != "Cold Start")
   THEN IF (UNIV_REVISION >= 6)
      THEN Test Result is FAIL                                    (3110)
   ELSE
      WARNING: "It is recommended practice to maintain cold
         start bits for both masters."
   ENDIF
END IF

Send Command 0
IF (UNIV_REVISION >= 6)
   THEN CALL VerifyResponseAndByteCount(0, 19)
ELSE
   THEN CALL VerifyResponseAndByteCount(0, 14)
END IF
CALL TestValidFrame
IF (DEVICE_STATUS == "Cold Start")
   THEN Test Result is FAIL                                       (3125)
END IF

IF (UNIV_REVISION >= 6)
   IF (configuration changed counter != cfgCntr)
      THEN Test Result is FAIL                                    (3130)
   END IF
END IF

END TEST
```

7.12 UAL011 Read Device Variables (Command 9)

Checks operation of command 9. Issue command 0 to determine the number of variables supported by the device. Then issue command 9 for 0 to 8 device variables, using device variable code 0 and checking the error response for each.

For HART 7 and later devices, verify that the time stamp increments during the test.

Table 4 Universal Command 9 Response Data Bytes for Varying Device Variables

Number of Device Variables Requested	Byte Count (BC)	
	HART 6	HART 7
1	11	15
2	19	23
3	27	31
4	35	39
5	35	47
6	35	55
7	35	63
8	35	71

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.10

Test Case A: Checking for Supported Device Variables

```
CALL IdentifyDevice
IF UNIV_REVISION < 6_THEN
   SEND Command 9 with no data bytes

   IF RESPONSE CODE != Not Implemented
      THEN Test result is FAIL                                    (3211)
   ELSE
      Abort (Test is not applicable to this device)               (5001)
   END IF
END IF
```

If the DUT does not expose any Device Variables then it returns PV, SV, TV, and QV

```
SEND Command 0 to read the = maxDeviceVars.
IF (maxDeviceVars = 0)
   THEN SET maxDeviceVars = 3
END IF
```

Find a supported Device Variable

```
dVar = -1
DO
   SET dVarFound = FALSE;
   DO
      INCREMENT dVar
      SEND Command 9 with one byte = dVar
      CALL TestValidFrame

      IF ( (RESPONSE_CODE == "Invalid Selection")
         IF (BYTE_COUNT != 2) )
            THEN Test result is FAIL                                    (3210)
         END IF
      ELSE IF ( (RESPONSE_CODE != "Update Failure")
                AND (RESPONSE_CODE != 0 ))
         THEN Test result is FAIL                                       (3220)
      ELSE IF (((BYTE_COUNT != 11) AND (UNIV_REVISION == 6))
            OR ((BYTE_COUNT != 15) AND (UNIV_REVISION > 6)))
         THEN Test result is FAIL                                       (3225)
```

If we get a NaN response make sure all the other fields are set correctly

```
      ELSE IF (dVar.Value == "7F A0 00 00"(NaN) THEN
         IF (dVar.Units != 250)
            THEN Test result is FAIL                                    (3226)
         END IF
         IF (dVar.Status != 0x30)
            THEN Test result is FAIL                                    (3227)
         END IF
         IF (dVar.Class != 0)
            THEN Test result is FAIL                                    (3228)
         END IF
         IF ( (RESPONSE_CODE != 0)
            THEN Test result is FAIL                                    (3224)
         END IF
```

Response is "Success" or "Update Failure", not a NaN, and the right Byte Count. I think we have it!

```
      ELSE
         dVarFound == TRUE:
      END IF
   WHILE ( (dVar < 239) AND (!dVarFound) )
```

We are out of the Do-Loop (either exhausted Device Variables or we found one) If we found one, check the classification for validity

```
   IF (dVarFound) THEN
      IF (slot 0 variable classification == [1-63 or 240-255] )
         THEN Test result is FAIL                                       (3230)
      END IF
```

Is this a legal Device Variable ?

```
      IF (dVar > maxDeviceVars)
         THEN Test result is FAIL                                       (3235)
      END IF
```

Send Command 9 with varying number of data bytes. Check command response length

```
            IF UNIV_REVISION > 6 THEN
               maxCnt = 8
               length[] = {23, 31, 39, 47, 55, 63, 71, 71}
               nBytes[] = { 2,  3,  4,  5,  6,  7,  8,  9}
            ELSE
               maxCnt = 4
               length[] = {19, 27, 35, 35}
               nBytes[] = { 2,  3,  4,  5}
            END IF

            FOR ( iter = [0 to maxCnt] )
               SEND Command 9 with nBytes[iter] of dVar
               CALL TestValidFrame

               SWITCH on RESPONSE_CODE

               CASE RESPONSE_CODE == 0
                  IF (BYTE_COUNT != length[iter])
                     THEN Test result is FAIL                          (3251)
                  END IF

               CASE RESPONSE_CODE == "Update Failure"
                  IF (BYTE_COUNT != length[iter])
                     THEN Test result is FAIL                          (3252)
                  END IF

               CASE RESPONSE_CODE == "Command Response Truncated"
                  IF (nBytes < 5) OR (UNIV_REVISION < 7)
                     THEN Test result is FAIL                          (3253)
                  END IF

                  IF (BYTE_COUNT != 39])
                     THEN Test result is FAIL                          (3254)
                  END IF

               CASE DEFAULT
                  THEN Test result is FAIL                             (3256)
               END IF
            END FOR

         END IF
      WHILE (dVar < 239)
```

Check "Invalid Selection" Response Code

```
      SEND Command 9 with 4 data bytes of 0xFF
      CALL VerifyResponseAndByteCount("Invalid Selection", 2)
      CALL TestValidFrame
      END TEST CASE
```

Test Case B: Checking Required Device Variables

```
      CALL IdentifyDevice
      IF UNIV_REVISION < 7 THEN
         Abort (Test is not applicable to this device)                 (5001)
      END IF
```

Read the mandatory device variables plus a couple more

```
deviceVars = [242, 243, 244, 245, 246, 247, 248, 249]
SEND Command 9 with deviceVars
CALL TestValidFrame
IF ( (RESPONSE_CODE != 0) AND
   (RESPONSE_CODE != "Update Failure") AND
   (RESPONSE_CODE != "Command Response Truncated"))
      THEN Test result is FAIL                                    (3245)
END IF
```

Did we get them all first try? If not send an extra command 9

```
IF (BYTE_COUNT < 71) THEN
   SWITCH on BYTE_COUNT
      CASE 39
         deviceVars = [246, 247, 248, 249]
         bc1 = 39
      CASE 47
         deviceVars = [247, 248, 249]
         bc1 = 31
      CASE 55
         deviceVars = [248, 249]
         bc1 = 23
      CASE 63
         deviceVars = [249]
         bc1 = 15
      CASE DEFAULT
         Test result is FAIL                                      (3257)
   END SWITCH

   SEND Command 9 with deviceVars
   CALL TestValidFrame
   IF ( (RESPONSE_CODE != 0) AND (RESPONSE_CODE != "Update Failure") )
      THEN Test result is FAIL                                    (3246)
   END IF
   IF (BYTE_COUNT != bc1) THEN
      THEN Test result is FAIL                                    (3247)
   END IF

END IF
```

Save values of variables for future comparison.

```
Battery = Device Variable 243 value
PctRange0 = Device Variable 244 value
loopCurrent0 = Device Variable 245 value

pv0 = Device Variable 246 value
sv0 = Device Variable 247 value
tv0 = Device Variable 248 value
qv0 = Device Variable 249 value

pvUnit0 = Device Variable 246 unit code
svUnit0 = Device Variable 247 unit code
tvUnit0 = Device Variable 248 unit code
qvUnit0 = Device Variable 249 unit code

dTime = Time Stamp from Command 9

IF (command777 = battery) THEN
```

```
   IF Battery != NaN
      THEN Test result is FAIL                                    (3248)
   END IF
END IF
IF Device Variable 242 value!= NaN
   THEN Test result is FAIL                                       (3249)
END IF
```

Use command 3 to read the device variables for PV, SV, TV, and QV for comparison to command 9 response.

```
SEND Command 2
LoopCurrent2 = loop current value from Command 2
PctRange2 = percent range from Command 2

IF (PROFILE > 128) AND (loopCurrent == NAN)
   THEN "WARNING: Wireless product without a loop current".
ELSE IF (loopCurrent2 != loopCurrent0)
   THEN Test result is FAIL                                       (3258)
END IF

IF (PctRange2 != PctRange0)
   THEN Test result is FAIL                                       (3260)
END IF


SEND Command 3

pv3 = PV value from Command 3
pvUnit3 = unit code for PV from Command 3

sv3 = SV value from Command 3
svUnit3 = unit code for SV from Command 3

tv3 = TV value from Command 3
tvUnit3 = unit code for TV from Command 3

qv3 = QV value from Command 3
qvUnit3 = unit code for QV from Command 3

IF (pv0 != pv3)
   THEN Test result is FAIL                                       (3259)
END IF

IF (pvUnit0 != pvUnit3)
   THEN Test result is FAIL                                       (3261)
END IF

IF (sv0!=sv3)
   THEN Test result is FAIL                                       (3262)
END IF

IF (svUnit0 != svUnit3)
   THEN Test result is FAIL                                       (3263)
END IF
```

```
IF (tv0 != tv3)
   THEN Test result is FAIL                                    (3264)
END IF

IF (tvUnit0 != tvUnit3)
   THEN Test result is FAIL                                    (3266)
END IF

IF (qv0 != qv3)
   THEN Test result is FAIL                                    (3267)
END IF

IF (qvUnit0 != qvUnit3)
   THEN Test result is FAIL                                    (3268)
END IF
```

Verify Byte Count for single variable request.

```
SEND Command 9 with Device Variables = 246
IF ((BYTE_COUNT != 15)
   THEN Test result is FAIL                                    (3269)
END IF
```

Check the last 4 bytes of command 9 are the Data Time Stamp and the time increments during the testcase.

```
dTime2 = Time Stamp from Command 9

IF dTime2 <= dTime1
   THEN Test result is FAIL                                    (3270)
END IF
```

Verify whether non-wireless device supports real time clock.

```
SEND Command 90 with no data bytes

IF (RESPONSE CODE != 0) AND (DEVICE_PROFILE < 128)
   THEN WARNING: It is recommended that all devices support
              a configurable real-time clock.                  (3271)

ELSE IF WIRELESS
   THEN Test result is FAIL                                    (3272)
END IF

SEND Command 89 with Time Set Code 0
IF (RESPONSE CODE == Command Not Implemented) AND (DEVICE_PROFILE < 128)
   THEN WARNING: It is recommended that all devices
         support a real-time clock.                           (3273)
END IF

END TEST CASE
```

7.13 UAL012 Read Dynamic Variable Classification

Tests command 8, Dynamic Variable Classification. Issue command 8 to read the dynamic variable classification codes for the device. Each classification code must be 250 or one of the device classification codes. Command 8 should have a Response Code of 0.

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.4, 6.9

Test Procedure

```
CALL IdentifyDevice
IF UNIV__REVISION < 6
   THEN Abort Test (i.e., test is not applicable to this device)     (5001)
END IF
```

Send Command 3 to see which Dynamic Variable are supported and their unit codes.

```
SEND Command 3
CALL TestValidFrame
IF ( (RESPONSE_CODE != 0) AND (RESPONSE_CODE != "Update Failure") )
   THEN Test result is FAIL                                          (3300)
END IF
IF (BYTE_COUNT is illegal)
   THEN Test result is FAIL                                          (3301)
```

Read and check the dynamic variable classification codes for the device.

```
SEND Command 8
CALL VerifyResponseAndByteCount(0, 6)
CALL TestValidFrame
```

check PV, it is always there

```
IF (PV classification == [1-63] or 250-255)
   THEN Test result is FAIL                                          (3305)
END IF
IF ( (PV classification == 0) AND (PVUnits > 169) AND (PVUnits < 220) )
   THEN Test result is FAIL                                          (3310)
END IF
IF (PVUnits == [250-255] )
   THEN Test result is FAIL                                          (3315)
END IF
```

Check SV

```
IF (BYTE_COUNT == [ 16, 21 or 26] ) THEN
   IF (SV classification == [1-63] or 240-255)
      THEN Test result is FAIL                                          (3320)
   END IF
   IF ( (SV classification == 0)
                    AND (SVUnits > 169) AND (SVUnits < 220) )
      THEN Test result is FAIL                                          (3325)
   END IF
   IF (SVUnits == [250-255] )
      THEN Test result is FAIL                                          (3330)
   END IF
ELSE
   IF (SV classification != 250)
      THEN Test result is FAIL                                          (3335)
   END IF
END IF
```

Check TV

```
IF (BYTE_COUNT == [21 or 26] ) THEN
   IF (TV classification == [1-63] or 240-255)
      THEN Test result is FAIL                                          (3340)
   END IF
   IF ( (TV classification == 0)
                    AND (TVUnits > 169) AND (TVUnits < 220) )
      THEN Test result is FAIL                                          (3345)
   END IF
   IF (TVUnits == [250-255] )
      THEN Test result is FAIL                                          (3350)
   END IF
ELSE
   IF (TV classification != 250)
      THEN Test result is FAIL                                          (3355)
   END IF
END IF
```

Check QV

```
IF (BYTE_COUNT == 26 ) THEN
   IF (QV classification == [1-63] or 240-255)
      THEN Test result is FAIL                                          (3360)
   END IF
   IF ( (QV classification == 0)
                    AND (QVUnits > 169) AND (QVUnits < 220) )
      THEN Test result is FAIL                                          (3365)
   END IF
   IF (QVUnits == [250-255] )
      THEN Test result is FAIL                                          (3370)
   END IF
ELSE
   IF (QV classification != 250)
      THEN Test result is FAIL                                          (3375)
   END IF
END IF
END TEST
```

7.14 UAL013 Write Long Tag

Verifies long tag write. Read the current long tag using command 20. Save this original tag for later use. Issue command 22 to write a new long tag. Issue command 20 again to ensure that the tag has changed to the new value. Test will proceed with the following sequence:

1. Valid long tag with 32 bytes of the character 'A' (Success expected).
2. Valid long tag with 33 bytes of the character 'r', one byte too long (Success expected).
3. Invalid tag of 31 bytes of the character 'f', one byte short (must produce error Response Code).
4. Valid original tag of 32 bytes (Success expected).

References:

Specification	Rev.	Sections
Universal Command Specification	6.0	6.20, 6.21

Test Procedure

```
CALL IdentifyDevice
IF UNIV__REVISION < 6
   THEN Abort Test (i.e., test is not applicable to this device)     (5001)
END IF

CALL VerifyNotWriteProtected()
CALL IssueCommand20 (to read tag0, 3400)                             (3400)
SEND Command 0 to read the configuration changed counter (cfgCntr)
```

Send Command 22 with a valid long tag. Slave should reply to command 20 with the same message.

```
Create tag1 with 32 bytes of 'A'
CALL IssueCommand22(tag1, 3405)                                      (3405)
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 34)
SET cfgCntr = cfgCntr + 1
CALL VailidateLongTag(tag1, cfgCntr, 3410)                           (3410)
```

Send Command 22 with a changed valid message + one byte. Slave must reply to command 20 with the new message, but with exactly 32 bytes.

```
Create tag2 with 32 bytes of 'r'
CALL IssueCommand22(tag2, 3415) with one extra byte                  (3415)
CALL TestValidFrame
IF ( (COMMUNICATION_ERROR == "Buffer Overflow")
   IF (BYTE_COUNT != 2) )
      THEN Test result is FAIL                                       (3416)
   END IF
```

```
ELSE
   CALL VerifyResponseAndByteCount(0, 34)
   SET cfgCntr = cfgCntr + 1
   CALL ValidateLongTag(tag2, cfgCntr, 3420)                    (3420-3422)
END IF
```

Send Command 22 with a 31 byte tag. Slave should not store the invalid message and should reply to command 22 with "Too Few Data Bytes Received". Slave must reply to command 20 with 32 bytes of 'r'.

```
Create tag3 with 31 bytes of 'f'
CALL IssueCommand22(tag3, 3425)                                    (3425)
CALL TestValidFrame
CALL VerifyResponseAndByteCount("Too Few Data Bytes Received",2)
CALL ValidateLongTag(tag2, cfgCntr, 3430)                         (3430-3432)
```

Restore original tag

```
CALL IssueCommand22(tag0, 3435)                                    (3435)
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 34)
SET cfgCntr = cfgCntr + 1
CALL ValidLongTag(tag0, cfgCntr, 3440)                            (3440-3442)

END TEST
```

ValidateLongTag (lTag, cfgCntr, failurePoint)

This procedure is unique to UAL014. It uses command 20 to read the long tag and compare it to "lTag".

```
PROCEDURE ValidateLongTag(lTag, cfgCntr, failurePoint)

SEND Command 21 (lTag)
CALL TestValidFrame
CALL VerifyResponseAndByteCount(0, 19)

CALL IssueCommand20 (tag, failurePoint)                        (failurePoint)
CALL VerifyResponseAndByteCount(0, 34)
CALL TestValidFrame
IF (lTag != tag)
   THEN Test result is FAIL                                   (failurePoint+1)
END IF

SEND Command 0 to read the = configuration
   changed counter (cfgCntr0)
IF (cfgCntr != cfgCntr0)
   THEN Test result is FAIL                                   (failurePoint+2)
END IF

PROCEDURE END
```

IssueCommand22 (lTag, failurePoint)

Issue command 22 taking care of "Busy" and Delayed Responses.

```
PROCEDURE IssueCommand22(lTag, failurePoint)
DO
   SEND Command 1
   SEND Command 22 (with lTag)
   CALL TestValidFrame()
   IF ( ( (RESPONSE_CODE == "Delayed Response Initiated")
            OR (RESPONSE_CODE == "Delayed Response Running") )
            AND (BYTE_COUNT != 2) )
      THEN Test result is FAIL                              (failurePoint)
   END IF
WHILE ( (RESPONSE_CODE == "Busy")
   OR (RESPONSE_CODE == "Delayed Response Initiated")
   OR (RESPONSE_CODE == "Delayed Response Running") )

RETURN RESPONSE_CODE

PROCEDURE END
```

7.15 UAL038 Reset Configuration Changed Flag

Checks Addresses, Command Number, Response Code and Byte Count for Command 38, Reset Configuration Changed Flag. This command is a universal command in revisions after HART 6.

When a write command is successfully issued to a device, the Configuration Changed Flag must be set to notify the masters that the Field Device's database has changed. In HART 6 and later, each master has a configuration change flag bit. As a result, the Primary Master issuing command 38 does not reset the Secondary's bit.

The DUT must start the test with both bits reset.

The message sequence used is as follows:

1. Command 17 (Write Message) is sent to write a sample message to the device. The Response Code is monitored to verify that the Response Code to the Command is 0 (Success). This will cause the Configuration Changed Flag to be set in the device.
2. A secondary master Command 38 is sent.
3. Command 17 (Write Message) is sent again.
4. A primary master Command 38 is sent.

The process is repeated to verify the primary does not interfere with the Secondary's bit

References:

Specification	Rev.	Sections
Universal Command Specification	7.0	6.23

Test Case A: Checking without sending the Configuration Changed Counter

```
CALL IdentifyDevice

SWITCH on (UNIV_REVISION)

CASE 5
   SET Cmd38BC = 2
CASE 6
   SET Cmd38BC = 2
CASE DEFAULT
   SET Cmd38BC = 4

END SWITCH

SEND Command 38 with no data bytes
CALL TestValidFrame ()
```

```
IF (RESPONSE_CODE != "Success")
   IF (UNIV_REVISION >= 7)
      THEN Test result is FAIL                                    (3470)
   ELSE
      Abort (Test is not applicable to this device)               (5001)
   END IF
END IF

CALL VerifyNotWriteProtected()
IF UNIV_REVISION >= 6
   SEND Command 0 (Primary)to read the
      configuration changed counter (cfgCntr)
   IF (DEVICE_STATUS == "Configuration Changed")
      SEND Command 38 (Primary) with no data bytes
      SEND Command 38 (Secondary) with no data bytes
      IF (DEVICE_STATUS == "Configuration Changed")
         THEN Test result is FAIL                                 (3471)
      END IF
   END IF
END IF
```

Command 17 (Write Message) is sent to write a sample message to the device. The DUT must set the Configuration Changed Flag.

```
SEND Command 17 (Primary) with message = 24 bytes of 0x00
CALL VerifyResponseAndByteCount(0, 26)
```

Cycle the power. Both configuration changed bits must stay set

```
Prompt user: "Cycle Power on the DUT"
DO
   SEND Command 0 (Primary)
WHILE (COMMUNICATIONS_ERROR != "No Response")
```

Now wait for the device to power back up

```
DO
   SEND Command 0 (Primary)
WHILE (COMMUNICATIONS_ERROR == "No Response")

IF (UNIV_REVISION >= 6) THEN
   IF (configuration changed counter != cfgCntr + 1)
      THEN Test Result is FAIL                                    (3472)
   END IF
END IF

IF (DEVICE_STATUS != "Cold Start")
   THEN Test Result is FAIL                                       (3473)
END IF
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                       (3474)
END IF

SEND Command 0 (Secondary)

IF (UNIV_REVISION >= 6) THEN
   IF (configuration changed counter != cfgCntr + 1)
      THEN Test Result is FAIL                                    (3475)
   END IF
END IF
```

```
IF (DEVICE_STATUS != "Cold Start")
   THEN Test Result is FAIL                                    (3476)
END IF
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                    (3477)
END IF
```

A secondary master Command 38 is sent.

```
SEND Command 38 (Secondary)
CALL VerifyResponseAndByteCount(0, Cmd38BC)

IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3478)
END IF
```

Make sure the Primary's bit is still set.

```
SEND Command 0 (Primary)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN IF (UNIV_REVISION >= 6)
      THEN Test Result is FAIL                                 (3479)
   ELSE
      WARNING: "It is recommended practice to maintain both
         cold start bits"
   END IF
END IF
```

A Primary Master Command 38 is sent.

```
SEND Command 38 (Primary)
CALL VerifyResponseAndByteCount(0, Cmd38BC)

IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3480)
END IF
```

Verified Secondary does not reset primary. Now try primary resetting the secondary. Command 17 (Write Message) is sent to write a sample message to the device. The DUT must set the Configuration Changed Flag.

```
SEND Command 17 (Secondary) with message = 24 bytes of 0xFF
CALL VerifyResponseAndByteCount(0, 26)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                    (3481)
END IF

SEND Command 0 (Primary)
IF (UNIV_REVISION >= 6) THEN
   IF (configuration changed counter != cfgCntr + 2)
      THEN Test Result is FAIL                                 (3482)
   END IF
END IF
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                    (3483)
END IF
```

A Primary master Command 38 is sent.

```
SEND Command 38 (Primary)
CALL VerifyResponseAndByteCount(0, Cmd38BC)

IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3484)
END IF
```

Make sure the Secondary's bit is still set..

```
SEND Command 0 (Secondary)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN IF (UNIV_REVISION >= 6)
      THEN Test Result is FAIL                                 (3485)
   ELSE
      WARNING: "It is recommended practice to maintain both
         cold start bits"
   END IF
END IF
IF (UNIV_REVISION >= 6) THEN
   IF (configuration changed counter != cfgCntr + 2)
      THEN Test Result is FAIL                                 (3486)
   END IF
END IF
```

A Secondary Master Command 38 is sent.

```
SEND Command 38
CALL VerifyResponseAndByteCount(0, Cmd38BC)

IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3487)
END IF

END TEST CASE
```

Test Case B: Checking with the Configuration Changed Counter

This test case verifies proper operationg matching and not matching configuration change counter values.

```
CALL IdentifyDevice

IF (UNIV_REVISION < 7)
   Abort (Test is not applicable to this device)                    (5001)
END IF

CALL VerifyNotWriteProtected()
SEND Command 0 (Primary)to read the configuration changed counter (cfgCntr)
IF (DEVICE_STATUS == "Configuration Changed")
   SEND Command 38 (Primary) with no data bytes
   SEND Command 38 (Secondary) with no data bytes
END IF
```

Command 17 (Write Message) is sent to write a sample message to the device. The DUT must set the Configuration Changed Flag.

```
SEND Command 17 (Primary) with message = 24 bytes of 0x00
CALL VerifyResponseAndByteCount(0, 26)
```

A Command 38 is sent with a single data byte - must return "Too Few Data Bytes Received".

```
SEND Command 38 with one byte of the configuration change counter
CALL VerifyResponseAndByteCount("Too Few Data Bytes Received", 2)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                         (3502)
END IF
```

A Primary Master Command 38 is sent without proper matching configuration change counter.

```
SEND Command 38 (Primary)with configuration change counter-1
CALL VerifyResponseAndByteCount(9, 2)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                         (3495)
END IF
```

A Primary Master Command 38 is sent with configuration change data.

```
SEND Command 38 (Primary) with configuration change counter
CALL VerifyResponseAndByteCount(0, 4)
IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                         (3496)
END IF
```

Make sure the Secondary's bit is set..

```
SEND Command 0 (Secondary)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                         (3497)
END IF
```

A Secondary Master Command 38 is sent without proper matching configuration change counter.

```
SEND Command 38 (Secondary)with configuration change counter-1
CALL VerifyResponseAndByteCount(9, 2)
IF (DEVICE_STATUS != "Configuration Changed")
   THEN Test Result is FAIL                                    (3498)
END IF
```

A Secondary Master Command 38 is sent with configuration change counter.

```
SEND Command 38 (Secondary) with configuration change counter
CALL VerifyResponseAndByteCount(0, 4)
```

Make sure the secondary's bit was cleared.

```
SEND Command 0 (Secondary)
IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3499)
END IF

SEND Command 17 (Secondary) with message = 24 bytes of 0xFF
CALL VerifyResponseAndByteCount(0, 26)
```

A Primary Master Command 38 is sent with configuration change counter.

```
SEND Command 38 (Primary) with configuration change counter
CALL VerifyResponseAndByteCount(0, 4)
SEND Command 38 (Secondary) with configuration change counter
CALL VerifyResponseAndByteCount(0, 4)
```

Make sure both bits are cleared.

```
SEND Command 0 (Primary)
IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3500)
END IF
SEND Command 0 (Secondary)
IF (DEVICE_STATUS == "Configuration Changed")
   THEN Test Result is FAIL                                    (3501)
END IF

END TEST CASE

END TEST
```

7.16 UAL048 Read Additional Device Status

For HART 7 Command 48 is a Universal Command and must be implemented.

Verifies the DUT responds properly to Command 48. Checks Addresses, Command Number, Response Code and Byte Count for Command 48, Read Additional Device Status. This command is a universal command in revisions after HART 6.

This read command may only return Response Codes indicating success (0) or an update in progress (8). Every device may implement the command with its own unique byte count. Thus this test can only send a series of Command 48 requests and compare the byte count in the response for consistency and possibly detect a Response Code 8. If Response Code 8 (warning) is returned then the data must also be returned. Thus the test will send ten Command 48 requests as rapidly as is possible for the host and the results examined.

The generation of "More Status Available" is device specific. The test system expects the device to generate More Status Available during the test. The methodology used to generate of More Status Available must be documented and attached to the test results. This methodology should be repeatable by the HCF should this test be audited during Device Registration.

References:

Specification	Rev.	Sections
Universal Command Specification	7.0	6.24

Test Case A: Basic checking of Command 48

```
CALL IdentifyDevice

SEND Command 48 with no data bytes
IF (RESPONSE_CODE = "Command not Implemented")
   IF (UNIV_REVISION > 6)
      THEN Test Result is FAIL                                    (3530)
   ELSE
      PRINT "Warning, Implementation of Command 48 is strongly
         recommended.  This command is implemented by most
         Field Devices and widely used in Host Applications."
   END IF
END IF
```

Read device status – byte counts for messages returned with Response Code 8 should be the same

```
DO
   SEND Command 48 with no data bytes
   CALL TestValidFrame
WHILE (RESPONSE_CODE != 0)

IF (UNIV_REVISION == 7) THEN
   IF (BYTE_COUNT != [11, 12, 14-27]
      THEN Test Result is FAIL                                    (3531)
   END IF
ELSE IF (UNIV_REVISION = 6) THEN
   IF (BYTE_COUNT != [10, 16-27]
      THEN Test Result is FAIL                                    (3532)
   END IF
ELSE IF (UNIV_REVISION = 5)
   IF (BYTE_COUNT != [3-8, 16-27]
      THEN Test Result is FAIL                                    (3533)
   END IF
END IF
```

Check 10 Command 48 response for consistent behavior.

```
SET Cmd48bc = BYTE_COUNT
FOR (n = [1-10])
   SEND Command 48 with no data bytes
   CALL TestValidFrame
   IF (RESPONSE_CODE == "SUCCESS")
      IF (BYTE_COUNT != Cmd48bc
         THEN Test Result is FAIL                                 (3434)
      END IF

      IF (UNIV_REVISION >= 6) THEN
         SET Cmd48eds = Extended Device Status
         SEND Command 0
         IF (Cmd48eds != Extended Device Status)
            THEN Test Result is FAIL                              (3536)
         END IF
      END IF
   ELSE IF (RESPONSE_CODE == "Update In Progress")
      IF (BYTE_COUNT != Cmd48bc
         THEN Test Result is FAIL                                 (3537)
      END IF
   ELSE
      Test Result is FAIL                                         (3538)
   END IF
END FOR

END TEST CASE
```

Test Case B: Testing the clearing of the "More Status Available" bit

```
CALL IdentifyDevice
SEND Command 48
IF (RESPONSE_CODE = "Command not Implemented")
   IF (UNIV_REVISION > 6)
      THEN Test Result is FAIL                                    (3539)
   END IF
ELSE
   Abort (Test is not applicable to this device)                  (5001)
END IF
DO
   SEND Command 48
   IF (Device Status != More Status Available) THEN
      PRINT: PLEASE alter device conditions to ensure
         "More Status Available" status bit is set.
      Wait for user to continue.
   END IF
While (Device Status != More Status Available)
```

Find the bit pattern set for the command 48 status.

```
bit48[] = 0
Check = 0
FOR (bc = [2 to BYTE_COUNT])
   bit48[bc] = data[bc]
   check = check | data[bc];
END FOR
```

Verify that there is a bit set in command 48 when more status is set in device status.

```
IF (check == 0)
   THEN test result is FAIL                                       (3540)
END IF
```

Verify that if too few data bytes are sent, the more Status Available remains set.

```
SEND Command 48 with 1 databyte (i.e., bit48[0] only)
CALL VerifyResponseAndByteCount("Too Few Data Bytes", 2)
IF (Device Status != More Status Available)
   THEN test result is FAIL                                       (3541)
END IF
```

Clear the more status available and clear command 48 bit that caused the status change.

```
SEND Command 48 with bit48[]

CALL VerifyResponseAndByteCount(0, Cmd48bc)
IF (Device Status == More Status Available)
   THEN test result is FAIL                                       (3542)
END IF

SEND Command 0 (AS OTHER MASTER)
IF (Device Status != More Status Available)
   THEN test result is FAIL                                       (3543)
END IF
SEND Command 48 (AS OTHER MASTER) with bit48[]
CALL VerifyResponseAndByteCount(0, Cmd48bc)
IF (Device Status == More Status Available)
   THEN test result is FAIL                                       (3544)
END IF
END TEST CASE
```

ANNEX A. REUSABLE TEST PROCEDURE DEFINITIONS

The procedures in this appendix are used in two or more of the UAL test definitions. They are presented here as reusable procedures to remove redundancy in the Test Body.

A.1 IdentifyDevice ( )

Identify the device, check its revision, record the number of preambles it desires for later requests and note its unique identifier for later requests.

```
PROCEDURE IdentifyDevice()
SET NUMBER_REQUEST_PREAMBLES to 15
pollAddress = 0, deviceFound = FALSE
While (( pollAddress < 63 ) AND (!deviceFound))
         SEND short frame Command 0 using POLL_ADDRESS = pollAddress
         IF ( COMMUNICATIONS_ERROR == "No Response" )
            THEN increment pollAddress
         ELSE  IF ( COMMUNICATIONS_ERROR OR (RESPONSE_CODE != 0))
            THEN Test result is FAIL                                  (5100)
         ELSE
            deviceFound = TRUE
         END IF
         END WHILE
IF (!deviceFound)
   THEN Test result is FAIL                                           (5101)
END IF

SET NUMBER_REQUEST_PREAMBLES. UNIV_COMMAND_REVISION, POLL_ADDRESS
IF UNIV_REVISION < 5
   THEN Abort Test (i.e., test is not applicable to this device)      (5001)
END IF
IF UNIV_REVISION == 5 AND pollAddress > 15
   THEN FAIL                                                          (5002)
END IF
IF UNIV__REVISION > 7
   THEN Abort Test (i.e., test is not applicable to this device)      (5003)
END IF
PROCEDURE END
```

A.2 VerifyResponseAndByteCount (rc, bc)

Verify that the reply to a command matches list of responses [r] and Byte Count b.

```
PROCEDURE VerifyResponseAndByteCount(r, b)
CALL TestValidFrame()
IF (RESPONSE_CODE != r)
         THEN Test result is FAIL                                     (5110)
END IF
IF (BYTE_COUNT != b)
         THEN Test result is FAIL                                     (5111)
END IF
PROCEDURE END
```

A.3 TestValidFrame ( )

This procedure checks that the DUT replies with the correct information from the command. It compares framing information in a request command and a reply command.

```
PROCEDURE TestValidFrame()
IF reply address does not agree with manufacturer id masked with 0x3f,
   manufacturer device type byte and the three byte ID number
   THEN Test Result is FAIL                                          (5115)
END IF
IF reply Command != request Command
   THEN Test Result is FAIL                                          (5116)
END IF
PROCEDURE END
```

A.4 VerifyNotWriteProtected ( )

The following procedure verifies that the DUT is not in write protect mode.

```
PROCEDURE VerifyNotWriteProtected()
SEND Command 15
DO
         IF (COMMUNICATIONS_ERROR)
            THEN Test result is FAIL                                 (5120)
         END IF
WHILE (RESPONSE_CODE == "Busy" )
```

Note: 251, "Not Used" is a valid response and equivalent to not write protected

```
IF (WRITE_PROTECT_CODE != [ 0, 1, or 251 ] )
   THEN Test result is FAIL                                          (5121)
END IF
IF (DUT is in "Write Protect")
   THEN Test result is FAIL                                          (5122)
END IF
PROCEDURE END
```

A.5 VerifyDate (date)

Verify that date is valid. The date is stored as a 3 byte field with 3 unsigned eight-bit fields: day, month, year – 1900.

```
PROCEDURE VerIFyDate(date)
SET day = first byte of date
SET month = second byte of date
SET year = third byte of date
IF NOT (1 <= day <= 31 AND 1 <= month <= 12)
   THEN Test result is FAIL                                          (5125)
END IF
PROCEDURE END
```

A.6 IssueCommand12 (msg, failurePoint)

Issue command 12 taking care of "Busy".

```
PROCEDURE IssueCommand12(msg, failurePoint)

RETURN RESPONSE_CODE
DO
   SEND Command 1
   SEND Command 12 to read msg
   CALL TestValidFrame()
         IF ( (RESPONSE_CODE == "Busy") AND (BYTE_COUNT != 2) )
            THEN Test result is FAIL                          (failurePoint)
         END IF
WHILE (RESPONSE_CODE == "Busy")

RETURN RESPONSE_CODE, msg

PROCEDURE END
```

A.7 IssueCommand13 (tag, desc, date, failurePoint)

Issue command 13 taking care of "Busy".

```
PROCEDURE IssueCommand13(tag, desc, date, failurePoint )
DO
   SEND Command 1
   SEND Command 13 to read tag, desc, date
   CALL TestValidFrame()
         IF ( (RESPONSE_CODE == "Busy") AND (BYTE_COUNT != 2) )
            THEN Test result is FAIL                          (failurePoint)
         END IF
WHILE (RESPONSE_CODE == "Busy")

RETURN RESPONSE_CODE, tag, desc, date

PROCEDURE END
```

A.8 IssueCommand16 (fan, failurePoint)

Issue command 16 taking care of "Busy".

```
PROCEDURE IssueCommand16(fan, failurePoint)
DO
   SEND Command 1
   SEND Command 16 to read fan
   CALL TestValidFrame()
         IF ( (RESPONSE_CODE == "Busy") AND (BYTE_COUNT != 2) )
            THEN Test result is FAIL                          (failurePoint)
         END IF
WHILE (RESPONSE_CODE == "Busy")

RETURN RESPONSE_CODE, fan

PROCEDURE END
```

A.9 IssueCommand20 (lTag, failurePoint)

Issue command 20 taking care of "Busy".

```
PROCEDURE IssueCommand20(lTag, failurePoint)
DO
   SEND Command 1
   SEND Command 20 to read lTag
   CALL TestValidFrame()
         IF ( (RESPONSE_CODE == "Busy") AND (BYTE_COUNT != 2) )
            THEN Test result is FAIL                          (failurePoint)
         END IF
WHILE (RESPONSE_CODE == "Busy")

RETURN RESPONSE_CODE, lTag

PROCEDURE END
```

ANNEX B. FAILURE POINT CROSS REFERENCE

The following table cross-references the failure point codes to the test where they can be found. The table consists of groups of ten codes (0-9) per row. An 'x' indicates the code was used in the test indicated for that row in the table.

FP Codes	Test	0	1	2	3	4	5	6	7	8	9
2000	UAL000	x	x	x	x	x	x	x	x	x	x
2010	UAL000	x	x	x	x	x	x	x	x	x	x
2020	UAL000	x	x	x	x	x	x	x	x	x	x
2030	UAL000						x	x	x	x	x
2040	UAL000	x	x	x	x	x	x	x	x	x	x
2050	UAL000	x	x	x	x	x	x	x	x	x	x
2060	UAL000	x	x	x	x	x					
2070	UAL000	x	x	x	x	x	x	x	x	x	x
2080	UAL000	x	x	x	x	x	x	x	x	x	x
2090	UAL000	x	x	x	x	x	x	x	x	x	x
2100											
2100	UAL001							x	x	x	
2110	UAL001		x						x		
2120	UAL001				x		x	x			
2130											
2140	UAL005	x									
2150	UAL005						x	x	x		
2160	UAL005										
2170	UAL005						x	x	x		
2180	UAL005										
2190	UAL005						x	x	x		
2200	UAL005										
2210	UAL005						x	x	x		
2300	UAL006	x									
2310	UAL006										
2320	UAL006	x	x	x	x	x					
2330	UAL006										
2340	UAL006	x	x	x	x	x					
2350	UAL006										
2360	UAL006						x	x	x	x	x
2370	UAL006										
2380	UAL006	x	x	x	x	x					

FP Codes	Test	0	1	2	3	4	5	6	7	8	9
2390	UAL006										
2400	UAL006										
2410	UAL006										
2420	UAL006	x	x	x	x	x					
2510	UAL007	x					x	x	x	x	
2520	UAL007	x									
2530	UAL007			x		x		x		x	x
2620	UAL008	x	x	x							
2630											
2640	UAL008	x	x	x							
2650	UAL008										
2660	UAL008	x	x	x							
2710	UAL009	x									
2720	UAL009	x	x	x	x						
2730	UAL009										
2740	UAL009										
2750	UAL009						x	x			
2760	UAL009										
2770	UAL009										
2780	UAL009						x				
2790	UAL009	x									
2800	UAL009	x									
2810	UAL009										
2820	UAL009						x				
2830	UAL009										
2840	UAL009	x									
2850	UAL009										
2860	UAL009	x									
2870	UAL009										
2880	UAL009	x									

FP Codes	Test	0	1	2	3	4	5	6	7	8	9
2890	UAL009						x		x		
2900	UAL009	x	x								
3000	UAL010						x				
3010	UAL010										
3020	UAL010	x									
3030	UAL010										
3040	UAL010	x									
3050	UAL010						x				
3060	UAL010	x									
3070	UAL010						x				
3080	UAL010										
3090	UAL010	x					x				
3100	UAL010										
3110	UAL010	x									
3120	UAL010						x				
3130	UAL010	x									
3210	UAL011	x	x								
3220	UAL011	x				x	x	x	x	x	
3230	UAL011	x					x				
3240	UAL011						x	x	x	x	x
3250	UAL011		x	x	x	x		x	x	x	x
3260	UAL011	x	x	x	x	x		x	x	x	x
3270	UAL011	x	x	x	x						
3280											
3290											
3300	UAL012	x	x				x				
3310	UAL012	x					x				
3320	UAL012	x					x				

FP Codes	Test	0	1	2	3	4	5	6	7	8	9
3330	UAL012	x					x				
3340	UAL012	x					x				
3350	UAL012	x					x				
3360	UAL012	x					x				
3370	UAL012	x					x				
3380											
3390											
3400	UAL013	x					x				
3410	UAL013	x					x	x			
3420	UAL013	x	x	x			x				
3430	UAL013	x	x	x			x				
3440	UAL013	x	x	x							
3450											
3460											
3470	UAL038	x	x	x	x	x	x	x	x	x	x
3480	UAL038	x	x	x	x	x	x	x	x		
3490	UAL038						x	x	x	x	x
3500	UAL038	x	x	x							
3510											
3520											
3530	UAL48	x	x	x	x	x		x	x	x	x
3540	UAL48	x	x	x	x	x					
3550	UAL48										
3560	UAL48										
5100	Annex A	x	x	x	x						
5110	Annex A	x	x				x	x			
5120	Annex A	x	x	x			x				

ANNEX C. TEST REPORT

The following Test Report must be completed for each Field Device tested.

1. Test Operator

Name		Company	
Title		Address	
Tel. No.			
FAX No.			
EMail			

2. Certification

I hereby affirm that all data provided in this Test Report is accurate and complete.

Signature		Date	
Name			
Title			

3. Test Device Identification

Manufacturer Name:		Model Name(s):	
Manufacture ID Code:	(Hex)	Device Type Code:	(Hex)
Device ID	Hex		
HART Protocol Revision		Device Revision:	
Hardware Revision		Software Revision:	
Device Profile			
Revision Release Date			
Physical Layers Supported		Notes:	
Physical Device Category			

4. Test Data

Test	Result
UAL000 Confirm All Universal Commands Supported	❑ Pass ❑ Fail
UAL001 Read Dynamic Variables (Commands 1, 2, and 3)	❑ Pass ❑ Fail Number of Dynamic Variables =
UAL005 Write Message	❑ Pass ❑ Fail
UAL006 Write Tag Descriptor and Date	❑ Pass ❑ Fail ❑ Invalid Date Detection Supported
UAL007 Verify Command 14 and 15 Response	❑ Pass ❑ Fail ❑ Sensor Limits Not Supported
UAL008 Write Final Assembly Number	❑ Pass ❑ Fail
UAL009 Verify Write Protect	❑ Pass ❑ Fail ❑ Not Applicable
UAL010 Verify Cold Start Bit	❑ Pass ❑ Fail ❑ Separate cold start bits not supported
UAL011 Read Device Variables (Command 9)	❑ Pass ❑ Fail ❑ Tested with Dynamic Variables
UAL012 Read Dynamic Variable Classification	❑ Pass ❑ Fail
UAL013 Write Long Tag	❑ Pass ❑ Fail
UAL038 Reset Configuration Changed Flag	❑ Pass ❑ Fail
UAL048 Read Additional Device Status	❑ Pass ❑ Fail

Attachment 1. Command 48 / More Status Available Test Methodology

The detailed procedure used to confirm the "More Status Available" bit can be properly reset must be attached. In particular, this detailed description must indicate how the More Status Available bit is set (i.e., what error is used to trigger the status change). This procedure is used, along with UAL048, to assess compliance with Command 48 requirements.

ANNEX D. REVISION HISTORY

D.1 Changes from Revision 3.0 to 4.0

This Test Specification is a companion to Revision 7.0 of the *Universal Command Specification*. The principle change to this version of the test specification is to provide support for HART7 in addition to HART 5 and HART 6. Specific changes to individual Tests include:

- Updated cover page, page 2
- UAL007 updated for HART 7
- In UAL009, in Command 38 response processing separated " IF ((RESPONSE_CODE != 0) AND (BYTE_COUNT != 2)) into two IF statements. IF (BYTE_COUNT != 2) then fail
- UAL009 After "SEND Command 9 with Device Variables = 246" changed BC check from 13 to 15.
- UAL011 updated for HART 7
- UAL011 Made checking mandatory device variables (PV-QV, etc) a separate test case.
- Added UAL038 and UAL048 to testing sequence. Changed names to UAL from CAL. Updated references
- UAL038 added test for too few data bytes.
- Replaced "CheckForRecommendedCommand" in UAL038
- Down near the comment " A Primary Master Command 38 is sent without proper matching configuration change counter." Move the check on config changed status inside the "If univ>6" conditional.
- Separated Cmd 38 resets with cfg cntrs out as a separate test case.
- Changed Failure Point Codes in UAL000, UAL038, UAL048
- IdentifyDevice now fails any device returning Universal Revision > 7
- Updated Deliverables

D.2 Changes from Revision 2.0 to 3.0

Test procedures now include support for the Universal Command revision 5 and 6. Thus, manufacturers may use these procedures to check device compatibility with either of these revisions. Section 5.1, Testing Sequence was added as well. Specific changes to individual Tests include:

UAL006 had the date code test values ordered incorrectly. These are now correct.

UAL010 returns a warning if cold start bits are not supported for each master. This has always been a recommended practice.

UAL011 check for a RESPONSE_CODE=0 when a NaN is returned.

UAL013 now allows a "Buffer Overflow" response when 33 bytes are written using Command 22.

IdentifyDevice now fails any device returning Universal Revision > 6

VerifyNotWriteProtected now treats 251 as identical to not write protected.

In addition, a number of minor typo's were also corrected.

D.3 Changes From Revision 1.2 to 2.0

This document was updated with Revision 2.0 to reflect changes to referenced documents and to reformat certain document elements. Specific changes to individual Tests include:

UAL003 and UAL004 have been deleted... Equivalent testing was already performed as part of the Data Link Layer testing.

The Write Command tests (UAL005, UAL006, UAL008, UAL013) now send several different values. In addition, operation of the Configuration Change Counter is confirmed.

Basic "sanity checks" are performed (UAL012) on Unit Code and Device Variable Classifications.

Tests UAL010 – UAL013 were added. These support the universal Commands added in HART 6 plus a new test to confirm the operation of the Cold Start status bit.

UAL006 now sends invalid dates to allow testing of DUT's supporting the new Response Code "Invalid Date Code Detected".

This revision is compatible with Revision 6 of the Universal Command Specification.

D.4 Changes From Revision 1.1 to 1.2

This document has been updated with Revision 1.2 to reflect changes to related documents and update formatting.

Occurrences of the term 'Universal Application Layer Specification' has been changed to either 'Universal Command Specification', 'Common Practice Command Specification' or both, as appropriate.

D.5 Changes From Revision 1.0 to 1.1

This document has been updated with Revision 1.1 to reflect changes to related documents. The reference to the document titled “HART Conformance Test Guidelines” (HCF_LIT-19) was deleted as the document is obsolete. The document titled “HART Slave Data Link Layer, Test Specification” (HCF_TEST-1) was updated to reflect the current revision number. No other changes were made to the document content.